

# The package `nicematrix`\*

F. Pantigny  
fpantigny@wanadoo.fr

April 8, 2021

## Abstract

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

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

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

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

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

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

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

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

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

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

\*This document corresponds to the version 5.14 of `nicematrix`, at the date of 2021/04/08.

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

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

# 1 The environments of this package

The package `nicematrix` defines the following new environments.

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

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

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

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

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

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

## Important

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

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

# 2 The vertical space between the rows

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

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

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

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

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

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

---

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

### 3 The vertical position of the arrays

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

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

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

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

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

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

1. an item

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

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

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

1. an item

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

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

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

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

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

## 4 The blocks

### 4.1 General case

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

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.

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

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

One may wish to raise the size of the “A” placed in the block of the previous example. Since this element is composed in math mode, it's not possible to use directly a command like `\large`, `\Large` and `\LARGE`. That's why the command `\Block` provides an option between angle brackets to specify some TeX code which will be inserted before the beginning of the math mode.<sup>4</sup>

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

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

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

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

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

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

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

---

<sup>4</sup>This argument between angular brackets may also be used to insert a command of font such as `\bfseries` when the command `\\` is used in the content of the block.

- the key `fill` takes in as value a color and fills the block with that color;
- the key `draw` takes in as value a color and strokes the frame of the block with that color (the default value of that key is the current color of the rules of the array);
- the key `color` takes in as value a color and apply that color the content of the block but draws also the frame of the block with that color;
- the key `line-width` is the width (thickness) of the frame (this key should be used only when the key `draw` is in force);
- the key `rounded-corners` requires rounded corners (for the frame drawn by `draw` and the shape drawn by `fill`) with a radius equal to the value of that key (the default value is 4 pt<sup>5</sup>);
- the key `borders` provides the ability to draw only some borders of the blocks; the value of that key is a (comma-separated) list of elements covered by `left`, `right`, `top` and `bottom`;
- **New 5.14** the keys `t` and `b` fix the base line that will be given to the block when it has a multi-line content (the lines are separated by `\\`).

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

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

```
\begin{NiceTabular}{cwc{2cm}wc{3cm}c}
rose      & tulipe & marguerite & dahlia \\
violette
& \Block[draw=red,fill=red!15,rounded-corners]{2-2}{\LARGE De très jolies fleurs}
& & souci \\
pervenche & & lys \\
arum      & iris & jacinthe & muguet
\end{NiceTabular}
```

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

## 4.2 The mono-column blocks

The mono-column blocks have a special behaviour.

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

---

<sup>5</sup>This value is the initial value of the *rounded corners* of Tikz.

```

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

```

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

### 4.3 The mono-row blocks

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

### 4.4 The mono-cell blocks

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

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

- It's possible to use the command `\` in a (mono-cell) block.
- It's possible to use the option of horizontal alignment of the block in derogation of the type of column given in the preamble of the array.
- It's possible to draw a frame around the cell with the key `draw` of the command `\Block` and to fill the background with rounded corners with the keys `fill` and `rounded-corners`.<sup>6</sup>
- It's possible to draw one or several borders of the cell with the key `borders`.

```

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

```

Writer	year of birth
Hugo	1802
Balzac	1799

We recall that if the first mandatory argument of `\Block` is left blank, the block is mono-cell.<sup>7</sup>

### 4.5 A small remark

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

<sup>6</sup>If one simply wishes to color the background of a unique celle, there is no point using the command `\Block`: it's possible to use the command `\cellcolor` (when the key `colortbl-like` is used).

<sup>7</sup>One may consider that the default value of the first mandatory argument of `\Block` is 1-1.

```

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

```

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

## 5 The rules

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

### 5.1 Some differences with the classical environments

#### 5.1.1 The vertical rules

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

```

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

```

First	Second
Peter	
Mary	George

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

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

```

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

```

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

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

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

However, in this case, it is probably more clever to add a command `\OnlyMainNiceMatrix` (cf. p. 37):

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

### 5.1.2 The command `\cline`

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

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

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

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

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

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

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

## 5.2 The thickness and the color of the rules

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

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

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

```
\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 37.

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

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

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



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

- These blocks are:
  - the blocks created by the command `\Block`<sup>8</sup> presented p. 4;
  - the blocks implicitly delimited by the continuous dotted lines created by `\Cdots`, `Vdots`, etc. (cf. p. 18).
- The corners are created by the key `corners` explained below (see p. 9).

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

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

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

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

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

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

### 5.3.2 The key `hvlines`

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

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

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

### 5.3.3 The (empty) corners

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

For each of these corners, we will call *empty corner* (or simply *corner*) the reunion of all the empty rectangles starting from the cell actually in the corner of the array.<sup>9</sup>

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

<sup>8</sup>And also the command `\multicolumn` also it's recommended to use instead `\Block` in the environments of `nicematrix`.

<sup>9</sup>For sake of completeness, we should also say that a cell contained in a block (even an empty cell) is not taken into account for the determination of the corners. That behaviour is natural.

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

```

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

```

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

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

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

## 5.4 The command `\diagbox`

```

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

```

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

---

<sup>10</sup>The author of this document considers that type of construction as graphically poor.

## 5.5 Dotted rules

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

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

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

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

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

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

It's possible to change in `nicematrix` the letter used to specify a vertical dotted line with the option `letter-for-dotted-lines` available in `\NiceMatrixOptions`. Thus released, the letter “:” can be used otherwise (for example by the package `arydshln`<sup>11</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>12</sup>. In `nicematrix`, the dotted lines drawn by `\hdottedline` and “:” do likewise.

## 6 The color of the rows and columns

### 6.1 Use of `colortbl`

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

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

However, there is two drawbacks:

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

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

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

<sup>11</sup>However, one should remark that the package `arydshln` is not fully compatible with `nicematrix`.

<sup>12</sup>In fact, with `array`, this is true only for `\hline` and “|” but not for `\cline`: cf p. 8

## 6.2 The tools of nicematrix in the \CodeBefore

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

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

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

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

New commands are available in that `\CodeBefore`: `\cellcolor`, `\rectanglecolor`, `\rowcolor`, `\columncolor`, `\rowcolors`, `\chessboardcolors` and `arraycolor`.<sup>13</sup>

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

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

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

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

a	b	c
e	f	g
h	i	j

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

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

a	b	c
e	f	g
h	i	j

<sup>13</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. 35.

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

```

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

```

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

- The command `\columncolor` takes its name from the command `\columncolor` of `colortbl`. Its syntax is similar to the syntax of `\rowcolor`.
- The command `\rowcolors` (with a  $s$ ) takes its name from the command `\rowcolors` of `xcolor`<sup>14</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>15</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).

<sup>14</sup>The command `\rowcolors` of `xcolor` is available when `xcolor` is loaded with the option `table`. That option also loads the package `colortbl`.

<sup>15</sup>Otherwise, the color of a given row relies only upon the parity of its absolute number.

```

\begin{NiceTabular}{clr}[hvlines]
\CodeBefore
  \rowcolors{2}{blue!10}{}[cols=2-3,restart]
\Body
\Block{1-*}{Results} \\\
John & 12 \\\
Stephen & 8 \\\
Sarah & 18 \\\
Ashley & 20 \\\
Henry & 14 \\\
Madison & 15
\end{NiceTabular}

```

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

```

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

```

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

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

```

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

```

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

We have used the key `r` which aligns all the columns rightwards (cf. p. 30).

- The command `\arraycolor` takes in as mandatory argument a color and color the whole tabular with that color (excepted the potential exterior rows and columns: cf. p. 17). This command is useful thanks to its key `except-corners` (in the third argument, which is a optional argument between square brackets). The definition of these “corners” has been given p. 9 when we have presented the key `corners`.

```

\begin{NiceTabular}{*{6}{c}}[cell-space-top-limit=3pt]
\CodeBefore
  \arraycolor{blue!10}[except-corners=NE]
\Body
1\\
1&1\\
1&2&1\\
1&3&3&1\\
1&4&6&4&1\\
1&5&10&10&5&1
\end{NiceTabular}

```

1					
1	1				
1	2	1			
1	3	3	1		
1	4	6	4	1	
1	5	10	10	5	1

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

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

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

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

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

It's possible to access the preceding tools with a syntax close to the syntax of `colortbl`. For that, one must use the key `colortbl-like` in the current environment.<sup>16</sup>

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

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

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

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

<sup>16</sup>As for now, this key is *not* available in `\NiceMatrixOptions`.

## 7 The width of the columns

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

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

Paris	New York	Madrid
Berlin	London	Roma
Rio	Tokyo	Oslo

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

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

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

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

It's possible to give the special value `auto` to the option `columns-width`: all the columns of the array will have a width equal to the widest cell of the array.<sup>17</sup>

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

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

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

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

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

But it's also possible to fix a zone where all the matrices will have their columns of the same width, equal to the widest cell of all the matrices. This construction uses the environment `{NiceMatrixBlock}` with the option `auto-columns-width`<sup>18</sup>. The environment `{NiceMatrixBlock}` has no direct link with the command `\Block` presented previously in this document (cf. p. 4).

<sup>17</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>18</sup>At this time, this is the only usage of the environment `{NiceMatrixBlock}` but it may have other usages in the future.



```

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

```

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

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

## 8 The exterior rows and columns

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

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

```

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

```

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

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

We have several remarks to do.

- For the environments with an explicit preamble (i.e. `{NiceTabular}`, `{NiceArray}` and its variants), no letter must be given in that preamble for the potential first column and the potential last column: they will automatically (and necessarily) be of type `r` for the first column and `l` for the last one.<sup>19</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. 32) is used, `nicematrix` has, in any case, to load the whole body of the environment (and that’s why it’s not possible to put verbatim material in the array with the option `light-syntax`). The analysis of this whole body gives the number of rows (but not the number of columns).

<sup>19</sup>The users wishing exterior columns with another type of alignment should consider the command `\SubMatrix` available in the `\CodeAfter` (cf. p. 24).

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

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

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

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

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

#### Remarks

- As shown in the previous example, the horizontal and vertical rules doesn't extend in the exterior rows and columns.
- However, if one wishes to define new specifiers for columns in order to draw vertical rules (for example thicker than the standard rules), he should consider the command `\OnlyMainNiceMatrix` described on page 37.
- A specification of color present in `code-for-first-row` also applies to a dotted line draw in this exterior "first row" (excepted if a value has been given to `xdots/color`). Idem for the other exterior rows and columns.
- Logically, the potential option `columns-width` (described p. 16) doesn't apply to the "first column" and "last column".
- For technical reasons, it's not possible to use the option of the command `\` after the "first row" or before the "last row". The placement of the delimiters would be wrong. If you are looking for a workaround, consider the command `\SubMatrix` in the `\CodeAfter` described p. 24.

## 9 The continuous dotted lines

Inside the environments of the package `nicematrix`, new commands are defined: `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, and `\Iddots`. These commands are intended to be used in place of `\dots`, `\cdots`,

`\vdots`, `\ddots` and `\iddots`.<sup>20</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>21</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>22</sup>

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

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

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

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

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

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

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

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

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

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

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

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

There are also other means to change the size of the matrix. Someone might want to use the optional argument of the command `\` for the vertical dimension and a command `\hspace*` in a cell for the horizontal dimension.<sup>23</sup>

<sup>20</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>21</sup>The precise definition of a “non-empty cell” is given below (cf. p. 38).

<sup>22</sup>It's also possible to change the color of all theses dotted lines with the option `xdots/color` (`xdots` to remind that it works for `\Cdots`, `\Ldots`, `\Vdots`, etc.): cf. p. 22.

<sup>23</sup>In `nicematrix`, one should use `\hspace*` and not `\hspace` for such an usage because `nicematrix` loads `array`. One may also remark that it's possible to fix the width of a column by using the environment `{NiceArray}` (or one of its variants) with a column of type `w` or `W`: see p. 16

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

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

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

## 9.1 The option `nullify-dots`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```


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


```

Remark: Unlike the command `\hdotsfor` of `amsmath`, the command `\Hdotsfor` may be used even when the package `colortbl`<sup>24</sup> is loaded (but you might have problem if you use `\rowcolor` on the same row as `\Hdotsfor`).

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

```

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

```

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

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

Imagine you have a document with a great number of mathematical matrices with ellipsis. You may wish to use the dotted lines of `nicematrix` without having to modify the code of each matrix. It's possible with the keys. `renew-dots` and `renew-matrix`.<sup>25</sup>

<sup>24</sup>We recall that when `xcolor` is loaded with the option `table`, the package `colortbl` is loaded.

<sup>25</sup>The options `renew-dots`, `renew-matrix` can be fixed with the command `\NiceMatrixOptions` like the other options. However, they can also be fixed as options of the command `\usepackage`. There is also a key `transparent` which is an alias for the conjunction of `renew-dots` and `renew-matrix` but it must be considered as obsolete.

- The option `renew-dots`

With this option, the commands `\ldots`, `\cdots`, `\vdots`, `\ddots`, `\iddots`<sup>20</sup> and `\hdotsfor` are redefined within the environments provided by `nicematrix` and behave like `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots` and `\Hdotsfor`; the command `\dots` (“automatic dots” of `amsmath`) is also redefined to behave like `\Ldots`.

- The option `renew-matrix`

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

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

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

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

## 9.4 The labels of the dotted lines

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

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

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

## 9.5 Customisation of the dotted lines

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

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

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

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

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

### The option `xdots/color`

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

### The option `xdots/shorten`

The option `xdots/shorten` fixes the margin of both extremities of the line. The name is derived from the options “`shorten >`” and “`shorten <`” of Tikz but one should notice that `nicematrix` only provides `xdots/shorten`. The initial value of this parameter is 0.3 em (it is recommended to use a unit of length dependent of the current font).

### The option `xdots/line-style`

It should be pointed that, by default, the lines drawn by Tikz with the parameter `dotted` are composed of square dots (and not rounded ones).<sup>26</sup>

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

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

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

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

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

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

## 9.6 The dotted lines and the rules

The dotted lines determine virtual blocks which have the same behaviour regarding the rules (the rules specified by the specifier `|` in the preamble, by the command `\Hline` and by the keys `hlines`, `vlines` and `hvlines` are not drawn within the blocks).<sup>27</sup>

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

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

<sup>26</sup>The first reason of this behaviour is that the PDF format includes a description for dashed lines. The lines specified with this descriptor are displayed very efficiently by the PDF readers. It’s easy, starting from these dashed lines, to create a line composed by square dots whereas a line of rounded dots needs a specification of each dot in the PDF file.

<sup>27</sup>On the other side, the command `\line` in the `\CodeAfter` (cf. p. 24) does *not* create block.

## 10 The `\CodeAfter`

The option `code-after` may be used to give some code that will be executed *after* the construction of the matrix.<sup>28</sup>

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

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

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

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

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

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

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

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

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

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

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

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

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

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

<sup>28</sup>There is also a key `code-before` described p. 12.



- the second argument is the upper-left corner of the submatrix with the syntax  $i-j$  where  $i$  the number of row and  $j$  the number of column;
- the third argument is the lower-right corner with the same syntax;
- the fourth argument is the right delimiter;
- the last argument, which is optional, is a list of key-value pairs.<sup>29</sup>

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

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

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

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

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

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

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

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

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

<sup>29</sup>There is no optional argument between square brackets in first position because a square bracket just after `\SubMatrix` must be interpreted as the first (mandatory) argument of the command `\SubMatrix`: that bracket is the left delimiter of the sub-matrix to construct (eg.: `\SubMatrix[{2-2}{4-7}]`).

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

```

\begin{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. 36.

## 11 The notes in the tabulars

### 11.1 The footnotes

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

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

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

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

### 11.2 The notes of tabular

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

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

```

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

```

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

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

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

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

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

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

### 11.3 Customisation of the tabular notes

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

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

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

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

We detail these keys.

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

Initial value: `false`

That key is also available within a given environment.

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

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

Some text before the notes.

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

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

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

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

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

Initial value: `false`

That key is also available within a given environment.

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

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

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

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

Initial value: `\textsuperscript{#1}`

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

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

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

Initial value: `\textsuperscript{#1}`

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

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

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

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

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

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

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

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

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

Initial value: `empty`

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

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

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

For an example of customisation of the tabular notes, see p. 39.

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

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

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

## 12 Other features

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

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

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

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

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

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

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

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

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

### 12.3 The command `\rotate`

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

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

```

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

```

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

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

```

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

```

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

## 12.4 The option `small`

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

```

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

```

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

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

## 12.5 The counters iRow and jCol

In the cells of the array, it's possible to use the LaTeX counters `iRow` and `jCol` which represent the number of the current row and the number of the current column<sup>30</sup>. Of course, the user must not change the value of these counters which are used internally by `nicematrix`.

In the `code-before` (cf. p. 12) and in the `\CodeAfter` (cf. p. 24), `iRow` represents the total number of rows (excepted the potential exterior rows) and `jCol` represents the total number of columns (excepted the potential exterior columns).

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

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

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

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

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

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

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

## 12.6 The option light-syntax

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

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

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

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

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

When the option `light-syntax` is used, it is not possible to put verbatim material (for example with the command `\verb`) in the cells of the array.<sup>31</sup>

<sup>30</sup>We recall that the exterior “first row” (if it exists) has the number 0 and that the exterior “first column” (if it exists) has also the number 0.

<sup>31</sup>The reason is that, when the option `light-syntax` is used, the whole content of the environment is loaded as a TeX argument to be analyzed. The environment doesn't behave in that case as a standard environment of LaTeX which only put TeX commands before and after the content.



## 12.7 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`.<sup>32</sup>

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

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

## 12.8 The environment `{NiceArrayWithDelims}`

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

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

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

# 13 Use of Tikz with nicematrix

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

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

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

However, it's possible to impose the creation of a node with the command `\NotEmpty`.<sup>33</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 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) ;
```

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

<sup>32</sup>`delimiters-color` is a synonymous (deprecated) for `delimiters/color`.

<sup>33</sup>One should note that, with that command, the cell is considered as non-empty, which has consequences for the continuous dotted lines (cf. p. 18) and the computation of the "corners" (cf. p. 9).

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. 45).

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

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

In fact, the package `nicematrix` can create “extra nodes”: the “medium nodes” and the “large nodes”. The first ones are created with the option `create-medium-nodes` and the second ones with the option `create-large-nodes`.<sup>34</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>35</sup>

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

The “large nodes” of the first column and last column may appear too small for some usage. That's why it's possible to use the options `left-margin` and `right-margin` to add space on both sides of the array and also space in the “large nodes” of the first column and last column. In the following example, we have used the options `left-margin` and `right-margin`.<sup>36</sup>

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

<sup>34</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>35</sup>There is no “large nodes” created in the exterior rows and columns (for these rows and columns, cf. p. 17).

<sup>36</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.

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

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

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

Here is an array composed with the following code:

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

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

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

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

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

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

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

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

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

	tulipe	lys
arum		violette mauve
muguet	dahlia	

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

```
\begin{NiceMatrix}
\CodeBefore
\tikz \draw [fill=red!15] (7-|4) |- (8-|5) |- (9-|6) |- cycle ;
\Body
1 \\\
1 & 1 \\\
```

```

1 & 2 & 1 \\
1 & 3 & 3 & 1 \\
1 & 4 & 6 & 4 & 1 \\
1 & 5 & 10 & 10 & 5 & 1 \\
1 & 6 & 15 & 20 & 15 & 6 & 1 \\
1 & 7 & 21 & 35 & 35 & 21 & 7 & 1 \\
1 & 8 & 28 & 56 & 70 & 56 & 28 & 8 & 1
\end{NiceMatrix}

```

```

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1

```

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

```

 $\begin{pNiceArray}{ccc|c}$ 

```

```

2 & 1 & 3 & 0 \\

```

```

3 & 3 & 1 & 0 \\

```

```

3 & 3 & 1 & 0

```

```

\CodeAfter

```

```

\tikz \draw [red] (3.5-|1) -- (3.5-|last) ;

```

```

\end{pNiceArray}$

```

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

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

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

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

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

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

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

## 14 API for the developpers

The package `nicematrix` provides two variables which are internal but public<sup>37</sup>:

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

<sup>37</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.

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

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

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

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

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

Here is an example of use:

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

Tokyo	Paris	London
Lima	Oslo	Miami
Los Angeles	Madrid	Roma

## 15 Technical remarks

### 15.1 Definition of new column types

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

For example, one may wish to define a new column type `?` in order to draw a (black) heavy rule of width 1 pt. The following definition will do the job<sup>38</sup>:

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

The heavy vertical rule won’t extend in the exterior rows.<sup>39</sup>

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

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

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.

<sup>38</sup>The command `\vrule` is a TeX (and not LaTeX) command.

<sup>39</sup>Of course, such rule is defined by the classical technics of `nicematrix` and, for this reason, won’t cross the double rules of `\hline``\hline`.

## 15.2 Diagonal lines

By default, all the diagonal lines<sup>40</sup> of a same array are “parallelized”. That means that the first diagonal line is drawn and, then, the other lines are drawn parallel to the first one (by rotation around the left-most extremity of the line). That’s why the position of the instructions `\Ddots` in the array can have a marked effect on the final result.

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

Example with parallelization (default):

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

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

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

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

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

The same example without parallelization:

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

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

## 15.3 The “empty” cells

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

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

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

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

- Each cell whose TeX output has a width equal to zero is empty.
- A cell containing the command `\NotEmpty` is not empty (and a PGF/Tikz node) is created in that cell.

<sup>40</sup>We speak of the lines created by `\Ddots` and not the lines created by a command `\line` in `code-after`.

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

## 15.4 The option `exterior-arraycolsep`

The environment `{array}` inserts an horizontal space equal to `\arraycolsep` before and after each column. In particular, there is a space equal to `\arraycolsep` before and after the array. This feature of the environment `{array}` was probably not a good idea<sup>41</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>42</sup>. The package `nicematrix` does the same in all its environments, `{NiceArray}` included. However, if the user wants the environment `{NiceArray}` behaving by default like the environment `{array}` of `array` (for example, when adapting an existing document) it's possible to control this behaviour with the option `exterior-arraycolsep`, set by the command `\NiceMatrixOptions`. With this option, exterior spaces of length `\arraycolsep` will be inserted in the environments `{NiceArray}` (the other environments of `nicematrix` are not affected).

## 15.5 Incompatibilities

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

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

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

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

# 16 Examples

## 16.1 Notes in the tabulars

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

Let's consider that we wish to number the notes of a tabular with stars.<sup>43</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>44</sup>

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

<sup>41</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>42</sup>And not by inserting `@{}` on both sides of the preamble of the array. As a consequence, the length of the `\hline` is not modified and may appear too long, in particular when using square brackets.

<sup>43</sup>Of course, it's realistic only when there is very few notes in the tabular.

<sup>44</sup>In fact: the value of its argument.

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

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

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

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

\*Achard is an old family of the Poitou.

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



## 16.2 Dotted lines

An example with the resultant of two polynoms:

```

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

```

An example for a linear system:

$$\begin{array}{ccccccc} 1 & & & & & & \\ 0 & & & & & & \\ 0 & & & & & & \\ & & & & & & \\ \text{\Vdots} & & & & & & \\ 0 & & & & & & \\ \end{array}$$

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

### 16.3 Dotted lines which are no longer dotted

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

```
\NiceMatrixOptions{code-for-first-row = \scriptstyle,code-for-first-col = \scriptstyle }
\setcounter{MaxMatrixCols}{12}
\newcommand{\blue}{\color{blue}}
[\begin{pNiceMatrix}[last-row,last-col,nullify-dots,xdots/line-style={dashed,blue}]
```

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

```

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

```

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

## 42

```

\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}

```

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

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

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

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

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

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

```

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

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

...
\end{NiceMatrixBlock}

```

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

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

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

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

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

```

```

\SubMatrix({5-1}{8-5})
\SubMatrix({9-1}{11-5})
\SubMatrix({12-1}{13-5})
\end{NiceMatrix}\]

```

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

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

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

## 16.5 How to highlight cells of a matrix

In order to highlight a cell of a matrix, it's possible to “draw” that cell with the key **draw** of the command `\Block` (this is one of the uses of a mono-cell block<sup>45</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 command `\hline`, the specifier “|” and the options `hlines`, `vlines` and `hvlines` spread the cells.<sup>46</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} \\\

```

<sup>45</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>46</sup>For the command `\cline`, see the remark p. 8.

```
A_{41} & A_{42} & A_{43} & \rowcolor{red!15}A_{44}
\end{pNiceArray}
```

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

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

**Caution** : Some PDF readers are not able to show transparency.<sup>47</sup>

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

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

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

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

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

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

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

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

<sup>47</sup>In Overleaf, the “built-in” PDF viewer does not show transparency. You can switch to the “native” viewer in that case.

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

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

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

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

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

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

We obtain the following matrix.

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

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

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

We obtain the following matrix.

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

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

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

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

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

$$\begin{array}{c}
 \begin{array}{c} L_i \end{array} \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{i1} & \cdots & a_{in} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} \vdots \\ \vdots \\ c_{ij} \\ \vdots \end{pmatrix} \\
 \begin{pmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & & \vdots \\ b_{kj} & & \vdots \\ \vdots & & \vdots \\ b_{n1} & \cdots & b_{nn} \end{pmatrix} \begin{array}{c} C_j \end{array}
 \end{array}$$

## 17 Implementation

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

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

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

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

### Declaration of the package and packages loaded

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

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

`<@@=nicematrix>`

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

```

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

```

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

```

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

```

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

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



```

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

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

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

```

## Technical definitions

```

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

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

```

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

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

```

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

```

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

```

50 \bool_new:N \c_@@_revtex_bool
51 \ifclassloaded { revtex4-1 }
52 { \bool_set_true:N \c_@@_revtex_bool }

```

```

53 { }
54 \ifclassloaded { revtex4-2 }
55 { \bool_set_true:N \c_@@_revtex_bool }
56 { }

```

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

```

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

```

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

```

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

```

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

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

```

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

```

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

```

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

```

88 \bool_new:N \c_@@_colortbl_loaded_bool
89 \AtBeginDocument
90 {
91   \ifpackageloaded { colortbl }
92   { \bool_set_true:N \c_@@_colortbl_loaded_bool }
93   {

```

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

```

94   \cs_set_protected:Npn \CT@arc@ { }

```

```

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

```

Idem for \CT@drs@.

```

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

```

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

```

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

```

The following \skip\_horizontal:N \c\_zero\_dim is to prevent a potential \unskip to delete the \leaders<sup>48</sup>

```

129     \skip_horizontal:N \c_zero_dim
130 }

```

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

```

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

```

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

```

135 \cs_set:Npn \@@_cline

```

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

```

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

```

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

---

<sup>48</sup>See question 99041 on TeX StackExchange.

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

The following command is a small shortcut.

```

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

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

163 \cs_set_eq:NN \@@_old_pgfpaintanchor \pgfpaintanchor

```

## The column S of siunitx

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

```

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

```

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

```

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

```

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

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

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

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

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

```
177   \cs_set_eq:NN \NC@find \prg_do_nothing:
178   \NC@rewrite@S { }
```

Conversion of the *toks* `\@temptokena` in a token list of `expl3` (the *toks* are not supported by `expl3` but we can, nevertheless, use the option `V` for `\tl_gset:NV`).

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

The token lists `\c_@@_table_collect_begin_tl` and `\c_@@_table_print_tl` contain now the two commands of `siunitx`.

If the adaptation has been done, the command `\@@_adapt_S_column:` becomes no-op (globally).

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

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

```
189 \AtBeginDocument
190 {
191   \bool_if:nTF { ! \c_@@_siunitx_loaded_bool }
192   { \cs_set_eq:NN \@@_renew_NC@rewrite@S: \prg_do_nothing: }
193   {
194     \cs_new_protected:Npn \@@_renew_NC@rewrite@S:
195     {
196       \renewcommand*{\NC@rewrite@S}[1] []
197       {
198         \@temptokena \exp_after:wN
```

```

199         {
200             \tex_the:D \@temptokena
201             > { \c_@@_Cell: \c_@@_table_collect_begin_t1 S {##1} }
\@@_true_c: will be replaced statically by c at the end of the construction of the preamble.
202             \@@_true_c:
203             < { \c_@@_table_print_t1 \@@_end_Cell: }
204         }
205     \NC@find
206 }
207 }
208 }
209 }

```

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

```

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

```

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

```

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

```

## Parameters

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

```

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

```

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

```

229 \int_new:N \g_@@_env_int

```

The following command is only a syntactic shortcut. It must *not* be protected (it will be used in names of PGF nodes).

```

230 \cs_new:Npn \@@_env: { nm - \int_use:N \g_@@_env_int }

```

The command `\NiceMatrixLastEnv` is not used by the package `nicematrix`. It's only a facility given to the final user. It gives the number of the last environment (in fact the number of the current environment but it's meant to be used after the environment in order to refer to that environment — and its nodes — without having to give it a name). This command *must* be expandable since it will be used in `pgf` nodes.

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

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

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

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

```
235 \int_new:N \g_@@_NiceMatrixBlock_int
```

The dimension `\l_@@_columns_width_dim` will be used when the options specify that all the columns must have the same width (but, if the key `columns-width` is used with the special value `auto`, the boolean `\l_@@_auto_columns_width_bool` also will be raised).

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

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

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

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

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

Idem pour the mono-row blocks.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The letter used for the 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
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

We also manage a sequence of the *positions* of the blocks. Of course, it’s redundant with the previous sequence, but it’s for efficiency. In that sequence, each block is represented by only the four first components: `{imin}{jmin}{imax}{jmax}`.

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

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

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

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

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

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

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

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`).

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

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

```
292 \int_new:N \l_@@_row_min_int
293 \int_new:N \l_@@_row_max_int
294 \int_new:N \l_@@_col_min_int
295 \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.

```
296 \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).

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

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

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

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

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

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

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

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

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

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

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

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

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

310 \dim_new:N \l_@@_submatrix_extra_height_dim
311 \dim_new:N \l_@@_submatrix_left_xshift_dim
312 \dim_new:N \l_@@_submatrix_right_xshift_dim
313 \clist_new:N \l_@@_hlines_clist
314 \clist_new:N \l_@@_vlines_clist
315 \clist_new:N \l_@@_submatrix_hlines_clist
316 \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.

```
317 \int_new:N \l_@@_first_row_int
318 \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.

```
319 \int_new:N \l_@@_first_col_int
320 \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).

```
321 \int_new:N \l_@@_last_row_int
322 \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>49</sup>

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

Idem for `\l_@@_last_col_without_value_bool`

```
324 \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.

```
325 \int_new:N \l_@@_last_col_int
326 \int_set:Nn \l_@@_last_col_int { -2 }
```

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

```
\begin{pNiceArray}{cc}[last-col]
1 & 2 \\\
3 & 4
\end{pNiceArray}
```

In such a code, the “last column” specified by the key `last-col` is not used. We want to be able to detect such a situation and we create a boolean for that job.

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

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

## The command `\tablarnote`

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

```
328 \newcounter { tablarnote }
```

---

<sup>49</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.

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

```
329 \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.

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

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

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

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

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

```
333 \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`.

```
334 \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`.

```
335 \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`).

```
336 \AtBeginDocument
337 {
338   \bool_if:nTF { ! \c_@@_enumitem_loaded_bool }
339   {
340     \NewDocumentCommand \tabularnote { m }
341     { \@@_error:n { enumitem-not-loaded } }
342   }
343 }
```

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.

```
344 \newlist { tabularnotes } { enumerate } { 1 }
345 \setlist [ tabularnotes ]
346 {
347   topsep = 0pt ,
348   noitemsep ,
349   leftmargin = * ,
350   align = left ,
351   labelsep = 0pt ,
352   label =
353   \@@_notes_label_in_list:n { \@@_notes_style:n { tabularnotesi } } ,
354 }
355 \newlist { tabularnotes* } { enumerate* } { 1 }
356 \setlist [ tabularnotes* ]
357 {
358   afterlabel = \nobreak ,
359   itemjoin = \quad ,
360   label =
361   \@@_notes_label_in_list:n { \@@_notes_style:n { tabularnotes*i } }
362 }
```

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>50</sup>

```

363     \NewDocumentCommand \tabularnote { m }
364     {
365         \bool_if:nTF { ! \l_@@_NiceArray_bool && \l_@@_in_env_bool }
366         { \@@_error:n { tabularnote~forbidden } }
367         {

```

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

```

368         \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`.

```

369         \seq_gput_right:Nn \g_@@_tabularnotes_seq { #1 }
370         \peek_meaning:NF \tabularnote
371         {

```

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.

```

372         \hbox_set:Nn \l_tmpa_box
373         {

```

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

```

374         \@@_notes_label_in_tabular:n
375         {
376             \stepcounter { tabularnote }
377             \@@_notes_style:n { tabularnote }
378             \prg_replicate:nn { \l_@@_number_of_notes_int - 1 }
379             {
380                 ,
381                 \stepcounter { tabularnote }
382                 \@@_notes_style:n { tabularnote }
383             }
384         }
385     }

```

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.

```

386         \addtocounter { tabularnote } { -1 }
387         \refstepcounter { tabularnote }
388         \int_zero:N \l_@@_number_of_notes_int
389         \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).

```

390         \skip_horizontal:n { \box_wd:N \l_tmpa_box }
391     }
392 }
393 }

```

---

<sup>50</sup>We should try to find a solution to that problem.

```

394     }
395 }

```

## 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.

```

396 \cs_new_protected:Npn \@@_pgf_rect_node:nnnnn #1 #2 #3 #4 #5
397 {
398   \begin { pgfscope }
399   \pgfset
400   {
401     outer~sep = \c_zero_dim ,
402     inner~sep = \c_zero_dim ,
403     minimum~size = \c_zero_dim
404   }
405   \pgftransformshift { \pgfpoint { 0.5 * ( #2 + #4 ) } { 0.5 * ( #3 + #5 ) } }
406   \pgfnode
407   { rectangle }
408   { center }
409   {
410     \vbox_to_ht:nn
411     { \dim_abs:n { #5 - #3 } }
412     {
413       \vfill
414       \hbox_to_wd:nn { \dim_abs:n { #4 - #2 } } { }
415     }
416   }
417   { #1 }
418   { }
419   \end { pgfscope }
420 }

```

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

```

421 \cs_new_protected:Npn \@@_pgf_rect_node:nnn #1 #2 #3
422 {
423   \begin { pgfscope }
424   \pgfset
425   {
426     outer~sep = \c_zero_dim ,
427     inner~sep = \c_zero_dim ,
428     minimum~size = \c_zero_dim
429   }
430   \pgftransformshift { \pgfpoint scale { 0.5 } { \pgfpointadd { #2 } { #3 } } }
431   \pgfpointdiff { #3 } { #2 }
432   \pgfgetlastxy \l_tmpa_dim \l_tmpb_dim
433   \pgfnode
434   { rectangle }
435   { center }
436   {
437     \vbox_to_ht:nn
438     { \dim_abs:n \l_tmpb_dim }
439     { \vfill \hbox_to_wd:nn { \dim_abs:n \l_tmpa_dim } { } }
440   }
441   { #1 }
442   { }
443   \end { pgfscope }
444 }

```

## 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.

```
445 \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`.

```
446 \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`).

```
447 \dim_new:N \l_@@_cell_space_top_limit_dim
448 \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.

```
449 \dim_new:N \l_@@_inter_dots_dim
450 \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).

```
451 \dim_new:N \l_@@_xdots_shorten_dim
452 \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.

```
453 \dim_new:N \l_@@_radius_dim
454 \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.

```
455 \tl_new:N \l_@@_xdots_line_style_tl
456 \tl_const:Nn \c_@@_standard_tl { standard }
457 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
```

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

```
458 \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).

```
459 \tl_new:N \l_@@_baseline_tl
460 \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`).

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

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

```
462 \bool_new:N \l_@@_parallelize_diags_bool
463 \bool_set_true:N \l_@@_parallelize_diags_bool
```

The following parameter correspond to the key `corners`.

```
464 \clist_new:N \l_@@_corners_clist
465 \dim_new:N \l_@@_notes_above_space_dim
466 \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.).

```
467 \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).

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

The string `\l_@@_name_str` will contain the optional name of the environment: this name can be used to access to the Tikz nodes created in the array from outside the environment.

```
469 \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”.

```
470 \bool_new:N \l_@@_medium_nodes_bool
471 \bool_new:N \l_@@_large_nodes_bool
```

The dimension `\l_@@_left_margin_dim` correspond to the option `left-margin`. Idem for the right margin. These parameters are involved in the creation of the “medium nodes” but also in the placement of the delimiters and the drawing of the horizontal dotted lines (`\hdottedline`).

```
472 \dim_new:N \l_@@_left_margin_dim
473 \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`.

```
474 \dim_new:N \l_@@_extra_left_margin_dim
475 \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.

```
476 \tl_new:N \l_@@_end_of_row_tl
477 \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 “:”.

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

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

```
479 \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 function 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.

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

```
481 \keys_define:nn { NiceMatrix / xdots }
482 {
483   line-style .code:n =
484   {
485     \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.

```
486     { \cs_if_exist_p:N \tikzpicture }
487     { \str_if_eq_p:nn { #1 } { standard } }
488     { \tl_set:Nn \l_@@_xdots_line_style_tl { #1 } }
489     { \@@_error:n { bad-option-for-line-style } }
490   } ,
491   line-style .value_required:n = true ,
492   color .tl_set:N = \l_@@_xdots_color_tl ,
493   color .value_required:n = true ,
494   shorten .dim_set:N = \l_@@_xdots_shorten_dim ,
495   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 `_`.

```
496   down .tl_set:N = \l_@@_xdots_down_tl ,
497   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).

```
498   draw-first .code:n = \prg_do_nothing: ,
499   unknown .code:n = \@@_error:n { Unknown-key-for-xdots }
500 }
```

```
501 \keys_define:nn { NiceMatrix / rules }
502 {
503   color .tl_set:N = \l_@@_rules_color_tl ,
504   color .value_required:n = true ,
505   width .dim_set:N = \arrayrulewidth ,
506   width .value_required:n = true
507 }
```

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

```
508 \keys_define:nn { NiceMatrix / Global }
509 {
510   rules .code:n = \keys_set:nn { NiceMatrix / rules } { #1 } ,
511   rules .value_required:n = true ,
512   standard-cline .bool_set:N = \l_@@_standard_cline_bool ,
513   standard-cline .default:n = true ,
514   cell-space-top-limit .dim_set:N = \l_@@_cell_space_top_limit_dim ,
```

```

515 cell-space-top-limit .value_required:n = true ,
516 cell-space-bottom-limit .dim_set:N = \l_@@_cell_space_bottom_limit_dim ,
517 cell-space-bottom-limit .value_required:n = true ,
518 cell-space-limits .meta:n =
519 {
520     cell-space-top-limit = #1 ,
521     cell-space-bottom-limit = #1 ,
522 } ,
523 cell-space-limits .value_required:n = true ,
524 xdots .code:n = \keys_set:nn { NiceMatrix / xdots } { #1 } ,
525 light-syntax .bool_set:N = \l_@@_light_syntax_bool ,
526 light-syntax .default:n = true ,
527 end-of-row .tl_set:N = \l_@@_end_of_row_tl ,
528 end-of-row .value_required:n = true ,
529 first-col .code:n = \int_zero:N \l_@@_first_col_int ,
530 first-row .code:n = \int_zero:N \l_@@_first_row_int ,
531 last-row .int_set:N = \l_@@_last_row_int ,
532 last-row .default:n = -1 ,
533 code-for-first-col .tl_set:N = \l_@@_code_for_first_col_tl ,
534 code-for-first-col .value_required:n = true ,
535 code-for-last-col .tl_set:N = \l_@@_code_for_last_col_tl ,
536 code-for-last-col .value_required:n = true ,
537 code-for-first-row .tl_set:N = \l_@@_code_for_first_row_tl ,
538 code-for-first-row .value_required:n = true ,
539 code-for-last-row .tl_set:N = \l_@@_code_for_last_row_tl ,
540 code-for-last-row .value_required:n = true ,
541 hlines .clist_set:N = \l_@@_hlines_clist ,
542 vlines .clist_set:N = \l_@@_vlines_clist ,
543 hlines .default:n = all ,
544 vlines .default:n = all ,
545 vlines-in-sub-matrix .code:n =
546 {
547     \tl_if_single_token:nTF { #1 }
548     { \tl_set:Nn \l_@@_letter_vlism_tl { #1 } }
549     { \@@_error:n { One-letter-allowed } }
550 } ,
551 vlines-in-sub-matrix .value_required:n = true ,
552 hvlines .code:n =
553 {
554     \clist_set:Nn \l_@@_vlines_clist { all }
555     \clist_set:Nn \l_@@_hlines_clist { all }
556 } ,
557 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.

```

558 renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
559 renew-dots .value_forbidden:n = true ,
560 nullify-dots .bool_set:N = \l_@@_nullify_dots_bool ,
561 create-medium-nodes .bool_set:N = \l_@@_medium_nodes_bool ,
562 create-large-nodes .bool_set:N = \l_@@_large_nodes_bool ,
563 create-extra-nodes .meta:n =
564 { create-medium-nodes , create-large-nodes } ,
565 left-margin .dim_set:N = \l_@@_left_margin_dim ,
566 left-margin .default:n = \arraycolsep ,
567 right-margin .dim_set:N = \l_@@_right_margin_dim ,
568 right-margin .default:n = \arraycolsep ,
569 margin .meta:n = { left-margin = #1 , right-margin = #1 } ,
570 margin .default:n = \arraycolsep ,
571 extra-left-margin .dim_set:N = \l_@@_extra_left_margin_dim ,
572 extra-right-margin .dim_set:N = \l_@@_extra_right_margin_dim ,
573 extra-margin .meta:n =
574 { extra-left-margin = #1 , extra-right-margin = #1 } ,

```

```

575     extra-margin .value_required:n = true ,
576 }

```

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

```

577 \keys_define:nn { NiceMatrix / Env }
578 {
579     delimiters/max-width .bool_set:N = \l_@@_delimiters_max_width_bool ,

```

The key `hvlines-except-corners` is now deprecated.

```

580     hvlines-except-corners .code:n =
581     {
582         \clist_set:Nn \l_@@_corners_clist { #1 }
583         \clist_set:Nn \l_@@_vlines_clist { all }
584         \clist_set:Nn \l_@@_hlines_clist { all }
585     } ,
586     hvlines-except-corners .default:n = { NW , SW , NE , SE } ,
587     corners .clist_set:N = \l_@@_corners_clist ,
588     corners .default:n = { NW , SW , NE , SE } ,
589     code-before .code:n =
590     {
591         \tl_if_empty:nF { #1 }
592         {
593             \tl_put_right:Nn \l_@@_code_before_tl { #1 }
594             \bool_set_true:N \l_@@_code_before_bool
595         }
596     } ,

```

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

```

597     c .code:n = \tl_set:Nn \l_@@_baseline_tl c ,
598     t .code:n = \tl_set:Nn \l_@@_baseline_tl t ,
599     b .code:n = \tl_set:Nn \l_@@_baseline_tl b ,
600     baseline .tl_set:N = \l_@@_baseline_tl ,
601     baseline .value_required:n = true ,
602     columns-width .code:n =
603     { \tl_if_eq:nnTF { #1 } { auto }
604       { \bool_set_true:N \l_@@_auto_columns_width_bool }
605       { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,
606     columns-width .value_required:n = true ,
607     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.

```

608     \legacy_if:nF { measuring@ }
609     {
610         \str_set:Nn \l_tmpa_str { #1 }
611         \seq_if_in:NVTF \g_@@_names_seq \l_tmpa_str
612         { \@@_error:nn { Duplicate~name } { #1 } }
613         { \seq_gput_left:NV \g_@@_names_seq \l_tmpa_str }
614         \str_set_eq:NN \l_@@_name_str \l_tmpa_str
615     } ,
616     name .value_required:n = true ,
617     code-after .tl_gset:N = \g_nicematrix_code_after_tl ,
618     code-after .value_required:n = true ,
619     colortbl-like .code:n =
620     { \bool_set_true:N \l_@@_colortbl_like_bool
621       { \bool_set_true:N \l_@@_code_before_bool } ,
622     colortbl-like .value_forbidden:n = true
623 }

```

```

624 \keys_define:nn { NiceMatrix / notes }
625 {
626   para .bool_set:N = \l_@@_notes_para_bool ,
627   para .default:n = true ,
628   code-before .tl_set:N = \l_@@_notes_code_before_tl ,
629   code-before .value_required:n = true ,
630   code-after .tl_set:N = \l_@@_notes_code_after_tl ,
631   code-after .value_required:n = true ,
632   bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
633   bottomrule .default:n = true ,
634   style .code:n = \cs_set:Nn \@@_notes_style:n { #1 } ,
635   style .value_required:n = true ,
636   label-in-tabular .code:n =
637     \cs_set:Nn \@@_notes_label_in_tabular:n { #1 } ,
638   label-in-tabular .value_required:n = true ,
639   label-in-list .code:n =
640     \cs_set:Nn \@@_notes_label_in_list:n { #1 } ,
641   label-in-list .value_required:n = true ,
642   enumitem-keys .code:n =
643     {
644       \bool_if:NTF \c_@@_in_preamble_bool
645       {
646         \AtBeginDocument
647         {
648           \bool_if:NT \c_@@_enumitem_loaded_bool
649             { \setlist* [ tabularnotes ] { #1 } }
650         }
651       }
652       {
653         \bool_if:NT \c_@@_enumitem_loaded_bool
654           { \setlist* [ tabularnotes ] { #1 } }
655       }
656     } ,
657   enumitem-keys .value_required:n = true ,
658   enumitem-keys-para .code:n =
659     {
660       \bool_if:NTF \c_@@_in_preamble_bool
661       {
662         \AtBeginDocument
663         {
664           \bool_if:NT \c_@@_enumitem_loaded_bool
665             { \setlist* [ tabularnotes* ] { #1 } }
666         }
667       }
668       {
669         \bool_if:NT \c_@@_enumitem_loaded_bool
670           { \setlist* [ tabularnotes* ] { #1 } }
671       }
672     } ,
673   enumitem-keys-para .value_required:n = true ,
674   unknown .code:n = \@@_error:n { Unknown~key~for~notes }
675 }

```

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

```

676 \keys_define:nn { NiceMatrix }
677 {
678   NiceMatrixOptions .inherit:n =
679     { NiceMatrix / Global } ,
680   NiceMatrixOptions / xdots .inherit:n = NiceMatrix / xdots ,
681   NiceMatrixOptions / rules .inherit:n = NiceMatrix / rules ,
682   NiceMatrixOptions / notes .inherit:n = NiceMatrix / notes ,
683   NiceMatrixOptions / sub-matrix .inherit:n = NiceMatrix / sub-matrix ,

```

```

684 SubMatrix / rules .inherit:n = NiceMatrix / rules ,
685 CodeAfter / xdots .inherit:n = NiceMatrix / xdots ,
686 NiceMatrix .inherit:n =
687 {
688     NiceMatrix / Global ,
689     NiceMatrix / Env ,
690 } ,
691 NiceMatrix / xdots .inherit:n = NiceMatrix / xdots ,
692 NiceMatrix / rules .inherit:n = NiceMatrix / rules ,
693 NiceTabular .inherit:n =
694 {
695     NiceMatrix / Global ,
696     NiceMatrix / Env
697 } ,
698 NiceTabular / xdots .inherit:n = NiceMatrix / xdots ,
699 NiceTabular / rules .inherit:n = NiceMatrix / rules ,
700 NiceArray .inherit:n =
701 {
702     NiceMatrix / Global ,
703     NiceMatrix / Env ,
704 } ,
705 NiceArray / xdots .inherit:n = NiceMatrix / xdots ,
706 NiceArray / rules .inherit:n = NiceMatrix / rules ,
707 pNiceArray .inherit:n =
708 {
709     NiceMatrix / Global ,
710     NiceMatrix / Env ,
711 } ,
712 pNiceArray / xdots .inherit:n = NiceMatrix / xdots ,
713 pNiceArray / rules .inherit:n = NiceMatrix / rules ,
714 }

```

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

```

715 \keys_define:nn { NiceMatrix / NiceMatrixOptions }
716 {
717     delimiters / max-width .bool_set:N = \l_@@_delimiters_max_width_bool ,
718     delimiters / color .tl_set:N = \l_@@_delimiters_color_tl ,
719     delimiters / color .value_required:n = true ,
720     delimiters-color .tl_set:N = \l_@@_delimiters_color_tl ,
721     delimiters-color .value_required:n = true ,
722     last-col .code:n = \tl_if_empty:nF { #1 }
723         { \@@_error:n { last-col~non-empty~for~NiceMatrixOptions } }
724         \int_zero:N \l_@@_last_col_int ,
725     small .bool_set:N = \l_@@_small_bool ,
726     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.

```

727     renew-matrix .code:n = \@@_renew_matrix: ,
728     renew-matrix .value_forbidden:n = true ,

```

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

```

729     transparent .code:n =
730     {
731         \@@_renew_matrix:
732         \bool_set_true:N \l_@@_renew_dots_bool
733         \@@_error:n { Key~transparent }
734     } ,
735     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}`.

```

736     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.

```

737   columns-width .code:n =
738     \tl_if_eq:nnTF { #1 } { auto }
739     { \@@_error:n { Option~auto~for~columns-width } }
740     { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,

```

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

```

741   allow-duplicate-names .code:n =
742     \@@_msg_redirect_name:nn { Duplicate-name } { none } ,
743   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`).

```

744   letter-for-dotted-lines .code:n =
745     {
746       \tl_if_single_token:nTF { #1 }
747       { \str_set:Nx \l_@@_letter_for_dotted_lines_str { #1 } }
748       { \@@_error:n { One~letter~allowed } }
749     } ,
750   letter-for-dotted-lines .value_required:n = true ,
751   notes .code:n = \keys_set:nn { NiceMatrix / notes } { #1 } ,
752   notes .value_required:n = true ,
753   sub-matrix .code:n =
754     \keys_set:nn { NiceMatrix / sub-matrix } { #1 } ,
755   sub-matrix .value_required:n = true ,
756   unknown .code:n = \@@_error:n { Unknown~key~for~NiceMatrixOptions }
757 }

758 \str_new:N \l_@@_letter_for_dotted_lines_str
759 \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.

```

760 \NewDocumentCommand \NiceMatrixOptions { m }
761 { \keys_set:nn { NiceMatrix / NiceMatrixOptions } { #1 } }

```

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

```

762 \keys_define:nn { NiceMatrix / NiceMatrix }
763 {
764   last-col .code:n = \tl_if_empty:nTF {#1}
765     {
766       \bool_set_true:N \l_@@_last_col_without_value_bool
767       \int_set:Nn \l_@@_last_col_int { -1 }
768     }
769     { \int_set:Nn \l_@@_last_col_int { #1 } } ,
770   l .code:n = \tl_set:Nn \l_@@_type_of_col_tl l ,
771   r .code:n = \tl_set:Nn \l_@@_type_of_col_tl r ,
772   small .bool_set:N = \l_@@_small_bool ,
773   small .value_forbidden:n = true ,
774   delimiters / color .tl_set:N = \l_@@_delimiters_color_tl ,
775   delimiters / color .value_required:n = true ,
776   delimiters-color .tl_set:N = \l_@@_delimiters_color_tl ,
777   delimiters-color .value_required:n = true ,
778   unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix }
779 }

```

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

```
780 \keys_define:nn { NiceMatrix / NiceArray }
781 {
```

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.

```
782   small .bool_set:N = \l_@@_small_bool ,
783   small .value_forbidden:n = true ,
784   last-col .code:n = \tl_if_empty:nF { #1 }
785     { \@@_error:n { last-col~non-empty~for~NiceArray } }
786     \int_zero:N \l_@@_last_col_int ,
787   notes / para .bool_set:N = \l_@@_notes_para_bool ,
788   notes / para .default:n = true ,
789   notes / bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
790   notes / bottomrule .default:n = true ,
791   tabularnote .tl_set:N = \l_@@_tabularnote_tl ,
792   tabularnote .value_required:n = true ,
793   delimiters-color .tl_set:N = \l_@@_delimiters_color_tl ,
794   delimiters-color .value_required:n = true ,
795   delimiters / color .tl_set:N = \l_@@_delimiters_color_tl ,
796   delimiters / color .value_required:n = true ,
797   r .code:n = \@@_error:n { r~or~l~with~preamble } ,
798   l .code:n = \@@_error:n { r~or~l~with~preamble } ,
799   unknown .code:n = \@@_error:n { Unknown~option~for~NiceArray }
800 }

801 \keys_define:nn { NiceMatrix / pNiceArray }
802 {
803   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
804   last-col .code:n = \tl_if_empty:nF {#1}
805     { \@@_error:n { last-col~non-empty~for~NiceArray } }
806     \int_zero:N \l_@@_last_col_int ,
807   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
808   small .bool_set:N = \l_@@_small_bool ,
809   small .value_forbidden:n = true ,
810   r .code:n = \@@_error:n { r~or~l~with~preamble } ,
811   l .code:n = \@@_error:n { r~or~l~with~preamble } ,
812   unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix }
813 }
```

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

```
814 \keys_define:nn { NiceMatrix / NiceTabular }
815 {
816   notes / para .bool_set:N = \l_@@_notes_para_bool ,
817   notes / para .default:n = true ,
818   notes / bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
819   notes / bottomrule .default:n = true ,
820   tabularnote .tl_set:N = \l_@@_tabularnote_tl ,
821   tabularnote .value_required:n = true ,
822   last-col .code:n = \tl_if_empty:nF {#1}
823     { \@@_error:n { last-col~non-empty~for~NiceArray } }
824     \int_zero:N \l_@@_last_col_int ,
825   r .code:n = \@@_error:n { r~or~l~with~preamble } ,
826   l .code:n = \@@_error:n { r~or~l~with~preamble } ,
827   unknown .code:n = \@@_error:n { Unknown~option~for~NiceTabular }
828 }
```



## Important code used by `{NiceArrayWithDelims}`

The pseudo-environment `\@@_Cell:–\@@_end_Cell:` will be used to format the cells of the array. In the code, the affectations are global because this pseudo-environment will be used in the cells of a `\halign` (via an environment `{array}`).

```
829 \cs_new_protected:Npn \@@_Cell:
830 {
```

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`).

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

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

```
832 \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.

```
833 \int_compare:nNnT \c@jCol = 1
834 { \int_compare:nNnT \l_@@_first_col_int = 1 \@@_begin_of_row: }
```

The content of the cell is composed in the box `\l_@@_cell_box` because we want to compute some dimensions of the box. The `\hbox_set_end:` corresponding to this `\hbox_set:Nw` will be in the `\@@_end_Cell:` (and the potential `\c_math_toggle_token` also).

```
835 \hbox_set:Nw \l_@@_cell_box
836 \bool_if:NF \l_@@_NiceTabular_bool
837 {
838   \c_math_toggle_token
839   \bool_if:NT \l_@@_small_bool \scriptstyle
840 }
```

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.

```
841 \int_compare:nNnTF \c@iRow = 0
842 {
843   \int_compare:nNnT \c@jCol > 0
844   {
845     \l_@@_code_for_first_row_tl
846     \xglobal \colorlet { nicematrix-first-row } { . }
847   }
848 }
849 {
850   \int_compare:nNnT \c@iRow = \l_@@_last_row_int
851   {
852     \l_@@_code_for_last_row_tl
853     \xglobal \colorlet { nicematrix-last-row } { . }
854   }
855 }
856 }
```

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.

```
857 \cs_new_protected:Npn \@@_begin_of_row:
858 {
859   \int_gincr:N \c@iRow
860   \dim_gset_eq:NN \g_@@_dp_ante_last_row_dim \g_@@_dp_last_row_dim
861   \dim_gset:Nn \g_@@_dp_last_row_dim { \box_dp:N \@arstrutbox }
862   \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
863   \pgfpicture
864   \pgfrememberpicturepositiononpagetrue
865   \pgfcoordinate
866   { \@@_env: - row - \int_use:N \c@iRow - base }
```

```

867     { \pgfpoint \c_zero_dim { 0.5 \arrayrulewidth } }
868     \str_if_empty:NF \l_@@_name_str
869     {
870         \pgfnodealias
871         { \l_@@_name_str - row - \int_use:N \c@iRow - base }
872         { \@@_env: - row - \int_use:N \c@iRow - base }
873     }
874     \endpgfpicture
875 }

```

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.

```

876 \cs_new_protected:Npn \@@_update_for_first_and_last_row:
877 {
878     \int_compare:nNnTF \c@iRow = 0
879     {
880         \dim_gset:Nn \g_@@_dp_row_zero_dim
881         { \dim_max:nn \g_@@_dp_row_zero_dim { \box_dp:N \l_@@_cell_box } }
882         \dim_gset:Nn \g_@@_ht_row_zero_dim
883         { \dim_max:nn \g_@@_ht_row_zero_dim { \box_ht:N \l_@@_cell_box } }
884     }
885     {
886         \int_compare:nNnT \c@iRow = 1
887         {
888             \dim_gset:Nn \g_@@_ht_row_one_dim
889             { \dim_max:nn \g_@@_ht_row_one_dim { \box_ht:N \l_@@_cell_box } }
890         }
891     }
892 }
893 \cs_new_protected:Npn \@@_rotate_cell_box:
894 {
895     \box_rotate:Nn \l_@@_cell_box { 90 }
896     \int_compare:nNnT \c@iRow = \l_@@_last_row_int
897     {
898         \vbox_set_top:Nn \l_@@_cell_box
899         {
900             \vbox_to_zero:n { }
901             \skip_vertical:n { - \box_ht:N \@arstrutbox + 0.8 ex }
902             \box_use:N \l_@@_cell_box
903         }
904     }
905     \bool_gset_false:N \g_@@_rotate_bool
906 }
907 \cs_new_protected:Npn \@@_adjust_size_box:
908 {
909     \dim_compare:nNnT \g_@@_blocks_wd_dim > \c_zero_dim
910     {
911         \box_set_wd:Nn \l_@@_cell_box
912         { \dim_max:nn { \box_wd:N \l_@@_cell_box } \g_@@_blocks_wd_dim }
913         \dim_gzero:N \g_@@_blocks_wd_dim
914     }
915     \dim_compare:nNnT \g_@@_blocks_dp_dim > \c_zero_dim
916     {
917         \box_set_dp:Nn \l_@@_cell_box
918         { \dim_max:nn { \box_dp:N \l_@@_cell_box } \g_@@_blocks_dp_dim }
919         \dim_gzero:N \g_@@_blocks_dp_dim
920     }
921     \dim_compare:nNnT \g_@@_blocks_ht_dim > \c_zero_dim
922     {
923         \box_set_ht:Nn \l_@@_cell_box

```

```

924         { \dim_max:nn { \box_ht:N \l_@@_cell_box } \g_@@_blocks_ht_dim }
925         \dim_gzero:N \g_@@_blocks_ht_dim
926     }
927 }

928 \cs_new_protected:Npn \@@_end_Cell:
929 {
930     \@@_math_toggle_token:
931     \hbox_set_end:
932     \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
933     \@@_adjust_size_box:

934     \box_set_ht:Nn \l_@@_cell_box
935         { \box_ht:N \l_@@_cell_box + \l_@@_cell_space_top_limit_dim }
936     \box_set_dp:Nn \l_@@_cell_box
937         { \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”).

```

938     \dim_gset:Nn \g_@@_max_cell_width_dim
939     { \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”.

```

940     \@@_update_for_first_and_last_row:

```

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

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

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

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

```

941     \bool_if:NTF \g_@@_empty_cell_bool
942     { \box_use_drop:N \l_@@_cell_box }
943     {
944         \bool_lazy_or:nnTF
945             \g_@@_not_empty_cell_bool
946             { \dim_compare_p:nNn { \box_wd:N \l_@@_cell_box } > \c_zero_dim }
947             \@@_node_for_the_cell:
948             { \box_use_drop:N \l_@@_cell_box }
949     }
950     \int_gset:Nn \g_@@_col_total_int { \int_max:nn \g_@@_col_total_int \c_jCol }
951     \bool_gset_false:N \g_@@_empty_cell_bool
952     \bool_gset_false:N \g_@@_not_empty_cell_bool
953 }

```

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

```

954 \cs_new_protected:Npn \@@_node_for_the_cell:
955 {
956     \pgfpicture
957     \pgfsetbaseline \c_zero_dim
958     \pgfrememberpicturepositiononpagetrue

```

```

959 \pgfset
960 {
961     inner~sep = \c_zero_dim ,
962     minimum~width = \c_zero_dim
963 }
964 \pgfnode
965 { rectangle }
966 { base }
967 { \box_use_drop:N \l_@@_cell_box }
968 { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
969 { }
970 \str_if_empty:NF \l_@@_name_str
971 {
972     \pgfnodealias
973     { \l_@@_name_str - \int_use:N \c@iRow - \int_use:N \c@jCol }
974     { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
975 }
976 \endpgfpicture
977 }

```

The second argument of the following command `\@@_instruction_of_type:nnn` defined below is the type of the instruction (`Cdots`, `Vdots`, `Ddots`, etc.). The third argument is the list of options. This command writes in the corresponding `\g_@@_type_lines_tl` the instruction which will actually draw the line after the construction of the matrix.

For example, for the following matrix,

```

\begin{pNiceMatrix}
1 & 2 & 3 & 4 \\
5 & \Cdots & & 6 \\
7 & \Cdots & [color=red]
\end{pNiceMatrix}

```

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

the content of `\g_@@_Cdots_lines_tl` will be:

```

\@@_draw_Cdots:nnn {2}{2}{}
\@@_draw_Cdots:nnn {3}{2}{color=red}

```

The first argument is a boolean which indicates whether you must put the instruction on the left or on the right on the list of instructions.

```

978 \cs_new_protected:Npn \@@_instruction_of_type:nnn #1 #2 #3
979 {
980     \bool_if:nTF { #1 } \tl_gput_left:cx \tl_gput_right:cx
981     { g_@@_ #2 _ lines _ tl }
982     {
983         \use:c { @@ _ draw _ #2 : nnn }
984         { \int_use:N \c@iRow }
985         { \int_use:N \c@jCol }
986         { \exp_not:n { #3 } }
987     }
988 }

```

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`.

```

989 \cs_new_protected:Npn \@@_revtex_array:
990 {
991     \cs_set_eq:NN \@acoll \@arrayacol
992     \cs_set_eq:NN \@acolr \@arrayacol
993     \cs_set_eq:NN \@acol \@arrayacol
994     \cs_set_nopar:Npn \@halignto { }
995     \@array@array
996 }

```

```

997 \cs_new_protected:Npn \l_@@_array:
998 {
999   \bool_if:NTF \c_@@_revtex_bool
1000     \l_@@_revtex_array:
1001   {
1002     \bool_if:NTF \l_@@_NiceTabular_bool
1003       { \dim_set_eq:NN \col@sep \tabcolsep }
1004       { \dim_set_eq:NN \col@sep \arraycolsep }
1005     \dim_compare:nNnTF \l_@@_tabular_width_dim = \c_zero_dim
1006       { \cs_set_nopar:Npn \@halignto { } }
1007       { \cs_set_nopar:Npx \@halignto { to \dim_use:N \l_@@_tabular_width_dim } }

```

It colortbl is loaded, \@tabarray has been redefined to incorporate \CTstart.

```

1008   \@tabarray
1009 }

```

\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.

```

1010   [ \str_if_eq:VnTF \l_@@_baseline_tl c c t ]
1011 }

```

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.

```

1012 \cs_set_eq:NN \l_@@_old_ialign: \ialign

```

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

```

1013 \cs_new_protected:Npn \l_@@_create_row_node:
1014 {

```

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

```

1015   \hbox
1016   {
1017     \bool_if:NT \l_@@_code_before_bool
1018     {
1019       \vtop
1020       {
1021         \skip_vertical:N 0.5\arrayrulewidth
1022         \pgfsys@markposition { \l_@@_env: - row - \l_@@_succ:n \c@iRow }
1023         \skip_vertical:N -0.5\arrayrulewidth
1024       }
1025     }
1026     \pgfpicture
1027     \pgfrememberpicturepositiononpagetrue
1028     \pgfcoordinate { \l_@@_env: - row - \l_@@_succ:n \c@iRow }
1029     { \pgfpoint \c_zero_dim { - 0.5 \arrayrulewidth } }
1030     \str_if_empty:NF \l_@@_name_str
1031     {
1032       \pgfnodealias
1033       { \l_@@_name_str - row - \int_use:N \c@iRow }
1034       { \l_@@_env: - row - \int_use:N \c@iRow }
1035     }
1036     \endpgfpicture
1037   }
1038 }

```

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

```

1039 \cs_new:Npn \l_@@_everycr: { \noalign { \l_@@_everycr_i: } }
1040 \cs_new_protected:Npn \l_@@_everycr_i:
1041 {
1042   \int_gzero:N \c@jCol
1043   \bool_gset_false:N \g_@@_after_col_zero_bool
1044   \bool_if:NF \g_@@_row_of_col_done_bool
1045   {
1046     \l_@@_create_row_node:

```

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

```

1047     \tl_if_empty:NF \l_@@_hlines_clist
1048     {
1049         \tl_if_eq:NnF \l_@@_hlines_clist { all }
1050         {
1051             \exp_args:NNx
1052             \clist_if_in:NnT
1053             \l_@@_hlines_clist
1054             { \@@_succ:n \c@iRow }
1055         }
1056     }

```

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.

```

1057         \int_compare:nNnT \c@iRow > { -1 }
1058         {
1059             \int_compare:nNnF \c@iRow = \l_@@_last_row_int

```

The command `\CT@arc@` is a command of `colortbl` which sets the color of the rules in the array. The package `nicematrix` uses it even if `colortbl` is not loaded. We use a TeX group in order to limit the scope of `\CT@arc@`.

```

1060             { \hrule height \arrayrulewidth width \c_zero_dim }
1061         }
1062     }
1063 }
1064 }
1065 }

```

The command `\@@_newcolumntype` is the command `\newcolumntype` of `array` without the warnings for redefinitions of columns types (we will use it to redefine the columns types `w` and `W`).

```

1066 \cs_set_protected:Npn \@@_newcolumntype #1
1067 {
1068     \cs_set:cpn { NC @ find @ #1 } ##1 #1 { \NC@ { ##1 } }
1069     \peek_meaning:NTF [
1070         { \newcol@ #1 }
1071         { \newcol@ #1 [ 0 ] }
1072     }

```

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

```

1073 \cs_set_protected:Npn \@@_renew_dots:
1074 {
1075     \cs_set_eq:NN \ldots \@@_Ldots
1076     \cs_set_eq:NN \cdots \@@_Cdots
1077     \cs_set_eq:NN \vdots \@@_Vdots
1078     \cs_set_eq:NN \ddots \@@_Ddots
1079     \cs_set_eq:NN \iddots \@@_Iddots
1080     \cs_set_eq:NN \dots \@@_Ldots
1081     \cs_set_eq:NN \hdotsfor \@@_Hdotsfor:
1082 }

```

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

```

1083 \cs_new_protected:Npn \@@_colortbl_like:
1084 {
1085     \cs_set_eq:NN \cellcolor \@@_cellcolor_tabular
1086     \cs_set_eq:NN \rowcolor \@@_rowcolor_tabular
1087     \cs_set_eq:NN \columncolor \@@_columncolor_preamble
1088 }

```

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

```
1089 \cs_new_protected:Npn \@@_pre_array_ii:
1090 {
```

If `booktabs` is loaded, we have to patch the macro `\@BTnormal` which is a macro of `booktabs`. The macro `\@BTnormal` draws an horizontal rule but it occurs after a vertical skip done by a low level TeX command. When this macro `\@BTnormal` occurs, the `row` node has yet been inserted by `nicematrix` *before* the vertical skip (and thus, at a wrong place). That why we decide to create a new `row` node (for the same row). We patch the macro `\@BTnormal` to create this `row` node. This new `row` node will overwrite the previous definition of that `row` node and we have managed to avoid the error messages of that redefinition <sup>51</sup>.

```
1091   \bool_if:NT \c_@@_booktabs_loaded_bool
1092   { \tl_put_left:Nn \@BTnormal \@_create_row_node: }
1093   \box_clear_new:N \l_@@_cell_box
1094   \cs_if_exist:NT \theiRow
1095   { \int_set_eq:NN \l_@@_old_iRow_int \c@iRow }
1096   \int_gzero_new:N \c@iRow
1097   \cs_if_exist:NT \thejCol
1098   { \int_set_eq:NN \l_@@_old_jCol_int \c@jCol }
1099   \int_gzero_new:N \c@jCol
1100   \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}`).

```
1101   \bool_if:NT \l_@@_small_bool
1102   {
1103     \cs_set_nopar:Npn \arraystretch { 0.47 }
1104     \dim_set:Nn \arraycolsep { 1.45 pt }
1105   }
```

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`.

```
1106   \cs_set_nopar:Npn \ialign
1107   {
1108     \bool_if:NTF \c_@@_colortbl_loaded_bool
1109     {
1110       \CT@everycr
1111       {
1112         \noalign { \cs_gset_eq:NN \CT@row@color \prg_do_nothing: }
1113         \@_everycr:
1114       }
1115     }
1116     { \everycr { \@_everycr: } }
1117     \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>52</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`.

```
1118     \dim_gzero_new:N \g_@@_dp_row_zero_dim
1119     \dim_gset:Nn \g_@@_dp_row_zero_dim { \box_dp:N \@arstrutbox }
1120     \dim_gzero_new:N \g_@@_ht_row_zero_dim
```

---

<sup>51</sup>cf. `\nicematrix@redefine@check@rerun`

<sup>52</sup>The option `small` of `nicematrix` changes (among other) the value of `\arraystretch`. This is done, of course, before the call of `{array}`.

```

1121 \dim_gset:Nn \g_@@_ht_row_zero_dim { \box_ht:N \@arstrutbox }
1122 \dim_gzero_new:N \g_@@_ht_row_one_dim
1123 \dim_gset:Nn \g_@@_ht_row_one_dim { \box_ht:N \@arstrutbox }
1124 \dim_gzero_new:N \g_@@_dp_ante_last_row_dim
1125 \dim_gzero_new:N \g_@@_ht_last_row_dim
1126 \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
1127 \dim_gzero_new:N \g_@@_dp_last_row_dim
1128 \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`.

```

1129 \cs_set_eq:NN \ialign \@@_old_ialign:
1130 \halign
1131 }

```

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).

```

1132 \cs_set_eq:NN \@@_old_ldots \ldots
1133 \cs_set_eq:NN \@@_old_cdots \cdots
1134 \cs_set_eq:NN \@@_old_vdots \vdots
1135 \cs_set_eq:NN \@@_old_ddots \ddots
1136 \cs_set_eq:NN \@@_old_iddots \iddots
1137 \bool_if:NTF \l_@@_standard_cline_bool
1138 { \cs_set_eq:NN \cline \@@_standard_cline }
1139 { \cs_set_eq:NN \cline \@@_cline }
1140 \cs_set_eq:NN \Ldots \@@_Ldots
1141 \cs_set_eq:NN \Cdots \@@_Cdots
1142 \cs_set_eq:NN \Vdots \@@_Vdots
1143 \cs_set_eq:NN \Ddots \@@_Ddots
1144 \cs_set_eq:NN \Iddots \@@_Iddots
1145 \cs_set_eq:NN \hdottedline \@@_hdottedline:
1146 \cs_set_eq:NN \Hline \@@_Hline:
1147 \cs_set_eq:NN \Hspace \@@_Hspace:
1148 \cs_set_eq:NN \Hdotsfor \@@_Hdotsfor:
1149 \cs_set_eq:NN \Vdotsfor \@@_Vdotsfor:
1150 \cs_set_eq:NN \multicolumn \@@_multicolumn:nnn
1151 \cs_set_eq:NN \Block \@@_Block:
1152 \cs_set_eq:NN \rotate \@@_rotate:
1153 \cs_set_eq:NN \OnlyMainNiceMatrix \@@_OnlyMainNiceMatrix:n
1154 \cs_set_eq:NN \dotfill \@@_old_dotfill:
1155 \cs_set_eq:NN \CodeAfter \@@_CodeAfter:
1156 \cs_set_eq:NN \diagbox \@@_diagbox:nn
1157 \cs_set_eq:NN \NotEmpty \@@_NotEmpty:
1158 \bool_if:NT \l_@@_colortbl_like_bool \@@_colortbl_like:
1159 \bool_if:NT \l_@@_renew_dots_bool \@@_renew_dots:

```

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

```

1160 \seq_gclear_new:N \g_@@_multicolumn_cells_seq
1161 \seq_gclear_new:N \g_@@_multicolumn_sizes_seq

```

The counter `\c@iRow` will be used to count the rows of the array (its incrementation will be in the first cell of the row).

```

1162 \int_gset:Nn \c@iRow { \l_@@_first_row_int - 1 }

```

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

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

```

1163 \int_gzero_new:N \g_@@_row_total_int

```



The counter `\c@jCol` will be used to count the columns of the array. Since we want to know the total number of columns of the matrix, we also create a counter `\g_@@_col_total_int`. These counters are updated in the command `\@@_Cell`: executed at the beginning of each cell.

```

1164 \int_gzero_new:N \g_@@_col_total_int
1165 \cs_set_eq:NN \@ifnextchar \new@ifnextchar
1166 \@@_renew_NC@rewrite@S:
1167 \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.

```

1168 \tl_gclear_new:N \g_@@_Cdots_lines_tl
1169 \tl_gclear_new:N \g_@@_Ldots_lines_tl
1170 \tl_gclear_new:N \g_@@_Vdots_lines_tl
1171 \tl_gclear_new:N \g_@@_Ddots_lines_tl
1172 \tl_gclear_new:N \g_@@_Iddots_lines_tl
1173 \tl_gclear_new:N \g_@@_HVDotsfor_lines_tl

1174 \tl_gclear_new:N \g_nicematrix_code_before_tl
1175 }

```

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

If the key `code-before` is used, we have to create the `col` nodes and the `row` nodes before the creation of the array. First, we have to test whether the size of the array has been written in the `aux` file in a previous run. In this case, a command `\@@_size_nb_of_env`: has been created.

```

1176 \cs_new_protected:Npn \@@_pre_array:
1177 {
1178   \seq_gclear:N \g_@@_submatrix_seq
1179   \bool_if:NT \l_@@_code_before_bool
1180   {
1181     \seq_if_exist:cT { @@_size_ \int_use:N \g_@@_env_int _ seq }
1182     {

```

First, we give values to the LaTeX counters `iRow` and `jCol`. We remind that, in the `code-before` (and in the `\CodeAfter`) they represent the numbers of rows and columns of the array (without the potential last row and last column).

```

1183 \int_zero_new:N \c@iRow
1184 \int_set:Nn \c@iRow
1185 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 2 }
1186 \int_zero_new:N \c@jCol
1187 \int_set:Nn \c@jCol
1188 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 4 }

```

We have to adjust the values of `\c@iRow` and `\c@jCol` to take into account the potential last row and last column. A value of `-2` for `\l_@@_last_row_int` means that there is no last row. Idem for the columns.

```

1189 \int_compare:nNnF \l_@@_last_row_int = { -2 }
1190 { \int_decr:N \c@iRow }
1191 \int_compare:nNnF \l_@@_last_col_int = { -2 }
1192 { \int_decr:N \c@jCol }

```

Now, we will create all the `col` nodes and `row` nodes with the informations written in the `aux` file. You use the technique described in the page 1229 of `pgfmanual.pdf`, version 3.1.4b.

```

1193 \pgfsys@markposition { \@@_env: - position }
1194 \pgfsys@getposition { \@@_env: - position } \@@_picture_position:
1195 \pgfpicture

```

First, the creation of the row nodes.

```

1196 \int_step_inline:nnn
1197 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 1 }
1198 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 2 + 1 }
1199 {

```

```

1200         \pgfsys@getposition { \@@_env: - row - ##1 } \@@_node_position:
1201         \pgfcoordinate { \@@_env: - row - ##1 }
1202         { \pgfpointdiff \@@_picture_position: \@@_node_position: }
1203     }

```

Now, the creation of the col nodes.

```

1204     \int_step_inline:nnn
1205     { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 3 }
1206     { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 4 + 1 }
1207     {
1208         \pgfsys@getposition { \@@_env: - col - ##1 } \@@_node_position:
1209         \pgfcoordinate { \@@_env: - col - ##1 }
1210         { \pgfpointdiff \@@_picture_position: \@@_node_position: }
1211     }
1212 \endpgfpicture

```

Now, you create the diagonal nodes by using the row nodes and the col nodes. If the engine is `xetex` or `luatex` we also create the “½ nodes”.

```

1213     \@@_create_diag_nodes:

```

Now, yet other settings before the execution of the `code-before`.

```

1214     \group_begin:
1215     \bool_if:NT \c_@@_tikz_loaded_bool
1216     {
1217         \tikzset
1218         {
1219             every-picture / .style =
1220             { overlay , name-prefix = \@@_env: - }
1221         }
1222     }
1223     \cs_set_eq:NN \cellcolor \@@_cellcolor
1224     \cs_set_eq:NN \rectanglecolor \@@_rectanglecolor
1225     \cs_set_eq:NN \roundedrectanglecolor \@@_roundedrectanglecolor
1226     \cs_set_eq:NN \rowcolor \@@_rowcolor
1227     \cs_set_eq:NN \rowcolors \@@_rowcolors
1228     \cs_set_eq:NN \arraycolor \@@_arraycolor
1229     \cs_set_eq:NN \columncolor \@@_columncolor
1230     \cs_set_eq:NN \chessboardcolors \@@_chessboardcolors
1231     \cs_set_eq:NN \SubMatrix \@@_SubMatrix_in_code_before

```

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

```

1232     \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
1233     \seq_gclear_new:N \g_@@_colors_seq

```

Here is the `\CodeBefore`. As of now, the keys that may be provided to the keyword `\CodeBefore` are the same as keys that may be provided to `\CodeAfter`, hence the `\@@_CodeAfter_keys:`.

```

1234     \exp_last_unbraced:NV \@@_CodeAfter_keys: \l_@@_code_before_tl

```

Now, all the cells which are specified to be colored by instructions in the `\CodeBefore` will actually be colored. It’s a two-stages mechanism because we want to draw all the cells with the same color at the same time to absolutely avoid thin white lines in some PDF viewers.

```

1235     \@@_actually_color:
1236     \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
1237     \group_end:
1238 }
1239 }

```

A value of `-1` for the counter `\l_@@_last_row_int` means that the user has used the option `last-row` without value, that is to say without specifying the number of that last row. In this case, we try to read that value from the aux file (if it has been written on a previous run).

```

1240     \int_compare:nNnT \l_@@_last_row_int > { -2 }
1241     {
1242         \tl_put_right:Nn \@@_update_for_first_and_last_row:
1243         {

```

```

1244         \dim_gset:Nn \g_@@_ht_last_row_dim
1245         { \dim_max:nn \g_@@_ht_last_row_dim { \box_ht:N \l_@@_cell_box } }
1246         \dim_gset:Nn \g_@@_dp_last_row_dim
1247         { \dim_max:nn \g_@@_dp_last_row_dim { \box_dp:N \l_@@_cell_box } }
1248     }
1249 }
1250 \int_compare:nNnT \l_@@_last_row_int = { -1 }
1251 {
1252     \bool_set_true:N \l_@@_last_row_without_value_bool

```

A value based on the name is more reliable than a value based on the number of the environment.

```

1253     \str_if_empty:NTF \l_@@_name_str
1254     {
1255         \cs_if_exist:cT { @@_last_row_ \int_use:N \g_@@_env_int }
1256         {
1257             \int_set:Nn \l_@@_last_row_int
1258             { \use:c { @@_last_row_ \int_use:N \g_@@_env_int } }
1259         }
1260     }
1261     {
1262         \cs_if_exist:cT { @@_last_row_ \l_@@_name_str }
1263         {
1264             \int_set:Nn \l_@@_last_row_int
1265             { \use:c { @@_last_row_ \l_@@_name_str } }
1266         }
1267     }
1268 }

```

A value of  $-1$  for the counter `\l_@@_last_col_int` means that the user has used the option `last-col` without value, that is to say without specifying the number of that last column. In this case, we try to read that value from the aux file (if it has been written on a previous run).

```

1269     \int_compare:nNnT \l_@@_last_col_int = { -1 }
1270     {
1271         \str_if_empty:NTF \l_@@_name_str
1272         {
1273             \cs_if_exist:cT { @@_last_col_ \int_use:N \g_@@_env_int }
1274             {
1275                 \int_set:Nn \l_@@_last_col_int
1276                 { \use:c { @@_last_col_ \int_use:N \g_@@_env_int } }
1277             }
1278         }
1279         {
1280             \cs_if_exist:cT { @@_last_col_ \l_@@_name_str }
1281             {
1282                 \int_set:Nn \l_@@_last_col_int
1283                 { \use:c { @@_last_col_ \l_@@_name_str } }
1284             }
1285         }
1286     }

```

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

```

1287     \@@_pre_array_ii:

```

We compute the width of both delimiters.

```

1288     \dim_zero_new:N \l_@@_left_delim_dim
1289     \dim_zero_new:N \l_@@_right_delim_dim
1290     \bool_if:NTF \l_@@_NiceArray_bool
1291     {
1292         \dim_gset:Nn \l_@@_left_delim_dim { 2 \arraycolsep }
1293         \dim_gset:Nn \l_@@_right_delim_dim { 2 \arraycolsep }
1294     }
1295     {

```

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

```

1296     \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 \l_@@_left_delim_tl $ }
1297     \dim_set:Nn \l_@@_left_delim_dim { \box_wd:N \l_tmpa_box }
1298     \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 \l_@@_right_delim_tl $ }
1299     \dim_set:Nn \l_@@_right_delim_dim { \box_wd:N \l_tmpa_box }
1300 }

```

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

```

1301     \box_clear_new:N \l_@@_the_array_box

```

If the user has loaded `nicematrix` with the option `define-L-C-R`, he will be able to use `L`, `C` and `R` instead of `l`, `c` and `r` in the preambles of the environments of `nicematrix` (it's a compatibility mode since `L`, `C` and `R` were mandatory before version 5.0).

```

1302     \bool_if:NT \c_@@_define_L_C_R_bool \@@_define_L_C_R:

```

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

```

1303     \@@_construct_preamble:

```

Now, the preamble is constructed in `\g_@@_preamble_tl`

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).

```

1304     \hbox_set:Nw \l_@@_the_array_box
1305     \skip_horizontal:N \l_@@_left_margin_dim
1306     \skip_horizontal:N \l_@@_extra_left_margin_dim
1307     \c_math_toggle_token
1308     \bool_if:NTF \l_@@_light_syntax_bool
1309     { \use:c { @@-light-syntax } }
1310     { \use:c { @@-normal-syntax } }
1311 }

```

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

```

1312 \cs_new_protected:Npn \@@_pre_array_i:w #1 \Body
1313 {
1314     \tl_put_right:Nn \l_@@_code_before_tl { #1 }
1315     \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...

```

1316     \@@_pre_array:
1317 }

```

## The environment `{NiceArrayWithDelims}`

```

1318 \NewDocumentEnvironment { NiceArrayWithDelims }
1319 { m m O { } m ! O { } t \CodeBefore }
1320 {
1321     \@@_provide_pgfsyspdfmark:
1322     \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.

```

1323     \bgroup

1324     \tl_set:Nn \l_@@_left_delim_tl { #1 }

```

```

1325 \tl_set:Nn \l_@@_right_delim_tl { #2 }
1326 \tl_gset:Nn \g_@@_preamble_tl { #4 }

1327 \int_gzero:N \g_@@_block_box_int
1328 \dim_zero:N \g_@@_width_last_col_dim
1329 \dim_zero:N \g_@@_width_first_col_dim
1330 \bool_gset_false:N \g_@@_row_of_col_done_bool
1331 \str_if_empty:NT \g_@@_name_env_str
1332 { \str_gset:Nn \g_@@_name_env_str { NiceArrayWithDelims } }
1333 \@@_adapt_S_column:
1334 \bool_if:NTF \l_@@_NiceTabular_bool
1335 \mode_leave_vertical:
1336 \@@_test_if_math_mode:
1337 \bool_if:NT \l_@@_in_env_bool { \@@_fatal:n { Yet-in-env } }
1338 \bool_set_true:N \l_@@_in_env_bool

```

The command `\CT@arc@` contains the instruction of color for the rules of the array<sup>53</sup>. This command is used by `\CT@arc@` but we use it also for compatibility with `colortbl`. But we want also to be able to use color for the rules of the array when `colortbl` is *not* loaded. That's why we do the following instruction which is in the patch of the beginning of arrays done by `colortbl`. Of course, we restore the value of `\CT@arc@` at the end of our environment.

```

1339 \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.

```

1340 \cs_if_exist:NT \tikz@library@external@loaded
1341 {
1342   \tikzexternaldisable
1343   \cs_if_exist:NT \ifstandalone
1344   { \tikzset { external / optimize = false } }
1345 }

```

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

```

1346 \int_gincr:N \g_@@_env_int
1347 \bool_if:NF \l_@@_block_auto_columns_width_bool
1348 { \dim_gzero_new:N \g_@@_max_cell_width_dim }

```

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

```

1349 \seq_gclear:N \g_@@_blocks_seq
1350 \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`.

```

1351 \seq_gclear:N \g_@@_pos_of_stroken_blocks_seq
1352 \seq_gclear:N \g_@@_pos_of_xdots_seq

1353 \tl_if_exist:cT { g_@@_code_before_ \int_use:N \g_@@_env_int _ tl }
1354 {
1355   \bool_set_true:N \l_@@_code_before_bool
1356   \exp_args:NNv \tl_put_right:Nn \l_@@_code_before_tl
1357   { g_@@_code_before_ \int_use:N \g_@@_env_int _ tl }
1358 }

```

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`.

```

1359 \bool_if:NTF \l_@@_NiceArray_bool
1360 { \keys_set:nn { NiceMatrix / NiceArray } }
1361 { \keys_set:nn { NiceMatrix / pNiceArray } }
1362 { #3 , #5 }

```

---

<sup>53</sup>e.g. `\color[rgb]{0.5,0.5,0}`

```

1363 \tl_if_empty:NF \l_@@_rules_color_tl
1364 { \exp_after:wN \@@_set_CT@arc@: \l_@@_rules_color_tl \q_stop }
1365 % \bigskip

```

The argument #6 is the last argument of {NiceArrayWithDelims}. With that argument of type “t \CodeBefore”, we test whether there is the keyword \CodeBefore at the beginning of the environment. If that keyword is present, we have now to extract all the content between that keyword \CodeBefore and the (other) keyword \Body. It’s the job that will do the command \@@\_pre\_array\_i:w. After that job, the command \@@\_pre\_array\_i:w will go on with \@@\_pre\_array:.

```

1366 \IfBooleanTF { #6 } \@@_pre_array_i:w \@@_pre_array:
1367 }
1368 {
1369 \bool_if:NTF \l_@@_light_syntax_bool
1370 { \use:c { end @@-light-syntax } }
1371 { \use:c { end @@-normal-syntax } }
1372 \c_math_toggle_token
1373 \skip_horizontal:N \l_@@_right_margin_dim
1374 \skip_horizontal:N \l_@@_extra_right_margin_dim
1375 \hbox_set_end:

```

End of the construction of the array (in the box \l\_@@\_the\_array\_box).

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

```

1376 \int_compare:nNnT \l_@@_last_row_int > { -2 }
1377 {
1378 \bool_if:NF \l_@@_last_row_without_value_bool
1379 {
1380 \int_compare:nNnF \l_@@_last_row_int = \c@iRow
1381 {
1382 \@@_error:n { Wrong~last~row }
1383 \int_gset_eq:NN \l_@@_last_row_int \c@iRow
1384 }
1385 }
1386 }

```

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>54</sup>

```

1387 \int_gset_eq:NN \c@jCol \g_@@_col_total_int
1388 \bool_if:nTF \g_@@_last_col_found_bool
1389 { \int_gdecr:N \c@jCol }
1390 {
1391 \int_compare:nNnT \l_@@_last_col_int > { -1 }
1392 { \@@_error:n { last~col~not~used } }
1393 }

```

We fix also the value of \c@iRow and \g\_@@\_row\_total\_int with the same principle.

```

1394 \int_gset_eq:NN \g_@@_row_total_int \c@iRow
1395 \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. 103).

```

1396 \int_compare:nNnT \l_@@_first_col_int = 0
1397 {
1398 \skip_horizontal:N \col@sep
1399 \skip_horizontal:N \g_@@_width_first_col_dim
1400 }

```

---

<sup>54</sup>We remind that the potential “first column” (exterior) has the number 0.

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

Remark that, in all cases, `@@_use_arraybox_with_notes_c:` is used.

```

1401 \bool_if:NTF \l_@@_NiceArray_bool
1402 {
1403   \str_case:VnF \l_@@_baseline_tl
1404   {
1405     b \@@_use_arraybox_with_notes_b:
1406     c \@@_use_arraybox_with_notes_c:
1407   }
1408   \@@_use_arraybox_with_notes:
1409 }

```

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).

```

1410 {
1411   \int_compare:nNnTF \l_@@_first_row_int = 0
1412   {
1413     \dim_set_eq:NN \l_tmpa_dim \g_@@_dp_row_zero_dim
1414     \dim_add:Nn \l_tmpa_dim \g_@@_ht_row_zero_dim
1415   }
1416   { \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>55</sup>

```

1417   \int_compare:nNnTF \l_@@_last_row_int > { -2 }
1418   {
1419     \dim_set_eq:NN \l_tmpb_dim \g_@@_ht_last_row_dim
1420     \dim_add:Nn \l_tmpb_dim \g_@@_dp_last_row_dim
1421   }
1422   { \dim_zero:N \l_tmpb_dim }
1423   \hbox_set:Nn \l_tmpa_box
1424   {
1425     \c_math_toggle_token
1426     \tl_if_empty:NF \l_@@_delimiters_color_tl
1427     { \color { \l_@@_delimiters_color_tl } }
1428     \exp_after:wN \left \l_@@_left_delim_tl
1429     \vcenter
1430     {

```

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.

```

1431       \skip_vertical:N -\l_tmpa_dim
1432       \hbox
1433       {
1434         \bool_if:NTF \l_@@_NiceTabular_bool
1435         { \skip_horizontal:N -\tabcolsep }
1436         { \skip_horizontal:N -\arraycolsep }
1437         \@@_use_arraybox_with_notes_c:
1438         \bool_if:NTF \l_@@_NiceTabular_bool
1439         { \skip_horizontal:N -\tabcolsep }
1440         { \skip_horizontal:N -\arraycolsep }
1441       }

```

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

```

1442       \skip_vertical:N -\l_tmpb_dim
1443     }

```

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.

```

1444       \tl_if_empty:NF \l_@@_delimiters_color_tl

```

---

<sup>55</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).

```

1445         { \color { \l_@@_delimiters_color_tl } }
1446         \exp_after:wN \right \l_@@_right_delim_tl
1447         \c_math_toggle_token
1448     }

```

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.

```

1449     \bool_if:NTF \l_@@_delimiters_max_width_bool
1450     { \@@_put_box_in_flow_bis:nn { #1 } { #2 } }
1451     \@@_put_box_in_flow:
1452 }

```

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. 104).

```

1453     \bool_if:NT \g_@@_last_col_found_bool
1454     {
1455         \skip_horizontal:N \g_@@_width_last_col_dim
1456         \skip_horizontal:N \col@sep
1457     }
1458     \bool_if:NF \l_@@_Matrix_bool
1459     {
1460         \int_compare:nNnT \c@jCol < \g_@@_static_num_of_col_int
1461         { \@@_error:n { columns-not-used } }
1462     }
1463     \group_begin:
1464     \globaldefs = 1
1465     \@@_msg_redirect_name:nn { columns-not-used } { error }
1466     \group_end:
1467     \@@_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.

```

1468     \egroup
1469     \bool_if:NT \c_@@_footnote_bool \endsavenotes
1470 }

```

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 use is in `\g_@@_preamble_tl` and the modified version will be stored in `\g_@@_preamble_tl` also.

```

1471 \cs_new_protected:Npn \@@_construct_preamble:
1472 {

```

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 `expl3`.

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

```

1473     \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).

```

1474 \bool_if:NF \l_@@_Matrix_bool
1475 {
1476   \@@_newcolumntype w [ 2 ] { \@@_w: { ##1 } { ##2 } }
1477   \@@_newcolumntype W [ 2 ] { \@@_W: { ##1 } { ##2 } }

```

First, we have to store our preamble in the token register `\@temptokena` (those “token registers” are *not* supported by `expl3`).

```

1478 \exp_args:NV \@temptokena \g_@@_preamble_tl

```

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

```

1479 \@tempswatrue

```

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

```

1480 \@whilesw \if@tempswa \fi { \@tempswafalse \the \NC@list }

```

Now, we have to “patch” that preamble by transforming some columns. We will insert in the TeX flow the preamble in its actual form (that is to say after the “expansion”) following by a marker `\q_stop` and we will consume these tokens constructing the (new form of the) preamble in `\g_@@_preamble_tl`. This is done recursively with the command `\@@_patch_preamble:n`. In the same time, we will count the columns with the counter `\c@jCol`.

```

1481 \int_gzero_new:N \c@jCol
1482 \tl_gclear:N \g_@@_preamble_tl
1483 \tl_if_eq:NnTF \l_@@_vlines_clist { all }
1484 {
1485   \tl_gset:Nn \g_@@_preamble_tl
1486   { ! { \skip_horizontal:N \arrayrulewidth } }
1487 }
1488 {
1489   \clist_if_in:NnT \l_@@_vlines_clist 1
1490   {
1491     \tl_gset:Nn \g_@@_preamble_tl
1492     { ! { \skip_horizontal:N \arrayrulewidth } }
1493   }
1494 }

```

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`).

```

1495 \seq_clear:N \g_@@_cols_vlsim_seq

```

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

```

1496 \int_zero:N \l_tmpa_int

```

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

```

1497 \exp_after:wN \@@_patch_preamble:n \the \@temptokena \q_stop
1498 \int_gset_eq:NN \g_@@_static_num_of_col_int \c@jCol
1499 }

```

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

```

1500 \bool_if:NT \l_@@_colortbl_like_bool
1501 {
1502   \regex_replace_all:NnN
1503   \c_@@_columncolor_regex
1504   { \c { @@_columncolor_preamble } }
1505   \g_@@_preamble_tl
1506 }

```

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

```

1507 \int_compare:nNnTF \l_@@_first_col_int = 0
1508 { \tl_gput_left:NV \g_@@_preamble_tl \c_@@_preamble_first_col_tl }
1509 {
1510   \bool_lazy_all:nT
1511   {

```

```

1512         \l_@@_NiceArray_bool
1513         { \bool_not_p:n \l_@@_NiceTabular_bool }
1514         { \tl_if_empty_p:N \l_@@_vlines_clist }
1515         { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
1516     }
1517     { \tl_gput_left:Nn \g_@@_preamble_tl { @ { } } }
1518 }
1519 \int_compare:nNnTF \l_@@_last_col_int > { -1 }
1520 { \tl_gput_right:NV \g_@@_preamble_tl \c_@@_preamble_last_col_tl }
1521 {
1522     \bool_lazy_all:nT
1523     {
1524         \l_@@_NiceArray_bool
1525         { \bool_not_p:n \l_@@_NiceTabular_bool }
1526         { \tl_if_empty_p:N \l_@@_vlines_clist }
1527         { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
1528     }
1529     { \tl_gput_right:Nn \g_@@_preamble_tl { @ { } } }
1530 }

```

We add a last column to raise a good error message when the user put more columns than allowed by its preamble. However, for technical reasons, it's not possible to do that in `{NiceTabular*}` (`\l_@@_tabular_width_dim=0pt`).

```

1531     \dim_compare:nNnT \l_@@_tabular_width_dim = \c_zero_dim
1532     {
1533         \tl_gput_right:Nn \g_@@_preamble_tl
1534         { > { \@@_error_too_much_cols: } 1 }
1535     }

```

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

```

1536     \group_end:
1537 }

```

```

1538 \cs_new_protected:Npn \@@_patch_preamble:n #1
1539 {
1540     \str_case:nnF { #1 }
1541     {
1542         c { \@@_patch_preamble_i:n #1 }
1543         l { \@@_patch_preamble_i:n #1 }
1544         r { \@@_patch_preamble_i:n #1 }
1545         > { \@@_patch_preamble_ii:nn #1 }
1546         ! { \@@_patch_preamble_ii:nn #1 }
1547         @ { \@@_patch_preamble_ii:nn #1 }
1548         | { \@@_patch_preamble_iii:n #1 }
1549         p { \@@_patch_preamble_iv:nnn t #1 }
1550         m { \@@_patch_preamble_iv:nnn c #1 }
1551         b { \@@_patch_preamble_iv:nnn b #1 }
1552         \@@_w: { \@@_patch_preamble_v:nnnn { } #1 }
1553         \@@_W: { \@@_patch_preamble_v:nnnn { \cs_set_eq:NN \hss \hfil } #1 }
1554         \@@_true_c: { \@@_patch_preamble_vi:n #1 }
1555         ( { \@@_patch_preamble_vii:n #1 }
1556         [ { \@@_patch_preamble_vii:n #1 }
1557         \{ { \@@_patch_preamble_vii:n #1 }
1558         ) { \@@_patch_preamble_viii:n #1 }
1559         ] { \@@_patch_preamble_viii:n #1 }
1560         \} { \@@_patch_preamble_viii:n #1 }
1561         C { \@@_error:nn { old-column-type } #1 }
1562         L { \@@_error:nn { old-column-type } #1 }
1563         R { \@@_error:nn { old-column-type } #1 }
1564         \q_stop { }
1565     }
1566 }

```

```

1567 \str_if_eq:VnTF \l_@@_letter_for_dotted_lines_str { #1 }
1568 { \@@_patch_preamble_xi:n #1 }
1569 {
1570 \str_if_eq:VnTF \l_@@_letter_vlism_tl { #1 }
1571 {
1572 \seq_gput_right:Nx \g_@@_cols_vlism_seq
1573 { \int_eval:n { \c@jCol + 1 } }
1574 \tl_gput_right:Nx \g_@@_preamble_tl
1575 { \exp_not:N ! { \skip_horizontal:N \arrayrulewidth } }
1576 \@@_patch_preamble:n
1577 }
1578 {
1579 \bool_lazy_and:nnTF
1580 { \str_if_eq_p:nn { : } { #1 } }
1581 \c_@@_arydshln_loaded_bool
1582 {
1583 \tl_gput_right:Nn \g_@@_preamble_tl { : }
1584 \@@_patch_preamble:n
1585 }
1586 { \@@_fatal:nn { unknown~column~type } { #1 } }
1587 }
1588 }
1589 }
1590 }

```

For c, l and r

```

1591 \cs_new_protected:Npn \@@_patch_preamble_i:n #1
1592 {
1593 \tl_gput_right:Nn \g_@@_preamble_tl
1594 {
1595 > { \@@_Cell: \tl_set:Nn \l_@@_cell_type_tl { #1 } }
1596 #1
1597 < \@@_end_Cell:
1598 }

```

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

```

1599 \int_gincr:N \c@jCol
1600 \@@_patch_preamble_x:n
1601 }

```

For >, ! and @

```

1602 \cs_new_protected:Npn \@@_patch_preamble_ii:nn #1 #2
1603 {
1604 \tl_gput_right:Nn \g_@@_preamble_tl { #1 { #2 } }
1605 \@@_patch_preamble:n
1606 }

```

For |

```

1607 \cs_new_protected:Npn \@@_patch_preamble_iii:n #1
1608 {

```

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

```

1609 \int_incr:N \l_tmpa_int
1610 \@@_patch_preamble_iii_i:n
1611 }
1612 \cs_new_protected:Npn \@@_patch_preamble_iii_i:n #1
1613 {
1614 \str_if_eq:nnTF { #1 } |
1615 { \@@_patch_preamble_iii:n | }
1616 {
1617 \tl_gput_right:Nx \g_@@_preamble_tl
1618 {
1619 \exp_not:N !

```

```

1620         {
1621             \skip_horizontal:n
1622             {
1623                 \dim_eval:n
1624                 {
1625                     \arrayrulewidth * \l_tmpa_int
1626                     + \doublerulesep * ( \l_tmpa_int - 1)
1627                 }
1628             }
1629         }
1630     }
1631     \tl_gput_right:Nx \g_@@_internal_code_after_tl
1632     { \@@_vline:nn { \@@_succ:n \c@jCol } { \int_use:N \l_tmpa_int } }
1633     \int_zero:N \l_tmpa_int
1634     \@@_patch_preamble:n #1
1635 }
1636 }

```

For p, m and b

```

1637 \cs_new_protected:Npn \@@_patch_preamble_iv:nnn #1 #2 #3
1638 {
1639     \tl_gput_right:Nn \g_@@_preamble_tl
1640     {
1641         > {
1642             \@@_Cell:
1643             \begin { minipage } [ #1 ] { #3 }
1644             \mode_leave_vertical:
1645             \arraybackslash
1646             \vrule height \box_ht:N \@arstrutbox depth 0 pt width 0 pt % v. 5.11
1647         }
1648         c
1649         < {
1650             \vrule height 0 pt depth \box_dp:N \@arstrutbox width 0 pt % v. 5.11
1651             \end { minipage }
1652             \@@_end_Cell:
1653         }
1654     }

```

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

```

1655     \int_gincr:N \c@jCol
1656     \@@_patch_preamble_x:n
1657 }

```

For w and W

```

1658 \cs_new_protected:Npn \@@_patch_preamble_v:nnnn #1 #2 #3 #4
1659 {
1660     \tl_gput_right:Nn \g_@@_preamble_tl
1661     {
1662         > {
1663             \hbox_set:Nw \l_@@_cell_box
1664             \@@_Cell:
1665             \tl_set:Nn \l_@@_cell_type_tl { #3 }
1666         }
1667         c
1668         < {
1669             \@@_end_Cell:
1670             #1
1671             \hbox_set_end:
1672             \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
1673             \@@_adjust_size_box:
1674             \makebox [ #4 ] [ #3 ] { \box_use_drop:N \l_@@_cell_box }
1675         }
1676     }

```

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

```
1677 \int_gincr:N \c@jCol
1678 \@@_patch_preamble_x:n
1679 }
```

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

```
1680 \cs_new_protected:Npn \@@_patch_preamble_vi:n #1
1681 {
1682 \tl_gput_right:Nn \g_@@_preamble_tl { c }
```

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

```
1683 \int_gincr:N \c@jCol
1684 \@@_patch_preamble_x:n
1685 }
```

For (, [ and \{.

```
1686 \cs_new_protected:Npn \@@_patch_preamble_vii:n #1
1687 {
1688 \bool_if:NT \l_@@_small_bool
1689 { \@@_fatal:n { Delimiter-with-small } }
```

If we are before the column 1, we reserve space for the left delimiter.

```
1690 \int_compare:nNnT \c@jCol = \c_zero_int
1691 { \tl_gput_right:Nx \g_@@_preamble_tl { ! { \enskip } } }
1692 \tl_gput_right:Nx \g_@@_internal_code_after_tl
1693 { \@@_delimiter:nnn #1 { \@@_succ:n \c@jCol } \c_true_bool }
1694 \@@_patch_preamble:n
1695 }
```

For ), ] and \}.

```
1696 \cs_new_protected:Npn \@@_patch_preamble_viii:n #1
1697 {
1698 \bool_if:NT \l_@@_small_bool
1699 { \@@_fatal:n { Delimiter-with-small } }
1700 \tl_gput_right:Nx \g_@@_internal_code_after_tl
1701 { \@@_delimiter:nnn #1 { \int_use:N \c@jCol } \c_false_bool }
```

After this closing delimiter, we have to test whether there is a opening delimiter (we consider that there can't be a second closing delimiter) in order to add an horizontal space (equal to 0.5 em).

```
1702 \@@_patch_preamble_viii_i:n
1703 }

1704 \cs_new_protected:Npn \@@_patch_preamble_viii_i:n #1
1705 {
1706 \bool_lazy_any:nT
1707 {
1708 { \str_if_eq_p:nn { #1 } ( }
1709 { \str_if_eq_p:nn { #1 } [ }
1710 { \str_if_eq_p:nn { #1 } \{ }
1711 { \str_if_eq_p:nn { #1 } \q_stop }
1712 }
1713 { \tl_gput_right:Nn \g_@@_preamble_tl { ! { \enskip } } }
1714 \@@_patch_preamble:n #1
1715 }

1716 \cs_new_protected:Npn \@@_patch_preamble_xi:n #1
1717 {
1718 \tl_gput_right:Nn \g_@@_preamble_tl
1719 { ! { \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.

```
1720 \tl_gput_right:Nx \g_@@_internal_code_after_tl
1721 { \@@_vdottedline:n { \int_use:N \c@jCol } }
1722 \@@_patch_preamble:n
1723 }
```

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.

```

1724 \cs_new_protected:Npn \@@_patch_preamble_x:n #1
1725 {
1726   \str_if_eq:nnTF { #1 } { < }
1727   \@@_patch_preamble_ix:n
1728   {
1729     \tl_if_eq:NnTF \l_@@_vlines_clist { all }
1730     {
1731       \tl_gput_right:Nn \g_@@_preamble_tl
1732       { ! { \skip_horizontal:N \arrayrulewidth } }
1733     }
1734     {
1735       \exp_args:NNx
1736       \clist_if_in:NnT \l_@@_vlines_clist { \@@_succ:n \c@jCol }
1737       {
1738         \tl_gput_right:Nn \g_@@_preamble_tl
1739         { ! { \skip_horizontal:N \arrayrulewidth } }
1740       }
1741     }
1742     \@@_patch_preamble:n { #1 }
1743   }
1744 }
1745 \cs_new_protected:Npn \@@_patch_preamble_ix:n #1
1746 {
1747   \tl_gput_right:Nn \g_@@_preamble_tl { < { #1 } }
1748   \@@_patch_preamble_x:n
1749 }

```

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).

```

1750 \cs_new_protected:Npn \@@_put_box_in_flow:
1751 {
1752   \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }
1753   \box_set_dp:Nn \l_tmpa_box { \box_dp:N \l_tmpa_box + \l_tmpb_dim }
1754   \tl_if_eq:NnTF \l_@@_baseline_tl { c }
1755   { \box_use_drop:N \l_tmpa_box }
1756   \@@_put_box_in_flow_i:
1757 }

```

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).

```

1758 \cs_new_protected:Npn \@@_put_box_in_flow_i:
1759 {
1760   \pgfpicture
1761   \@@_qpoint:n { row - 1 }
1762   \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1763   \@@_qpoint:n { row - \@@_succ:n \c@iRow }
1764   \dim_gadd:Nn \g_tmpa_dim \pgf@y
1765   \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).

```

1766   \str_if_in:NnTF \l_@@_baseline_tl { line- }
1767   {
1768     \int_set:Nn \l_tmpa_int
1769     {
1770       \str_range:Nnn
1771       \l_@@_baseline_tl
1772       6

```

```

1773         { \tl_count:V \l_@@_baseline_tl }
1774     }
1775     \@@_qpoint:n { row - \int_use:N \l_tmpa_int }
1776 }
1777 {
1778     \str_case:VnF \l_@@_baseline_tl
1779     {
1780         { t } { \int_set:Nn \l_tmpa_int 1 }
1781         { b } { \int_set_eq:NN \l_tmpa_int \c@iRow }
1782     }
1783     { \int_set:Nn \l_tmpa_int \l_@@_baseline_tl }
1784     \bool_lazy_or:nnT
1785     { \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int }
1786     { \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int }
1787     {
1788         \@@_error:n { bad-value-for-baseline }
1789         \int_set:Nn \l_tmpa_int 1
1790     }
1791     \@@_qpoint:n { row - \int_use:N \l_tmpa_int - base }

```

We take into account the position of the mathematical axis.

```

1792     \dim_gsub:Nn \g_tmpa_dim { \fontdimen22 \textfont2 }
1793 }
1794 \dim_gsub:Nn \g_tmpa_dim \pgf@y

```

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

```

1795     \endpgfpicture
1796     \box_move_up:nn \g_tmpa_dim { \box_use_drop:N \l_tmpa_box }
1797     \box_use_drop:N \l_tmpa_box
1798 }

```

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).

```

1799 \cs_new_protected:Npn \@@_use_arraybox_with_notes_c:
1800 {

```

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

```

1801     \begin { minipage } [ t ] { \box_wd:N \l_@@_the_array_box }
1802     \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.

```

1803     \@@_create_extra_nodes:
1804     \seq_if_empty:NF \g_@@_blocks_seq \@@_draw_blocks:
1805     \bool_lazy_or:nnT
1806     { \int_compare_p:nNn \c@tabularnote > 0 }
1807     { ! \tl_if_empty_p:V \l_@@_tabularnote_tl }
1808     \@@_insert_tabularnotes:
1809     \end { minipage }
1810 }
1811 \cs_new_protected:Npn \@@_insert_tabularnotes:
1812 {
1813     \skip_vertical:N 0.65ex

```

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

```

1814     \group_begin:
1815     \l_@@_notes_code_before_tl
1816     \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.

```

1817 \int_compare:nNnT \c@tabularnote > 0
1818 {
1819   \bool_if:NTF \l_@@_notes_para_bool
1820   {
1821     \begin { tabularnotes* }
1822     \seq_map_inline:Nn \g_@@_tabularnotes_seq { \item ##1 } \strut
1823     \end { tabularnotes* }

```

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

```

1824     \par
1825   }
1826 {
1827   \tabularnotes
1828   \seq_map_inline:Nn \g_@@_tabularnotes_seq { \item ##1 } \strut
1829   \endtabularnotes
1830 }
1831 }
1832 \unskip
1833 \group_end:
1834 \bool_if:NT \l_@@_notes_bottomrule_bool
1835 {
1836   \bool_if:NTF \c_@@_booktabs_loaded_bool
1837   {

```

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

```

1838     \skip_vertical:N \aboverulesep
\CT@arc@ is the specification of color defined by colortbl but you use it even if colortbl is not loaded.
1839     { \CT@arc@ \hrule height \heavyrulewidth }
1840   }
1841   { @@_error:n { bottomrule-without-booktabs } }
1842 }
1843 \l_@@_notes_code_after_tl
1844 \seq_gclear:N \g_@@_tabularnotes_seq
1845 \int_gzero:N \c@tabularnote
1846 }

```

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`.

```

1847 \cs_new_protected:Npn @@_use_arraybox_with_notes_b:
1848 {
1849   \pgfpicture
1850   @@_qpoint:n { row - 1 }
1851   \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1852   @@_qpoint:n { row - \int_use:N \c@iRow - base }
1853   \dim_gsub:Nn \g_tmpa_dim \pgf@y
1854   \endpgfpicture
1855   \dim_gadd:Nn \g_tmpa_dim \arrayrulewidth
1856   \int_compare:nNnT \l_@@_first_row_int = 0
1857   {
1858     \dim_gadd:Nn \g_tmpa_dim \g_@@_ht_row_zero_dim
1859     \dim_gadd:Nn \g_tmpa_dim \g_@@_dp_row_zero_dim
1860   }
1861   \box_move_up:nn \g_tmpa_dim { \hbox { @@_use_arraybox_with_notes_c: } }
1862 }

```

Now, the general case.

```

1863 \cs_new_protected:Npn @@_use_arraybox_with_notes:
1864 {

```



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

```
1865 \tl_if_eq:NnT \l_@@_baseline_tl { t }
1866 { \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`.

```
1867 \pgfpicture
1868 \@@_qpoint:n { row - 1 }
1869 \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1870 \str_if_in:NnTF \l_@@_baseline_tl { line- }
1871 {
1872   \int_set:Nn \l_tmpa_int
1873   {
1874     \str_range:Nnn
1875     \l_@@_baseline_tl
1876     6
1877     { \tl_count:V \l_@@_baseline_tl }
1878   }
1879   \@@_qpoint:n { row - \int_use:N \l_tmpa_int }
1880 }
1881 {
1882   \int_set:Nn \l_tmpa_int \l_@@_baseline_tl
1883   \bool_lazy_or:nnT
1884   { \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int }
1885   { \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int }
1886   {
1887     \@@_error:n { bad~value~for~baseline }
1888     \int_set:Nn \l_tmpa_int 1
1889   }
1890   \@@_qpoint:n { row - \int_use:N \l_tmpa_int - base }
1891 }
1892 \dim_gsub:Nn \g_tmpa_dim \pgf@y
1893 \endpgfpicture
1894 \dim_gadd:Nn \g_tmpa_dim \arrayrulewidth
1895 \int_compare:nNnT \l_@@_first_row_int = 0
1896 {
1897   \dim_gadd:Nn \g_tmpa_dim \g_@@_ht_row_zero_dim
1898   \dim_gadd:Nn \g_tmpa_dim \g_@@_dp_row_zero_dim
1899 }
1900 \box_move_up:nn \g_tmpa_dim { \hbox { \@@_use_arraybox_with_notes_c: } }
1901 }
```

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.

```
1902 \cs_new_protected:Npn \@@_put_box_in_flow_bis:nn #1 #2
1903 {
```

We will compute the real width of both delimiters used.

```
1904 \dim_zero_new:N \l_@@_real_left_delim_dim
1905 \dim_zero_new:N \l_@@_real_right_delim_dim
1906 \hbox_set:Nn \l_tmpb_box
1907 {
1908   \c_math_toggle_token
1909   \left #1
1910   \vcenter
1911   {
1912     \vbox_to_ht:nn
1913     { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1914     { }
1915   }
1916   \right .
1917   \c_math_toggle_token
```

```

1918     }
1919     \dim_set:Nn \l_@@_real_left_delim_dim
1920     { \box_wd:N \l_tmpb_box - \nulldelimiterspace }
1921     \hbox_set:Nn \l_tmpb_box
1922     {
1923       \c_math_toggle_token
1924       \left .
1925       \vbox_to_ht:nn
1926       { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1927       { }
1928       \right #2
1929       \c_math_toggle_token
1930     }
1931     \dim_set:Nn \l_@@_real_right_delim_dim
1932     { \box_wd:N \l_tmpb_box - \nulldelimiterspace }

```

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

```

1933     \skip_horizontal:N \l_@@_left_delim_dim
1934     \skip_horizontal:N -\l_@@_real_left_delim_dim
1935     \@@_put_box_in_flow:
1936     \skip_horizontal:N \l_@@_right_delim_dim
1937     \skip_horizontal:N -\l_@@_real_right_delim_dim
1938   }

```

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).

```

1939 \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).

```

1940   {
1941     \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`).

```

1942     { \exp_args:NV \@@_array: \g_@@_preamble_tl }
1943   }
1944   {
1945     \@@_create_col_nodes:
1946     \endarray
1947   }

```

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`).

```

1948 \NewDocumentEnvironment { @@-light-syntax } { b }
1949   {

```

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.

```

1950     \tl_if_empty:nT { #1 } { \@@_fatal:n { empty-environment } }
1951     \tl_map_inline:nn { #1 }
1952     {
1953       \str_if_eq:nnT { ##1 } { & }
1954       { \@@_fatal:n { ampersand-in-light-syntax } }
1955       \str_if_eq:nnT { ##1 } { \ }
1956       { \@@_fatal:n { double-backslash-in-light-syntax } }
1957     }

```

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`.

```
1958   \@@_light_syntax_i #1 \CodeAfter \q_stop
1959 }
```

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`.

```
1960 { }

1961 \cs_new_protected:Npn \@@_light_syntax_i #1\CodeAfter #2\q_stop
1962 {
1963   \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).

```
1964   \seq_gclear_new:N \g_@@_rows_seq
1965   \tl_set_rescan:Nno \l_@@_end_of_row_tl { } \l_@@_end_of_row_tl
1966   \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.

```
1967   \int_compare:nNnT \l_@@_last_row_int = { -1 }
1968     { \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`).

```
1969   \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.

```
1970   \seq_gpop_left:NN \g_@@_rows_seq \l_tmpa_tl
1971   \exp_args:NV \@@_line_with_light_syntax_i:n \l_tmpa_tl
1972   \seq_map_function:NN \g_@@_rows_seq \@@_line_with_light_syntax:n
1973   \@@_create_col_nodes:
1974   \endarray
1975 }

1976 \cs_new_protected:Npn \@@_line_with_light_syntax:n #1
1977 { \tl_if_empty:nF { #1 } { \ \ \@@_line_with_light_syntax_i:n { #1 } } }

1978 \cs_new_protected:Npn \@@_line_with_light_syntax_i:n #1
1979 {
1980   \seq_gclear_new:N \g_@@_cells_seq
1981   \seq_gset_split:Nnn \g_@@_cells_seq { ~ } { #1 }
1982   \seq_gpop_left:NN \g_@@_cells_seq \l_tmpa_tl
1983   \l_tmpa_tl
1984   \seq_map_inline:Nn \g_@@_cells_seq { & ##1 }
1985 }
```

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).

```
1986 \cs_new_protected:Npn \@@_analyze_end:Nn #1 #2
1987 {
1988   \str_if_eq:VnT \g_@@_name_env_str { #2 }
1989     { \@@_fatal:n { empty-environment } } }
```

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

```
1990   \end { #2 }
1991 }
```

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

```

1992 \cs_new:Npn \@@_create_col_nodes:
1993 {
1994   \crcr
1995   \int_compare:nNnT \l_@@_first_col_int = 0
1996   {
1997     \omit
1998     \hbox_overlap_left:n
1999       {
2000         \bool_if:NT \l_@@_code_before_bool
2001           { \pgfsys@markposition { \@@_env: - col - 0 } }
2002         \pgfpicture
2003         \pgfrememberpicturerepositiononpagetrue
2004         \pgfcoordinate { \@@_env: - col - 0 } \pgfpointorigin
2005         \str_if_empty:NF \l_@@_name_str
2006           { \pgfnodealias { \l_@@_name_str - col - 0 } { \@@_env: - col - 0 } }
2007         \endpgfpicture
2008         \skip_horizontal:N 2\col@sep
2009         \skip_horizontal:N \g_@@_width_first_col_dim
2010       }
2011     &
2012   }
2013   \omit

```

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

```

2014   \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`).

```

2015   \int_compare:nNnTF \l_@@_first_col_int = 0
2016   {
2017     \bool_if:NT \l_@@_code_before_bool
2018     {
2019       \hbox
2020       {
2021         \skip_horizontal:N -0.5\arrayrulewidth
2022         \pgfsys@markposition { \@@_env: - col - 1 }
2023         \skip_horizontal:N 0.5\arrayrulewidth
2024       }
2025     }
2026     \pgfpicture
2027     \pgfrememberpicturerepositiononpagetrue
2028     \pgfcoordinate { \@@_env: - col - 1 }
2029       { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
2030     \str_if_empty:NF \l_@@_name_str
2031       { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
2032     \endpgfpicture
2033   }
2034   {
2035     \bool_if:NT \l_@@_code_before_bool
2036     {
2037       \hbox
2038       {
2039         \skip_horizontal:N 0.5\arrayrulewidth
2040         \pgfsys@markposition { \@@_env: - col - 1 }
2041         \skip_horizontal:N -0.5\arrayrulewidth
2042       }
2043     }
2044     \pgfpicture
2045     \pgfrememberpicturerepositiononpagetrue
2046     \pgfcoordinate { \@@_env: - col - 1 }
2047       { \pgfpoint { 0.5 \arrayrulewidth } \c_zero_dim }

```

```

2048 \str_if_empty:NF \l_@@_name_str
2049 { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
2050 \endpgfpicture
2051 }

```

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.

```

2052 \skip_gset:Nn \g_tmpa_skip { 0 pt+plus 1 fill }
2053 \bool_if:NF \l_@@_auto_columns_width_bool
2054 { \dim_compare:nNnT \l_@@_columns_width_dim > \c_zero_dim }
2055 {
2056   \bool_lazy_and:nnTF
2057   \l_@@_auto_columns_width_bool
2058   { \bool_not_p:n \l_@@_block_auto_columns_width_bool }
2059   { \skip_gset_eq:NN \g_tmpa_skip \g_@@_max_cell_width_dim }
2060   { \skip_gset_eq:NN \g_tmpa_skip \l_@@_columns_width_dim }
2061   \skip_gadd:Nn \g_tmpa_skip { 2 \col@sep }
2062 }
2063 \skip_horizontal:N \g_tmpa_skip
2064 \hbox
2065 {
2066   \bool_if:NT \l_@@_code_before_bool
2067   {
2068     \hbox
2069     {
2070       \skip_horizontal:N -0.5\arrayrulewidth
2071       \pgfsys@markposition { \@@_env: - col - 2 }
2072       \skip_horizontal:N 0.5\arrayrulewidth
2073     }
2074   }
2075   \pgfpicture
2076   \pgfrememberpicturepositiononpagetrue
2077   \pgfcoordinate { \@@_env: - col - 2 }
2078   { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
2079   \str_if_empty:NF \l_@@_name_str
2080   { \pgfnodealias { \l_@@_name_str - col - 2 } { \@@_env: - col - 2 } }
2081   \endpgfpicture
2082 }

```

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.

```

2083 \int_gset:Nn \g_tmpa_int 1
2084 \bool_if:NTF \g_@@_last_col_found_bool
2085 { \prg_replicate:nn { \g_@@_col_total_int - 2 } }
2086 { \prg_replicate:nn { \g_@@_col_total_int - 1 } }
2087 {
2088   &
2089   \omit
2090   \int_gincr:N \g_tmpa_int

```

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

```

2091 \skip_horizontal:N \g_tmpa_skip
2092 \bool_if:NT \l_@@_code_before_bool
2093 {
2094   \hbox
2095   {
2096     \skip_horizontal:N -0.5\arrayrulewidth
2097     \pgfsys@markposition { \@@_env: - col - \@@_succ:n \g_tmpa_int }
2098     \skip_horizontal:N 0.5\arrayrulewidth
2099   }

```

```

2100     }
We create the col node on the right of the current column.
2101     \pgfpicture
2102     \pgfrememberpicturepositiononpagetrue
2103     \pgfcoordinate { \l_@@_env: - col - \l_@@_succ:n \g_tmpa_int }
2104     { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
2105     \str_if_empty:NF \l_@@_name_str
2106     {
2107         \pgfnodealias
2108         { \l_@@_name_str - col - \l_@@_succ:n \g_tmpa_int }
2109         { \l_@@_env: - col - \l_@@_succ:n \g_tmpa_int }
2110     }
2111     \endpgfpicture
2112 }
2113 \bool_if:NT \g_@@_last_col_found_bool
2114 {
2115     \hbox_overlap_right:n
2116     {
2117         % \skip_horizontal:N \col@sep
2118         \skip_horizontal:N \g_@@_width_last_col_dim
2119         \bool_if:NT \l_@@_code_before_bool
2120         {
2121             \pgfsys@markposition
2122             { \l_@@_env: - col - \l_@@_succ:n \g_@@_col_total_int }
2123         }
2124         \pgfpicture
2125         \pgfrememberpicturepositiononpagetrue
2126         \pgfcoordinate { \l_@@_env: - col - \l_@@_succ:n \g_@@_col_total_int }
2127         \pgfpointorigin
2128         \str_if_empty:NF \l_@@_name_str
2129         {
2130             \pgfnodealias
2131             { \l_@@_name_str - col - \l_@@_succ:n \g_@@_col_total_int }
2132             { \l_@@_env: - col - \l_@@_succ:n \g_@@_col_total_int }
2133         }
2134         \endpgfpicture
2135     }
2136 }
2137 \cr
2138 }

```

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

```

2139 \tl_const:Nn \c_@@_preamble_first_col_tl
2140 {
2141     >
2142     {

```

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`).

```

2143     \cs_set_eq:NN \CodeAfter \l_@@_CodeAfter_i:n
2144     \bool_gset_true:N \g_@@_after_col_zero_bool
2145     \l_@@_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.

```

2146     \hbox_set:Nw \l_@@_cell_box
2147     \l_@@_math_toggle_token:
2148     \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”.

```

2149     \bool_lazy_and:nnT
2150     { \int_compare_p:nNn \c@iRow > 0 }

```

```

2151     {
2152         \bool_lazy_or_p:nn
2153         { \int_compare_p:nNn \l_@@_last_row_int < 0 }
2154         { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
2155     }
2156     {
2157         \l_@@_code_for_first_col_tl
2158         \xglobal \colorlet { nicematrix-first-col } { . }
2159     }
2160 }

```

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`.

```

2161     l
2162     <
2163     {
2164         \@@_math_toggle_token:
2165         \hbox_set_end:
2166         \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
2167         \@@_adjust_size_box:
2168         \@@_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.

```

2169         \dim_gset:Nn \g_@@_width_first_col_dim
2170         { \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.

```

2171         \hbox_overlap_left:n
2172         {
2173             \dim_compare:nNnTF { \box_wd:N \l_@@_cell_box } > \c_zero_dim
2174             \@@_node_for_the_cell:
2175             { \box_use_drop:N \l_@@_cell_box }
2176             \skip_horizontal:N \l_@@_left_delim_dim
2177             \skip_horizontal:N \l_@@_left_margin_dim
2178             \skip_horizontal:N \l_@@_extra_left_margin_dim
2179         }
2180         \bool_gset_false:N \g_@@_empty_cell_bool
2181         \skip_horizontal:N -2\col@sep
2182     }
2183 }

```

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

```

2184 \tl_const:Nn \c_@@_preamble_last_col_tl
2185 {
2186     >
2187     {

```

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`).

```

2188         \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.

```

2189         \bool_gset_true:N \g_@@_last_col_found_bool
2190         \int_gincr:N \c@jCol
2191         \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.

```

2192         \hbox_set:Nw \l_@@_cell_box
2193         \@@_math_toggle_token:
2194         \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”.

```

2195     \int_compare:nNnT \c@iRow > 0
2196     {
2197         \bool_lazy_or:nnT
2198         { \int_compare_p:nNn \l_@@_last_row_int < 0 }
2199         { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
2200         {
2201             \l_@@_code_for_last_col_tl
2202             \xglobal \colorlet { nicematrix-last-col } { . }
2203         }
2204     }
2205 }
2206 1
2207 <
2208 {
2209     \@@_math_toggle_token:
2210     \hbox_set_end:
2211     \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
2212     \@@_adjust_size_box:
2213     \@@_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.

```

2214     \dim_gset:Nn \g_@@_width_last_col_dim
2215     { \dim_max:nn \g_@@_width_last_col_dim { \box_wd:N \l_@@_cell_box } }
2216     \skip_horizontal:N -2\col@sep

```

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

```

2217     \hbox_overlap_right:n
2218     {
2219         \dim_compare:nNnT { \box_wd:N \l_@@_cell_box } > \c_zero_dim
2220         {
2221             \skip_horizontal:N \l_@@_right_delim_dim
2222             \skip_horizontal:N \l_@@_right_margin_dim
2223             \skip_horizontal:N \l_@@_extra_right_margin_dim
2224             \@@_node_for_the_cell:
2225         }
2226     }
2227     \bool_gset_false:N \g_@@_empty_cell_bool
2228 }
2229 }

```

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.

```

2230 \NewDocumentEnvironment { NiceArray } { }
2231 {
2232     \bool_set_true:N \l_@@_NiceArray_bool
2233     \str_if_empty:NT \g_@@_name_env_str
2234     { \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).

```

2235     \NiceArrayWithDelims . .
2236 }
2237 { \endNiceArrayWithDelims }

```

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

```

2238 \cs_new_protected:Npn \@@_def_env:nnn #1 #2 #3
2239 {

```



```

2240 \NewDocumentEnvironment { #1 NiceArray } { }
2241 {
2242   \str_if_empty:NT \g_@@_name_env_str
2243   { \str_gset:Nn \g_@@_name_env_str { #1 NiceArray } }
2244   \@@_test_if_math_mode:
2245   \NiceArrayWithDelims #2 #3
2246 }
2247 { \endNiceArrayWithDelims }
2248 }
2249 \@@_def_env:nnn p ( )
2250 \@@_def_env:nnn b [ ]
2251 \@@_def_env:nnn B \{ \}
2252 \@@_def_env:nnn v | |
2253 \@@_def_env:nnn V \| \|

```

## The environment {NiceMatrix} and its variants

```

2254 \cs_new_protected:Npn \@@_begin_of_NiceMatrix:nn #1 #2
2255 {
2256   \bool_set_true:N \l_@@_Matrix_bool
2257   \use:c { #1 NiceArray }
2258   {
2259     *
2260     {
2261       \int_compare:nNnTF \l_@@_last_col_int < 0
2262       \c@MaxMatrixCols
2263       { \@@_pred:n \l_@@_last_col_int }
2264     }
2265     { > \@@_Cell: #2 < \@@_end_Cell: }
2266   }
2267 }
2268 \clist_map_inline:nn { { } , p , b , B , v , V }
2269 {
2270   \NewDocumentEnvironment { #1 NiceMatrix } { ! 0 { } }
2271   {
2272     \str_gset:Nn \g_@@_name_env_str { #1 NiceMatrix }
2273     \tl_set:Nn \l_@@_type_of_col_tl c
2274     \keys_set:nn { NiceMatrix / NiceMatrix } { ##1 }
2275     \exp_args:Nne \@@_begin_of_NiceMatrix:nn { #1 } \l_@@_type_of_col_tl
2276   }
2277   { \use:c { end #1 NiceArray } }
2278 }

```

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

```

2279 \cs_new_protected:Npn \@@_NotEmpty:
2280 { \bool_gset_true:N \g_@@_not_empty_cell_bool }

```

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

```

2281 \NewDocumentEnvironment { NiceTabular } { 0 { } m ! 0 { } }
2282 {
2283   \str_gset:Nn \g_@@_name_env_str { NiceTabular }
2284   \keys_set:nn { NiceMatrix / NiceTabular } { #1 , #3 }
2285   \bool_set_true:N \l_@@_NiceTabular_bool
2286   \NiceArray { #2 }
2287 }
2288 { \endNiceArray }
2289 \NewDocumentEnvironment { NiceTabular* } { m 0 { } m ! 0 { } }

```

```

2290 {
2291   \str_gset:Nn \g_@@_name_env_str { NiceTabular* }
2292   \dim_set:Nn \l_@@_tabular_width_dim { #1 }
2293   \keys_set:nn { NiceMatrix / NiceTabular } { #2 , #4 }
2294   \bool_set_true:N \l_@@_NiceTabular_bool
2295   \NiceArray { #3 }
2296 }
2297 { \endNiceArray }

```

## After the construction of the array

```

2298 \cs_new_protected:Npn \@@_after_array:
2299 {
2300   \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.

```

2301   \bool_if:NT \g_@@_last_col_found_bool
2302   { \int_set_eq:NN \l_@@_last_col_int \g_@@_col_total_int }

```

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

```

2303   \bool_if:NT \l_@@_last_col_without_value_bool
2304   {
2305     \dim_set_eq:NN \l_@@_last_col_int \g_@@_col_total_int
2306     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
2307     \iow_shipout:Nx \@mainaux
2308     {
2309       \cs_gset:cpn { @@_last_col_ \int_use:N \g_@@_env_int }
2310       { \int_use:N \g_@@_col_total_int }
2311     }
2312     \str_if_empty:NF \l_@@_name_str
2313     {
2314       \iow_shipout:Nx \@mainaux
2315       {
2316         \cs_gset:cpn { @@_last_col_ \l_@@_name_str }
2317         { \int_use:N \g_@@_col_total_int }
2318       }
2319     }
2320     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
2321   }

```

It's also time to give to `\l_@@_last_row_int` its real value. But, if the user had used the option `last-row` without value, we write in the aux file the number of that last row for the next run.

```

2322   \bool_if:NT \l_@@_last_row_without_value_bool
2323   {
2324     \dim_set_eq:NN \l_@@_last_row_int \g_@@_row_total_int

```

If the option `light-syntax` is used, we have nothing to write since, in this case, the number of rows is directly determined.

```

2325   \bool_if:NF \l_@@_light_syntax_bool
2326   {
2327     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
2328     \iow_shipout:Nx \@mainaux
2329     {
2330       \cs_gset:cpn { @@_last_row_ \int_use:N \g_@@_env_int }
2331       { \int_use:N \g_@@_row_total_int }
2332     }

```

If the environment has a name, we also write a value based on the name because it's more reliable than a value based on the number of the environment.

```

2333     \str_if_empty:NF \l_@@_name_str
2334     {
2335         \iow_shipout:Nx \@mainaux
2336         {
2337             \cs_gset:cpn { @@_last_row_ \l_@@_name_str }
2338             { \int_use:N \g_@@_row_total_int }
2339         }
2340     }
2341     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
2342 }
2343 }

```

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

```

2344     \bool_if:NT \l_@@_code_before_bool
2345     {
2346         \iow_now:Nn \@mainaux \ExplSyntaxOn
2347         \iow_now:Nx \@mainaux
2348         { \seq_clear_new:c { @@_size _ \int_use:N \g_@@_env_int _ seq } }
2349         \iow_now:Nx \@mainaux
2350         {
2351             \seq_gset_from_clist:cn { @@_size _ \int_use:N \g_@@_env_int _ seq }
2352             {
2353                 \int_use:N \l_@@_first_row_int ,
2354                 \int_use:N \g_@@_row_total_int ,
2355                 \int_use:N \l_@@_first_col_int ,

```

If the user has used a key last-row in an environment with preamble (like {pNiceArray}) and that that last row has not been found, we have to increment the value because it will be decreased when used in the code-before.

```

2356         \bool_lazy_and:nnTF
2357         { \int_compare_p:nNn \l_@@_last_col_int > { -2 } }
2358         { \bool_not_p:n \g_@@_last_col_found_bool }
2359         \@@_succ:n
2360         \int_use:N
2361         \g_@@_col_total_int
2362     }

```

We write also the potential content of \g\_@@\_pos\_of\_blocks\_seq (it will be useful if the commands \rowcolors is used with the key respect-blocks).

```

2363         \seq_gset_from_clist:cn
2364         { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq }
2365         { \seq_use:Nnnn \g_@@_pos_of_blocks_seq , , , }
2366     }
2367     \iow_now:Nn \@mainaux \ExplSyntaxOff
2368 }

```

Now, you create the diagonal nodes by using the row nodes and the col nodes. If the engine is xetex or luatex we also create the “½ nodes”.

```

2369     \@@_create_diag_nodes:

```

By default, the diagonal lines will be parallelized<sup>56</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.

```

2370     \bool_if:NT \l_@@_parallelize_diags_bool
2371     {
2372         \int_gzero_new:N \g_@@_ddots_int
2373         \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.

```

2374         \dim_gzero_new:N \g_@@_delta_x_one_dim

```

<sup>56</sup>It's possible to use the option parallelize-diags to disable this parallelization.

```

2375     \dim_gzero_new:N \g_@@_delta_y_one_dim
2376     \dim_gzero_new:N \g_@@_delta_x_two_dim
2377     \dim_gzero_new:N \g_@@_delta_y_two_dim
2378   }
2379   \int_zero_new:N \l_@@_initial_i_int
2380   \int_zero_new:N \l_@@_initial_j_int
2381   \int_zero_new:N \l_@@_final_i_int
2382   \int_zero_new:N \l_@@_final_j_int
2383   \bool_set_false:N \l_@@_initial_open_bool
2384   \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.

```

2385     \bool_if:NT \l_@@_small_bool
2386     {
2387       \dim_set:Nn \l_@@_radius_dim { 0.37 pt }
2388       \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.

```

2389       \dim_set:Nn \l_@@_xdots_shorten_dim { 0.6 \l_@@_xdots_shorten_dim }
2390     }

```

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

```

2391     \@@_draw_dotted_lines:

```

The following computes the “corners” (made up of empty cells) but if there is no corner to compute, it will do nothing. The corners are computed in `\l_@@_empty_corner_cells_seq` which will contain all the empty cells (and not in a block) considered in the corners of the array.

```

2392     \@@_compute_corners:

```

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

```

2393     \@@_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`.

```

2394     \bool_lazy_all:nT
2395     {
2396       { \seq_if_empty_p:N \g_@@_pos_of_blocks_seq }
2397       { \seq_if_empty_p:N \g_@@_pos_of_xdots_seq }
2398       { \seq_if_empty_p:N \l_@@_empty_corner_cells_seq }
2399     }
2400     {
2401       \cs_set_eq:NN \@@_vline_i:nn \@@_vline_i_complete:nn
2402       \cs_set_eq:NN \@@_hline_i:nn \@@_hline_i_complete:nn
2403     }
2404     \tl_if_empty:NF \l_@@_hlines_clist \@@_draw_hlines:
2405     \tl_if_empty:NF \l_@@_vlines_clist \@@_draw_vlines:
2406     \cs_set_eq:NN \SubMatrix \@@_SubMatrix

```

Now, the internal code-after and then, the `\CodeAfter`.

```

2407     \bool_if:NT \c_@@_tikz_loaded_bool
2408     {
2409       \tikzset
2410       {
2411         every~picture / .style =

```

```

2412         {
2413             overlay ,
2414             remember~picture ,
2415             name~prefix = \@@_env: -
2416         }
2417     }
2418 }
2419 \cs_set_eq:NN \line \@@_line
2420 \g_@@_internal_code_after_tl
2421 \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.

```

2422 \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).

```

2423 \seq_gclear:N \g_@@_submatrix_names_seq

```

And here's the `\CodeAfter`. Since the `\CodeAfter` may begin with an “argument” between square brackets of the options, we extract and treat that potential “argument” with the command `\@@_CodeAfter_keys:`.

```

2424 \exp_last_unbraced:NV \@@_CodeAfter_keys: \g_nicematrix_code_after_tl
2425 \scan_stop:
2426 \tl_gclear:N \g_nicematrix_code_after_tl
2427 \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.

```

2428 \tl_if_empty:NF \g_nicematrix_code_before_tl
2429 {

```

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`.

```

2430 \cs_set_protected:Npn \rectanglecolor { }
2431 \cs_set_protected:Npn \columncolor { }
2432 \iow_now:Nn \@mainaux \ExplSyntaxOn
2433 \iow_now:Nx \@mainaux
2434 {
2435     \tl_gset:cn
2436     { g_@@_code_before_ \int_use:N \g_@@_env_int _ tl }
2437     { \exp_not:V \g_nicematrix_code_before_tl }
2438 }
2439 \iow_now:Nn \@mainaux \ExplSyntaxOff
2440 \bool_set_true:N \l_@@_code_before_bool
2441 }

2442 \str_gclear:N \g_@@_name_env_str
2443 \@@_restore_iRow_jCol:

```

The command `\CT@arc@` contains the instruction of color for the rules of the array<sup>57</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`.

```

2444 \cs_gset_eq:NN \CT@arc@ \@@_old_CT@arc@
2445 }

```

---

<sup>57</sup>e.g. `\color[rgb]{0.5,0.5,0}`

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`).

```
2446 \NewDocumentCommand \@@_CodeAfter_keys: { 0 { } }
2447 { \keys_set:nn { NiceMatrix / CodeAfter } { #1 } }
```

We remind that the first mandatory argument of the command `\Block` is the size of the block with the special format  $i-j$ . However, the user is allowed to omit  $i$  or  $j$  (or both). This will be interpreted as: the last row (resp. column) of the block will be the last row (resp. column) of the block (without the potential exterior row—resp. column—of the array). By convention, this is stored in `\g_@@_pos_of_blocks_seq` (and `\g_@@_blocks_seq`) as a number of rows (resp. columns) for the block equal to 100. It’s possible, after the construction of the array, to replace these values by the correct ones (since we know the number of rows and columns of the array).

```
2448 \cs_new_protected:Npn \@@_adjust_pos_of_blocks_seq:
2449 {
2450   \seq_gset_map_x:NNn \g_@@_pos_of_blocks_seq \g_@@_pos_of_blocks_seq
2451   { \@@_adjust_pos_of_blocks_seq_i:nnnn #1 }
2452 }
```

The following command must *not* be protected.

```
2453 \cs_new:Npn \@@_adjust_pos_of_blocks_seq_i:nnnn #1 #2 #3 #4
2454 {
2455   { #1 }
2456   { #2 }
2457   {
2458     \int_compare:nNnTF { #3 } > { 99 }
2459     { \int_use:N \c@iRow }
2460     { #3 }
2461   }
2462   {
2463     \int_compare:nNnTF { #4 } > { 99 }
2464     { \int_use:N \c@jCol }
2465     { #4 }
2466   }
2467 }
```

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).

```
2468 \AtBeginDocument
2469 {
2470   \cs_new_protected:Npx \@@_draw_dotted_lines:
2471   {
2472     \c_@@_pgfortikzpicture_tl
2473     \@@_draw_dotted_lines_i:
2474     \c_@@_endpgfortikzpicture_tl
2475   }
2476 }
```

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

```
2477 \cs_new_protected:Npn \@@_draw_dotted_lines_i:
2478 {
2479   \pgfrememberpicturepositiononpagetrue
2480   \pgf@relevantforpicturesizefalse
2481   \g_@@_HVdotsfor_lines_tl
2482   \g_@@_Vdots_lines_tl
2483   \g_@@_Ddots_lines_tl
2484   \g_@@_Iddots_lines_tl
2485   \g_@@_Cdots_lines_tl
2486   \g_@@_Ldots_lines_tl
2487 }
```

```

2488 \cs_new_protected:Npn \@@_restore_iRow_jCol:
2489 {
2490     \cs_if_exist:NT \theiRow { \int_gset_eq:NN \c@iRow \l_@@_old_iRow_int }
2491     \cs_if_exist:NT \thejCol { \int_gset_eq:NN \c@jCol \l_@@_old_jCol_int }
2492 }

```

We define a new PGF shape for the diag nodes because we want to provide a anchor called .5 for those nodes.

```

2493 \pgfdeclareshape { @a_diag_node }
2494 {
2495     \savedanchor { \five }
2496     {
2497         \dim_gset_eq:NN \pgf@x \l_tmpa_dim
2498         \dim_gset_eq:NN \pgf@y \l_tmpb_dim
2499     }
2500     \anchor { 5 } { \five }
2501     \anchor { center } { \pgfpointorigin }
2502 }

```

The following command creates the diagonal nodes (in fact, if the matrix is not a square matrix, not all the nodes are on the diagonal).

```

2503 \cs_new_protected:Npn \@@_create_diag_nodes:
2504 {
2505     \pgfpicture
2506     \pgfrememberpicturepositiononpagetrue
2507     \int_step_inline:nn { \int_max:nn \c@iRow \c@jCol }
2508     {
2509         \@@_qpoint:n { col - \int_min:nn { ##1 } { \c@jCol + 1 } }
2510         \dim_set_eq:NN \l_tmpa_dim \pgf@x
2511         \@@_qpoint:n { row - \int_min:nn { ##1 } { \c@iRow + 1 } }
2512         \dim_set_eq:NN \l_tmpb_dim \pgf@y
2513         \@@_qpoint:n { col - \int_min:nn { ##1 + 1 } { \c@jCol + 1 } }
2514         \dim_set_eq:NN \l_tmpc_dim \pgf@x
2515         \@@_qpoint:n { row - \int_min:nn { ##1 + 1 } { \c@iRow + 1 } }
2516         \dim_set_eq:NN \l_tmpd_dim \pgf@y
2517         \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 @a\_diag\_node) that we will construct.

```

2518         \dim_set:Nn \l_tmpa_dim { ( \l_tmpc_dim - \l_tmpa_dim ) / 2 }
2519         \dim_set:Nn \l_tmpb_dim { ( \l_tmpd_dim - \l_tmpb_dim ) / 2 }
2520         \pgfnode { @a_diag_node } { center } { } { \@@_env: - ##1 } { }
2521     }

```

Now, the last node. Of course, that is only a coordinate because there is not .5 anchor for that node.

```

2522     \int_set:Nn \l_tmpa_int { \int_max:nn \c@iRow \c@jCol + 1 }
2523     \@@_qpoint:n { row - \int_min:nn { \l_tmpa_int } { \c@iRow + 1 } }
2524     \dim_set_eq:NN \l_tmpa_dim \pgf@y
2525     \@@_qpoint:n { col - \int_min:nn { \l_tmpa_int } { \c@jCol + 1 } }
2526     \pgfcoordinate
2527     { \@@_env: - \int_use:N \l_tmpa_int } { \pgfpoint \pgf@x \l_tmpa_dim }
2528     \pgfnodealias
2529     { \@@_env: - last }
2530     { \@@_env: - \int_eval:n { \int_max:nn \c@iRow \c@jCol + 1 } }
2531     \endpgfpicture
2532 }

```

## We draw the dotted lines

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

its right.

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

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

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

This command computes:

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

```
2533 \cs_new_protected:Npn \l_@@_find_extremities_of_line:nnnn #1 #2 #3 #4
2534 {
```

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

```
2535 \cs_set:cpn { @@ _ dotted _ #1 - #2 } { }
```

Initialization of variables.

```
2536 \int_set:Nn \l_@@_initial_i_int { #1 }
2537 \int_set:Nn \l_@@_initial_j_int { #2 }
2538 \int_set:Nn \l_@@_final_i_int { #1 }
2539 \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.

```
2540 \bool_set_false:N \l_@@_stop_loop_bool
2541 \bool_do_until:Nn \l_@@_stop_loop_bool
2542 {
2543   \int_add:Nn \l_@@_final_i_int { #3 }
2544   \int_add:Nn \l_@@_final_j_int { #4 }
```

We test if we are still in the matrix.

```
2545   \bool_set_false:N \l_@@_final_open_bool
2546   \int_compare:nNnTF \l_@@_final_i_int > \l_@@_row_max_int
2547   {
2548     \int_compare:nNnTF { #3 } = 1
2549     { \bool_set_true:N \l_@@_final_open_bool }
2550     {
2551       \int_compare:nNnTF \l_@@_final_j_int > \l_@@_col_max_int
2552       { \bool_set_true:N \l_@@_final_open_bool }
2553     }
2554   }
2555   {
2556     \int_compare:nNnTF \l_@@_final_j_int < \l_@@_col_min_int
2557     {
2558       \int_compare:nNnTF { #4 } = { -1 }
2559       { \bool_set_true:N \l_@@_final_open_bool }
2560     }
2561     {
```



```

2562         \int_compare:nNtT \l_@@_final_j_int > \l_@@_col_max_int
2563         {
2564             \int_compare:nNtT { #4 } = 1
2565             { \bool_set_true:N \l_@@_final_open_bool }
2566         }
2567     }
2568 }
2569 \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.

```

2570 {

```

We do a step backwards.

```

2571     \int_sub:Nn \l_@@_final_i_int { #3 }
2572     \int_sub:Nn \l_@@_final_j_int { #4 }
2573     \bool_set_true:N \l_@@_stop_loop_bool
2574 }

```

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`.

```

2575 {
2576     \cs_if_exist:cTF
2577     {
2578         @@ _ dotted _
2579         \int_use:N \l_@@_final_i_int -
2580         \int_use:N \l_@@_final_j_int
2581     }
2582     {
2583         \int_sub:Nn \l_@@_final_i_int { #3 }
2584         \int_sub:Nn \l_@@_final_j_int { #4 }
2585         \bool_set_true:N \l_@@_final_open_bool
2586         \bool_set_true:N \l_@@_stop_loop_bool
2587     }
2588     {
2589         \cs_if_exist:cTF
2590         {
2591             pgf @ sh @ ns @ \@@_env:
2592             - \int_use:N \l_@@_final_i_int
2593             - \int_use:N \l_@@_final_j_int
2594         }
2595         { \bool_set_true:N \l_@@_stop_loop_bool }

```

If the case is empty, we declare that the cell as non-empty. Indeed, we will draw a dotted line and the cell will be on that dotted line. All the cells of a dotted line have to be marked as “dotted” because we don't want intersections between dotted lines. We recall that the research of the extremities of the lines are all done in the same TeX group (the group of the environment), even though, when the extremities are found, each line is drawn in a TeX group that we will open for the options of the line.

```

2596     {
2597         \cs_set:cpn
2598         {
2599             @@ _ dotted _
2600             \int_use:N \l_@@_final_i_int -
2601             \int_use:N \l_@@_final_j_int
2602         }
2603         { }
2604     }
2605 }
2606 }
2607 }

```

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

```

2608     \bool_set_false:N \l_@@_stop_loop_bool

```

```

2609 \bool_do_until:Nn \l_@@_stop_loop_bool
2610 {
2611   \int_sub:Nn \l_@@_initial_i_int { #3 }
2612   \int_sub:Nn \l_@@_initial_j_int { #4 }
2613   \bool_set_false:N \l_@@_initial_open_bool
2614   \int_compare:nNnTF \l_@@_initial_i_int < \l_@@_row_min_int
2615   {
2616     \int_compare:nNnTF { #3 } = 1
2617     { \bool_set_true:N \l_@@_initial_open_bool }
2618     {
2619       \int_compare:nNnT \l_@@_initial_j_int = { \l_@@_col_min_int -1 }
2620       { \bool_set_true:N \l_@@_initial_open_bool }
2621     }
2622   }
2623   {
2624     \int_compare:nNnTF \l_@@_initial_j_int < \l_@@_col_min_int
2625     {
2626       \int_compare:nNnT { #4 } = 1
2627       { \bool_set_true:N \l_@@_initial_open_bool }
2628     }
2629     {
2630       \int_compare:nNnT \l_@@_initial_j_int > \l_@@_col_max_int
2631       {
2632         \int_compare:nNnT { #4 } = { -1 }
2633         { \bool_set_true:N \l_@@_initial_open_bool }
2634       }
2635     }
2636   }
2637   \bool_if:NnTF \l_@@_initial_open_bool
2638   {
2639     \int_add:Nn \l_@@_initial_i_int { #3 }
2640     \int_add:Nn \l_@@_initial_j_int { #4 }
2641     \bool_set_true:N \l_@@_stop_loop_bool
2642   }
2643   {
2644     \cs_if_exist:cTF
2645     {
2646       @@ _ dotted _
2647       \int_use:N \l_@@_initial_i_int -
2648       \int_use:N \l_@@_initial_j_int
2649     }
2650     {
2651       \int_add:Nn \l_@@_initial_i_int { #3 }
2652       \int_add:Nn \l_@@_initial_j_int { #4 }
2653       \bool_set_true:N \l_@@_initial_open_bool
2654       \bool_set_true:N \l_@@_stop_loop_bool
2655     }
2656     {
2657       \cs_if_exist:cTF
2658       {
2659         pgf @ sh @ ns @ \@@_env:
2660         - \int_use:N \l_@@_initial_i_int
2661         - \int_use:N \l_@@_initial_j_int
2662       }
2663       { \bool_set_true:N \l_@@_stop_loop_bool }
2664       {
2665         \cs_set:cpn
2666         {
2667           @@ _ dotted _
2668           \int_use:N \l_@@_initial_i_int -
2669           \int_use:N \l_@@_initial_j_int
2670         }
2671         { }

```

```

2672         }
2673     }
2674 }
2675 }

```

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.

```

2676 \seq_gput_right:Nx \g_@@_pos_of_xdots_seq
2677 {
2678     { \int_use:N \l_@@_initial_i_int }
2679     { \int_use:N \l_@@_initial_j_int }
2680     { \int_use:N \l_@@_final_i_int }
2681     { \int_use:N \l_@@_final_j_int }
2682 }
2683 }

```

The following command (*when it will be written*) will set the four counters `\l_@@_row_min_int`, `\l_@@_row_max_int`, `\l_@@_col_min_int` and `\l_@@_col_max_int` to the intersections of the submatrices which contains the cell of row #1 and column #2. As of now, it’s only the whole array (excepted exterior row and columns).

```

2684 \cs_new_protected:Npn \@@_adjust_to_submatrix:nn #1 #2
2685 {
2686     \int_set:Nn \l_@@_row_min_int 1
2687     \int_set:Nn \l_@@_col_min_int 1
2688     \int_set_eq:NN \l_@@_row_max_int \c@iRow
2689     \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`.

```

2690 \seq_map_inline:Nn \g_@@_submatrix_seq
2691 { \@@_adjust_to_submatrix:nnnnnn { #1 } { #2 } ##1 }
2692 }

```

#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.

```

2693 \cs_set_protected:Npn \@@_adjust_to_submatrix:nnnnnn #1 #2 #3 #4 #5 #6
2694 {
2695     \bool_if:nT
2696     {
2697         \int_compare_p:n { #3 <= #1 }
2698         && \int_compare_p:n { #1 <= #5 }
2699         && \int_compare_p:n { #4 <= #2 }
2700         && \int_compare_p:n { #2 <= #6 }
2701     }
2702     {
2703         \int_set:Nn \l_@@_row_min_int { \int_max:nn \l_@@_row_min_int { #3 } }
2704         \int_set:Nn \l_@@_col_min_int { \int_max:nn \l_@@_col_min_int { #4 } }
2705         \int_set:Nn \l_@@_row_max_int { \int_min:nn \l_@@_row_max_int { #5 } }
2706         \int_set:Nn \l_@@_col_max_int { \int_min:nn \l_@@_col_max_int { #6 } }
2707     }
2708 }

```

```

2709 \cs_new_protected:Npn \@@_set_initial_coords:
2710 {
2711     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2712     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2713 }
2714 \cs_new_protected:Npn \@@_set_final_coords:
2715 {
2716     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2717     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2718 }

```

```

2719 \cs_new_protected:Npn \@@_set_initial_coords_from_anchor:n #1
2720 {
2721   \pgfpointanchor
2722   {
2723     \@@_env:
2724     - \int_use:N \l_@@_initial_i_int
2725     - \int_use:N \l_@@_initial_j_int
2726   }
2727   { #1 }
2728   \@@_set_initial_coords:
2729 }
2730 \cs_new_protected:Npn \@@_set_final_coords_from_anchor:n #1
2731 {
2732   \pgfpointanchor
2733   {
2734     \@@_env:
2735     - \int_use:N \l_@@_final_i_int
2736     - \int_use:N \l_@@_final_j_int
2737   }
2738   { #1 }
2739   \@@_set_final_coords:
2740 }
2741 \cs_new_protected:Npn \@@_open_x_initial_dim:
2742 {
2743   \dim_set_eq:NN \l_@@_x_initial_dim \c_max_dim
2744   \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
2745   {
2746     \cs_if_exist:cT
2747     { pgf @ sh @ ns @ \@@_env: - ##1 - \int_use:N \l_@@_initial_j_int }
2748     {
2749       \pgfpointanchor
2750       { \@@_env: - ##1 - \int_use:N \l_@@_initial_j_int }
2751       { west }
2752       \dim_set:Nn \l_@@_x_initial_dim
2753       { \dim_min:nn \l_@@_x_initial_dim \pgf@x }
2754     }
2755   }
2756   \dim_compare:nNnT \l_@@_x_initial_dim = \c_max_dim
2757   {
2758     \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2759     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2760     \dim_add:Nn \l_@@_x_initial_dim \col@sep
2761   }
2762 }
2763 \cs_new_protected:Npn \@@_open_x_final_dim:
2764 {
2765   \dim_set:Nn \l_@@_x_final_dim { - \c_max_dim }
2766   \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
2767   {
2768     \cs_if_exist:cT
2769     { pgf @ sh @ ns @ \@@_env: - ##1 - \int_use:N \l_@@_final_j_int }
2770     {
2771       \pgfpointanchor
2772       { \@@_env: - ##1 - \int_use:N \l_@@_final_j_int }
2773       { east }
2774       \dim_set:Nn \l_@@_x_final_dim
2775       { \dim_max:nn \l_@@_x_final_dim \pgf@x }
2776     }
2777   }

```

If, in fact, all the cells of the columns are empty (no PGF/Tikz nodes in those cells).

```

2778 \dim_compare:nNnT \l_@@_x_final_dim = { - \c_max_dim }
2779 {
2780   \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
2781   \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2782   \dim_sub:Nn \l_@@_x_final_dim \col@sep
2783 }
2784 }

```

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.

```

2785 \cs_new_protected:Npn \@@_draw_Ldots:nnn #1 #2 #3
2786 {
2787   \@@_adjust_to_submatrix:nn { #1 } { #2 }
2788   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2789   {
2790     \@@_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.

```

2791   \group_begin:
2792     \int_compare:nNnTF { #1 } = 0
2793     { \color { nicematrix-first-row } }
2794     {

```

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.

```

2795       \int_compare:nNnT { #1 } = \l_@@_last_row_int
2796       { \color { nicematrix-last-row } }
2797     }
2798     \keys_set:nn { NiceMatrix / xdots } { #3 }
2799     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2800     \@@_actually_draw_Ldots:
2801   \group_end:
2802 }
2803 }

```

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`.

```

2804 \cs_new_protected:Npn \@@_actually_draw_Ldots:
2805 {
2806   \bool_if:NTF \l_@@_initial_open_bool
2807   {
2808     \@@_open_x_initial_dim:
2809     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int - base }
2810     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2811   }
2812   { \@@_set_initial_coords_from_anchor:n { base-east } }
2813   \bool_if:NTF \l_@@_final_open_bool
2814   {
2815     \@@_open_x_final_dim:
2816     \@@_qpoint:n { row - \int_use:N \l_@@_final_i_int - base }

```

```

2817     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2818   }
2819   { \@@_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 (?).

```

2820     \dim_add:Nn \l_@@_y_initial_dim \l_@@_radius_dim
2821     \dim_add:Nn \l_@@_y_final_dim \l_@@_radius_dim
2822     \@@_draw_line:
2823   }

```

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.

```

2824 \cs_new_protected:Npn \@@_draw_Cdots:nnn #1 #2 #3
2825 {
2826   \@@_adjust_to_submatrix:nn { #1 } { #2 }
2827   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2828   {
2829     \@@_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.

```

2830     \group_begin:
2831     \int_compare:nNnTF { #1 } = 0
2832       { \color { nicematrix-first-row } }
2833     {

```

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.

```

2834       \int_compare:nNnT { #1 } = \l_@@_last_row_int
2835       { \color { nicematrix-last-row } }
2836     }
2837     \keys_set:nn { NiceMatrix / xdots } { #3 }
2838     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2839     \@@_actually_draw_Cdots:
2840   \group_end:
2841 }
2842 }

```

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`.

```

2843 \cs_new_protected:Npn \@@_actually_draw_Cdots:
2844 {
2845   \bool_if:NTF \l_@@_initial_open_bool
2846     { \@@_open_x_initial_dim: }
2847     { \@@_set_initial_coords_from_anchor:n { mid~east } }
2848   \bool_if:NTF \l_@@_final_open_bool
2849     { \@@_open_x_final_dim: }
2850     { \@@_set_final_coords_from_anchor:n { mid~west } }
2851   \bool_lazy_and:nnTF
2852     \l_@@_initial_open_bool
2853     \l_@@_final_open_bool
2854   {

```

```

2855 \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2856 \dim_set_eq:NN \l_tmpa_dim \pgf@y
2857 \@@_qpoint:n { row - \@@_succ:n \l_@@_initial_i_int }
2858 \dim_set:Nn \l_@@_y_initial_dim { ( \l_tmpa_dim + \pgf@y ) / 2 }
2859 \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim
2860 }
2861 {
2862 \bool_if:NT \l_@@_initial_open_bool
2863 { \dim_set_eq:NN \l_@@_y_initial_dim \l_@@_y_final_dim }
2864 \bool_if:NT \l_@@_final_open_bool
2865 { \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim }
2866 }
2867 \@@_draw_line:
2868 }

2869 \cs_new_protected:Npn \@@_open_y_initial_dim:
2870 {
2871 \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int - base }
2872 \dim_set:Nn \l_@@_y_initial_dim
2873 { \pgf@y + ( \box_ht:N \strutbox + \extrarowheight ) * \arraystretch }
2874 \int_step_inline:nnn \l_@@_first_col_int \g_@@_col_total_int
2875 {
2876 \cs_if_exist:cT
2877 { \pgf @ sh @ ns @ \@@_env: - \int_use:N \l_@@_initial_i_int - ##1 }
2878 {
2879 \pgfpointanchor
2880 { \@@_env: - \int_use:N \l_@@_initial_i_int - ##1 }
2881 { north }
2882 \dim_set:Nn \l_@@_y_initial_dim
2883 { \dim_max:nn \l_@@_y_initial_dim \pgf@y }
2884 }
2885 }
2886 }

2887 \cs_new_protected:Npn \@@_open_y_final_dim:
2888 {
2889 \@@_qpoint:n { row - \int_use:N \l_@@_final_i_int - base }
2890 \dim_set:Nn \l_@@_y_final_dim
2891 { \pgf@y - ( \box_dp:N \strutbox ) * \arraystretch }
2892 \int_step_inline:nnn \l_@@_first_col_int \g_@@_col_total_int
2893 {
2894 \cs_if_exist:cT
2895 { \pgf @ sh @ ns @ \@@_env: - \int_use:N \l_@@_final_i_int - ##1 }
2896 {
2897 \pgfpointanchor
2898 { \@@_env: - \int_use:N \l_@@_final_i_int - ##1 }
2899 { south }
2900 \dim_set:Nn \l_@@_y_final_dim
2901 { \dim_min:nn \l_@@_y_final_dim \pgf@y }
2902 }
2903 }
2904 }

```

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.

```

2905 \cs_new_protected:Npn \@@_draw_Vdots:nnn #1 #2 #3
2906 {
2907 \@@_adjust_to_submatrix:nn { #1 } { #2 }
2908 \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2909 {
2910 \@@_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.

```

2911 \group_begin:

```

```

2912 \int_compare:nNnTF { #2 } = 0
2913 { \color { nicematrix-first-col } }
2914 {
2915 \int_compare:nNnT { #2 } = \l_@@_last_col_int
2916 { \color { nicematrix-last-col } }
2917 }
2918 \keys_set:nn { NiceMatrix / xdots } { #3 }
2919 \tl_if_empty:VF \l_@@_xdots_color_tl
2920 { \color { \l_@@_xdots_color_tl } }
2921 \@@_actually_draw_Vdots:
2922 \group_end:
2923 }
2924 }

```

The command `\@@_actually_draw_Vdots:` has the following implicit arguments:

- `\l_@@_initial_i_int`
- `\l_@@_initial_j_int`
- `\l_@@_initial_open_bool`
- `\l_@@_final_i_int`
- `\l_@@_final_j_int`
- `\l_@@_final_open_bool`.

The following function is also used by `\Vdotsfor`.

```

2925 \cs_new_protected:Npn \@@_actually_draw_Vdots:
2926 {

```

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

```

2927 \bool_set_false:N \l_tmpa_bool

```

First the case when the line is closed on both ends.

```

2928 \bool_lazy_or:nnF \l_@@_initial_open_bool \l_@@_final_open_bool
2929 {
2930 \@@_set_initial_coords_from_anchor:n { south-west }
2931 \@@_set_final_coords_from_anchor:n { north-west }
2932 \bool_set:Nn \l_tmpa_bool
2933 { \dim_compare_p:nNn \l_@@_x_initial_dim = \l_@@_x_final_dim }
2934 }

```

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

```

2935 \bool_if:NNTF \l_@@_initial_open_bool
2936 \@@_open_y_initial_dim:
2937 { \@@_set_initial_coords_from_anchor:n { south } }
2938 \bool_if:NNTF \l_@@_final_open_bool
2939 \@@_open_y_final_dim:
2940 { \@@_set_final_coords_from_anchor:n { north } }
2941 \bool_if:NNTF \l_@@_initial_open_bool
2942 {
2943 \bool_if:NNTF \l_@@_final_open_bool
2944 {
2945 \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2946 \dim_set_eq:NN \l_tmpa_dim \pgf@x
2947 \@@_qpoint:n { col - \@@_succ:n \l_@@_initial_j_int }
2948 \dim_set:Nn \l_@@_x_initial_dim { ( \pgf@x + \l_tmpa_dim ) / 2 }
2949 \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.

```

2950 \int_compare:nNnT \l_@@_last_col_int > { -2 }
2951 {

```



```

2952         \int_compare:nNtT \l_@@_initial_j_int = \g_@@_col_total_int
2953         {
2954             \dim_set_eq:NN \l_tmpa_dim \l_@@_right_margin_dim
2955             \dim_add:Nn \l_tmpa_dim \l_@@_extra_right_margin_dim
2956             \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
2957             \dim_add:Nn \l_@@_x_final_dim \l_tmpa_dim
2958         }
2959     }
2960 }
2961 { \dim_set_eq:NN \l_@@_x_initial_dim \l_@@_x_final_dim }
2962 }
2963 {
2964     \bool_if:NTF \l_@@_final_open_bool
2965     { \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim }
2966     {

```

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.

```

2967         \dim_compare:nNf \l_@@_x_initial_dim = \l_@@_x_final_dim
2968         {
2969             \dim_set:Nn \l_@@_x_initial_dim
2970             {
2971                 \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
2972                 \l_@@_x_initial_dim \l_@@_x_final_dim
2973             }
2974             \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim
2975         }
2976     }
2977 }
2978 \@@_draw_line:
2979 }

```

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.

```

2980 \cs_new_protected:Npn \@@_draw_Ddots:nnn #1 #2 #3
2981 {
2982     \@@_adjust_to_submatrix:nn { #1 } { #2 }
2983     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2984     {
2985         \@@_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.

```

2986     \group_begin:
2987     \keys_set:nn { NiceMatrix / xdots } { #3 }
2988     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2989     \@@_actually_draw_Ddots:
2990     \group_end:
2991 }
2992 }

```

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`.

```

2993 \cs_new_protected:Npn \@@_actually_draw_Ddots:
2994 {
2995   \bool_if:NTF \l_@@_initial_open_bool
2996   {
2997     \@@_open_y_initial_dim:
2998     % \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2999     % \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
3000     \@@_open_x_initial_dim:
3001   }
3002   { \@@_set_initial_coords_from_anchor:n { south-east } }
3003   \bool_if:NTF \l_@@_final_open_bool
3004   {
3005     % \@@_open_y_final_dim:
3006     % \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
3007     \@@_open_x_final_dim:
3008     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
3009   }
3010   { \@@_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.

```

3011   \bool_if:NT \l_@@_parallelize_diags_bool
3012   {
3013     \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).

```

3014     \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.

```

3015     {
3016       \dim_gset:Nn \g_@@_delta_x_one_dim
3017       { \l_@@_x_final_dim - \l_@@_x_initial_dim }
3018       \dim_gset:Nn \g_@@_delta_y_one_dim
3019       { \l_@@_y_final_dim - \l_@@_y_initial_dim }
3020     }

```

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`.

```

3021     {
3022       \dim_set:Nn \l_@@_y_final_dim
3023       {
3024         \l_@@_y_initial_dim +
3025         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
3026         \dim_ratio:nn \g_@@_delta_y_one_dim \g_@@_delta_x_one_dim
3027       }
3028     }
3029   }
3030   \@@_draw_line:
3031 }

```

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.

```

3032 \cs_new_protected:Npn \@@_draw_Iddots:nnn #1 #2 #3
3033 {
3034   \@@_adjust_to_submatrix:nn { #1 } { #2 }
3035   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
3036   {
3037     \@@_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.

```

3038     \group_begin:
3039     \keys_set:nn { NiceMatrix / xdots } { #3 }
3040     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3041     \@@_actually_draw_Iddots:
3042     \group_end:
3043   }
3044 }

```

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`.

```

3045 \cs_new_protected:Npn \@@_actually_draw_Iddots:
3046 {
3047   \bool_if:NTF \l_@@_initial_open_bool
3048   {
3049     \@@_open_y_initial_dim:
3050     \@@_open_x_initial_dim:
3051   }
3052   { \@@_set_initial_coords_from_anchor:n { south-west } }
3053   \bool_if:NTF \l_@@_final_open_bool
3054   {
3055     \@@_open_y_final_dim:
3056     \@@_open_x_final_dim:
3057   }
3058   { \@@_set_final_coords_from_anchor:n { north-east } }
3059   \bool_if:NT \l_@@_parallelize_diags_bool
3060   {
3061     \int_gincr:N \g_@@_iddots_int
3062     \int_compare:nNnTF \g_@@_iddots_int = 1
3063     {
3064       \dim_gset:Nn \g_@@_delta_x_two_dim
3065       { \l_@@_x_final_dim - \l_@@_x_initial_dim }
3066       \dim_gset:Nn \g_@@_delta_y_two_dim
3067       { \l_@@_y_final_dim - \l_@@_y_initial_dim }
3068     }
3069     {
3070       \dim_set:Nn \l_@@_y_final_dim
3071       {
3072         \l_@@_y_initial_dim +
3073         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
3074         \dim_ratio:nn \g_@@_delta_y_two_dim \g_@@_delta_x_two_dim
3075       }
3076     }
3077   }
3078   \@@_draw_line:
3079 }

```

## The actual instructions for drawing the dotted line with Tikz

The command `\@@_draw_line:` should be used in a `{pgfpicture}`. It has six implicit arguments:

- `\l_@@_x_initial_dim`
- `\l_@@_y_initial_dim`
- `\l_@@_x_final_dim`
- `\l_@@_y_final_dim`
- `\l_@@_initial_open_bool`
- `\l_@@_final_open_bool`

```

3080 \cs_new_protected:Npn \@@_draw_line:
3081 {
3082   \pgfrememberpicturepositiononpagetrue
3083   \pgf@relevantforpicturesizefalse
3084   \tl_if_eq:NNTF \l_@@_xdots_line_style_tl \c_@@_standard_tl
3085     \@@_draw_standard_dotted_line:
3086     \@@_draw_non_standard_dotted_line:
3087 }

```

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.

```

3088 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:
3089 {
3090   \begin { scope }
3091   \exp_args:No \@@_draw_non_standard_dotted_line:n
3092     { \l_@@_xdots_line_style_tl , \l_@@_xdots_color_tl }
3093 }

```

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.

```

3094 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:n #1
3095 {
3096   \@@_draw_non_standard_dotted_line:nVV
3097   { #1 }
3098   \l_@@_xdots_up_tl
3099   \l_@@_xdots_down_tl
3100 }
3101 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:nnn #1 #2 #3
3102 {
3103   \draw
3104   [
3105     #1 ,
3106     shorten~> = \l_@@_xdots_shorten_dim ,
3107     shorten~< = \l_@@_xdots_shorten_dim ,
3108   ]
3109   ( \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).

```

3110   -- node [ sloped , above ] { $ \scriptstyle #2 $ }
3111   node [ sloped , below ] { $ \scriptstyle #3 $ }
3112   ( \l_@@_x_final_dim , \l_@@_y_final_dim ) ;
3113 \end { scope }
3114 }
3115 \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).

```

3116 \cs_new_protected:Npn \@@_draw_standard_dotted_line:
3117 {
3118   \bool_lazy_and:nnF
3119     { \tl_if_empty_p:N \l_@@_xdots_up_tl }
3120     { \tl_if_empty_p:N \l_@@_xdots_down_tl }
3121   {
3122     \pgfscope
3123     \pgftransformshift
3124       {
3125         \pgfpointlineattime { 0.5 }
3126         { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
3127         { \pgfpoint \l_@@_x_final_dim \l_@@_y_final_dim }
3128       }
3129     \pgftransformrotate
3130       {
3131         \fp_eval:n
3132           {
3133             atand
3134             (
3135               \l_@@_y_final_dim - \l_@@_y_initial_dim ,
3136               \l_@@_x_final_dim - \l_@@_x_initial_dim
3137             )
3138           }
3139       }
3140     \pgfnode
3141       { rectangle }
3142       { south }
3143       {
3144         \c_math_toggle_token
3145         \scriptstyle \l_@@_xdots_up_tl
3146         \c_math_toggle_token
3147       }
3148       { }
3149       { \pgfusepath { } }
3150     \pgfnode
3151       { rectangle }
3152       { north }
3153       {
3154         \c_math_toggle_token
3155         \scriptstyle \l_@@_xdots_down_tl
3156         \c_math_toggle_token
3157       }
3158       { }
3159       { \pgfusepath { } }
3160     \endpgfscope
3161   }
3162   \group_begin:

```

The dimension `\l_@@_l_dim` is the length  $\ell$  of the line to draw. We use the floating point reals of `expl3` to compute this length.

```

3163   \dim_zero_new:N \l_@@_l_dim
3164   \dim_set:Nn \l_@@_l_dim
3165     {
3166       \fp_to_dim:n
3167         {
3168           sqrt
3169             (
3170               ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) ^ 2
3171               +
3172               ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) ^ 2
3173             )
3174         }

```

3175        }

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.

```

3176        \bool_lazy_or:nnF
3177        { \dim_compare_p:nNn { \dim_abs:n \l_@@_l_dim } > \c_@@_max_l_dim }
3178        { \dim_compare_p:nNn \l_@@_l_dim = \c_zero_dim }
3179        \@@_draw_standard_dotted_line_i:
3180    \group_end:
3181    }

3182 \dim_const:Nn \c_@@_max_l_dim { 50 cm }

3183 \cs_new_protected:Npn \@@_draw_standard_dotted_line_i:
3184    {

```

The number of dots will be `\l_tmpa_int + 1`.

```

3185        \bool_if:NTF \l_@@_initial_open_bool
3186        {
3187        \bool_if:NTF \l_@@_final_open_bool
3188        {
3189        \int_set:Nn \l_tmpa_int
3190        { \dim_ratio:nn \l_@@_l_dim \l_@@_inter_dots_dim }
3191        }
3192        {
3193        \int_set:Nn \l_tmpa_int
3194        {
3195        \dim_ratio:nn
3196        { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
3197        \l_@@_inter_dots_dim
3198        }
3199        }
3200        }
3201        {
3202        \bool_if:NTF \l_@@_final_open_bool
3203        {
3204        \int_set:Nn \l_tmpa_int
3205        {
3206        \dim_ratio:nn
3207        { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
3208        \l_@@_inter_dots_dim
3209        }
3210        }
3211        {
3212        \int_set:Nn \l_tmpa_int
3213        {
3214        \dim_ratio:nn
3215        { \l_@@_l_dim - 2 \l_@@_xdots_shorten_dim }
3216        \l_@@_inter_dots_dim
3217        }
3218        }
3219        }

```

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

```

3220        \dim_set:Nn \l_tmpa_dim
3221        {
3222        ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
3223        \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
3224        }
3225        \dim_set:Nn \l_tmpb_dim
3226        {
3227        ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *

```

```

3228     \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
3229 }

```

The length  $\ell$  is the length of the dotted line. We note  $\Delta$  the length between two dots and  $n$  the number of intervals between dots. We note  $\delta = \frac{1}{2}(\ell - n\Delta)$ . The distance between the initial extremity of the line and the first dot will be equal to  $k \cdot \delta$  where  $k = 0, 1$  or  $2$ . We first compute this number  $k$  in `\l_tmpb_int`.

```

3230 \int_set:Nn \l_tmpb_int
3231 {
3232   \bool_if:NTF \l_@@_initial_open_bool
3233     { \bool_if:NTF \l_@@_final_open_bool 1 0 }
3234     { \bool_if:NTF \l_@@_final_open_bool 2 1 }
3235 }

```

In the loop over the dots, the dimensions `\l_@@_x_initial_dim` and `\l_@@_y_initial_dim` will be used for the coordinates of the dots. But, before the loop, we must move until the first dot.

```

3236 \dim_gadd:Nn \l_@@_x_initial_dim
3237 {
3238   ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
3239   \dim_ratio:nn
3240   { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
3241   { 2 \l_@@_l_dim }
3242   * \l_tmpb_int
3243 }
3244 \dim_gadd:Nn \l_@@_y_initial_dim
3245 {
3246   ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
3247   \dim_ratio:nn
3248   { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
3249   { 2 \l_@@_l_dim }
3250   * \l_tmpb_int
3251 }
3252 \pgf@relevantforpicturesizefalse
3253 \int_step_inline:nnn 0 \l_tmpa_int
3254 {
3255   \pgfpathcircle
3256   { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
3257   { \l_@@_radius_dim }
3258   \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
3259   \dim_add:Nn \l_@@_y_initial_dim \l_tmpb_dim
3260 }
3261 \pgfusepathqfill
3262 }

```

## User commands available in the new environments

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

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

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

```

3263 \AtBeginDocument
3264 {

```

```

3265 \tl_set:Nn \l_@@_argspec_tl { 0 { } E { _ ^ } { { } { } } }
3266 \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
3267 \exp_args:NNV \NewDocumentCommand \@@_Ldots \l_@@_argspec_tl
3268 {
3269   \int_compare:nNnTF \c@jCol = 0
3270   { \@@_error:nn { in~first~col } \Ldots }
3271   {
3272     \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
3273     { \@@_error:nn { in~last~col } \Ldots }
3274     {
3275       \@@_instruction_of_type:nnn \c_false_bool { Ldots }
3276       { #1 , down = #2 , up = #3 }
3277     }
3278   }
3279   \bool_if:NF \l_@@_nullify_dots_bool
3280   { \phantom { \ensuremath { \@@_old_ldots } } }
3281   \bool_gset_true:N \g_@@_empty_cell_bool
3282 }

3283 \exp_args:NNV \NewDocumentCommand \@@_Cdots \l_@@_argspec_tl
3284 {
3285   \int_compare:nNnTF \c@jCol = 0
3286   { \@@_error:nn { in~first~col } \Cdots }
3287   {
3288     \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
3289     { \@@_error:nn { in~last~col } \Cdots }
3290     {
3291       \@@_instruction_of_type:nnn \c_false_bool { Cdots }
3292       { #1 , down = #2 , up = #3 }
3293     }
3294   }
3295   \bool_if:NF \l_@@_nullify_dots_bool
3296   { \phantom { \ensuremath { \@@_old_cdots } } }
3297   \bool_gset_true:N \g_@@_empty_cell_bool
3298 }

3299 \exp_args:NNV \NewDocumentCommand \@@_Vdots \l_@@_argspec_tl
3300 {
3301   \int_compare:nNnTF \c@iRow = 0
3302   { \@@_error:nn { in~first~row } \Vdots }
3303   {
3304     \int_compare:nNnTF \c@iRow = \l_@@_last_row_int
3305     { \@@_error:nn { in~last~row } \Vdots }
3306     {
3307       \@@_instruction_of_type:nnn \c_false_bool { Vdots }
3308       { #1 , down = #2 , up = #3 }
3309     }
3310   }
3311   \bool_if:NF \l_@@_nullify_dots_bool
3312   { \phantom { \ensuremath { \@@_old_vdots } } }
3313   \bool_gset_true:N \g_@@_empty_cell_bool
3314 }

3315 \exp_args:NNV \NewDocumentCommand \@@_Ddots \l_@@_argspec_tl
3316 {
3317   \int_case:nnF \c@iRow
3318   {
3319     0 { \@@_error:nn { in~first~row } \Ddots }
3320     \l_@@_last_row_int { \@@_error:nn { in~last~row } \Ddots }
3321   }
3322   {

```



```

3323 \int_case:nnF \c@jCol
3324 {
3325     0 { \@@_error:nn { in~first~col } \Ddots }
3326     \l_@@_last_col_int { \@@_error:nn { in~last~col } \Ddots }
3327 }
3328 {
3329     \keys_set_known:nn { NiceMatrix / Ddots } { #1 }
3330     \@@_instruction_of_type:nnn \l_@@_draw_first_bool { Ddots }
3331     { #1 , down = #2 , up = #3 }
3332 }
3333
3334 }
3335 \bool_if:NF \l_@@_nullify_dots_bool
3336 { \phantom { \ensuremath { \@@_old_ddots } } }
3337 \bool_gset_true:N \g_@@_empty_cell_bool
3338 }

3339 \exp_args:NNV \NewDocumentCommand \@@_Iddots \l_@@_argspec_tl
3340 {
3341     \int_case:nnF \c@iRow
3342     {
3343         0 { \@@_error:nn { in~first~row } \Iddots }
3344         \l_@@_last_row_int { \@@_error:nn { in~last~row } \Iddots }
3345     }
3346     {
3347         \int_case:nnF \c@jCol
3348         {
3349             0 { \@@_error:nn { in~first~col } \Iddots }
3350             \l_@@_last_col_int { \@@_error:nn { in~last~col } \Iddots }
3351         }
3352         {
3353             \keys_set_known:nn { NiceMatrix / Ddots } { #1 }
3354             \@@_instruction_of_type:nnn \l_@@_draw_first_bool { Iddots }
3355             { #1 , down = #2 , up = #3 }
3356         }
3357     }
3358     \bool_if:NF \l_@@_nullify_dots_bool
3359     { \phantom { \ensuremath { \@@_old_iddots } } }
3360     \bool_gset_true:N \g_@@_empty_cell_bool
3361 }
3362 }

```

End of the \AtBeginDocument.

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

```

3363 \keys_define:nn { NiceMatrix / Ddots }
3364 {
3365     draw-first .bool_set:N = \l_@@_draw_first_bool ,
3366     draw-first .default:n = true ,
3367     draw-first .value_forbidden:n = true
3368 }

```

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

```

3369 \cs_new_protected:Npn \@@_Hspace:
3370 {
3371     \bool_gset_true:N \g_@@_empty_cell_bool
3372     \hspace
3373 }

```

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

```

3374 \cs_set_eq:NN \@@_old_multicolumn \multicolumn
3375 \cs_new:Npn \@@_multicolumn:nnn #1 #2 #3
3376 {

```

We have to act in an expandable way since it will begin by a `\multicolumn`.

```

3377 \exp_args:NNe
3378 \@@_old_multicolumn
3379 { #1 }
3380 {
3381 \exp_args:Ne \str_case:nn { \str_foldcase:n { #2 } }
3382 {
3383 1 { > \@@_Cell: 1 < \@@_end_Cell: }
3384 r { > \@@_Cell: r < \@@_end_Cell: }
3385 c { > \@@_Cell: c < \@@_end_Cell: }
3386 { 1 | } { > \@@_Cell: 1 < \@@_end_Cell: | }
3387 { r | } { > \@@_Cell: r < \@@_end_Cell: | }
3388 { c | } { > \@@_Cell: c < \@@_end_Cell: | }
3389 { | 1 } { | > \@@_Cell: 1 < \@@_end_Cell: }
3390 { | r } { | > \@@_Cell: r < \@@_end_Cell: }
3391 { | c } { | > \@@_Cell: c < \@@_end_Cell: }
3392 { | 1 | } { | > \@@_Cell: 1 < \@@_end_Cell: | }
3393 { | r | } { | > \@@_Cell: r < \@@_end_Cell: | }
3394 { | c | } { | > \@@_Cell: c < \@@_end_Cell: | }
3395 }
3396 }
3397 { #3 }

```

The `\peek_remove_spaces:n` is mandatory.

```

3398 \peek_remove_spaces:n
3399 {
3400 \int_compare:nNnT #1 > 1
3401 {
3402 \seq_gput_left:Nx \g_@@_multicolumn_cells_seq
3403 { \int_use:N \c@iRow - \int_use:N \c@jCol }
3404 \seq_gput_left:Nn \g_@@_multicolumn_sizes_seq { #1 }
3405 \seq_gput_right:Nx \g_@@_pos_of_blocks_seq
3406 {
3407 { \int_use:N \c@iRow }
3408 { \int_use:N \c@jCol }
3409 { \int_use:N \c@iRow }
3410 { \int_eval:n { \c@jCol + #1 - 1 } }
3411 }
3412 }
3413 \int_gadd:Nn \c@jCol { #1 - 1 }
3414 \int_compare:nNnT \c@jCol > \g_@@_col_total_int
3415 { \int_gset_eq:NN \g_@@_col_total_int \c@jCol }
3416 }
3417 }

```

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`.

```

3418 \cs_new:Npn \@@_Hdotsfor:
3419 {
3420 \bool_lazy_and:nnTF
3421 { \int_compare_p:nNn \c@jCol = 0 }
3422 { \int_compare_p:nNn \l_@@_first_col_int = 0 }
3423 {
3424 \bool_if:NTF \g_@@_after_col_zero_bool
3425 {
3426 \multicolumn { 1 } { c } { }
3427 \@@_Hdotsfor_i
3428 }

```

```

3429     { \@@_fatal:n { Hdotsfor~in~col~0 } }
3430   }
3431   {
3432     \multicolumn { 1 } { c } { }
3433     \@@_Hdotsfor_i
3434   }
3435 }

```

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:`).

```

3436 \AtBeginDocument
3437 {
3438   \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
3439   \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.

```

3440   \exp_args:NNV \NewDocumentCommand \@@_Hdotsfor_i \l_@@_argspec_tl
3441   {
3442     \tl_gput_right:Nx \g_@@_HVdotsfor_lines_tl
3443     {
3444       \@@_Hdotsfor:nnnn
3445       { \int_use:N \c@iRow }
3446       { \int_use:N \c@jCol }
3447       { #2 }
3448       {
3449         #1 , #3 ,
3450         down = \exp_not:n { #4 } , up = \exp_not:n { #5 }
3451       }
3452     }
3453     \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { c } { } }
3454   }
3455 }

```

Enf of `\AtBeginDocument`.

```

3456 \cs_new_protected:Npn \@@_Hdotsfor:nnnn #1 #2 #3 #4
3457 {
3458   \bool_set_false:N \l_@@_initial_open_bool
3459   \bool_set_false:N \l_@@_final_open_bool

```

For the row, it's easy.

```

3460   \int_set:Nn \l_@@_initial_i_int { #1 }
3461   \int_set_eq:NN \l_@@_final_i_int \l_@@_initial_i_int

```

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

```

3462   \int_compare:nNnTF #2 = 1
3463   {
3464     \int_set:Nn \l_@@_initial_j_int 1
3465     \bool_set_true:N \l_@@_initial_open_bool
3466   }
3467   {
3468     \cs_if_exist:cTF
3469     {
3470       pgf @ sh @ ns @ \@@_env:
3471       - \int_use:N \l_@@_initial_i_int
3472       - \int_eval:n { #2 - 1 }
3473     }
3474     { \int_set:Nn \l_@@_initial_j_int { #2 - 1 } }
3475     {
3476       \int_set:Nn \l_@@_initial_j_int { #2 }
3477       \bool_set_true:N \l_@@_initial_open_bool
3478     }
3479   }

```

```

3480 \int_compare:nNnTF { #2 + #3 - 1 } = \c@jCol
3481 {
3482   \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
3483   \bool_set_true:N \l_@@_final_open_bool
3484 }
3485 {
3486   \cs_if_exist:cTF
3487   {
3488     pgf @ sh @ ns @ \@@_env:
3489     - \int_use:N \l_@@_final_i_int
3490     - \int_eval:n { #2 + #3 }
3491   }
3492   { \int_set:Nn \l_@@_final_j_int { #2 + #3 } }
3493   {
3494     \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
3495     \bool_set_true:N \l_@@_final_open_bool
3496   }
3497 }
3498 \group_begin:
3499 \int_compare:nNnTF { #1 } = 0
3500 { \color { nicematrix-first-row } }
3501 {
3502   \int_compare:nNnT { #1 } = \g_@@_row_total_int
3503   { \color { nicematrix-last-row } }
3504 }
3505 \keys_set:nn { NiceMatrix / xdots } { #4 }
3506 \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3507 \@@_actually_draw_Ldots:
3508 \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).

```

3509 \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
3510 { \cs_set:cpn { @@ _ dotted _ #1 - ##1 } { } }
3511 }

3512 \AtBeginDocument
3513 {
3514   \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
3515   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
3516   \exp_args:NNV \NewDocumentCommand \@@_Vdotsfor: \l_@@_argspec_tl
3517   {
3518     \tl_gput_right:Nx \g_@@_HVdotsfor_lines_tl
3519     {
3520       \@@_Vdotsfor:nnnn
3521       { \int_use:N \c@iRow }
3522       { \int_use:N \c@jCol }
3523       { #2 }
3524       {
3525         #1 , #3 ,
3526         down = \exp_not:n { #4 } , up = \exp_not:n { #5 }
3527       }
3528     }
3529   }
3530 }

```

Enf of `\AtBeginDocument`.

```

3531 \cs_new_protected:Npn \@@_Vdotsfor:nnnn #1 #2 #3 #4
3532 {
3533   \bool_set_false:N \l_@@_initial_open_bool
3534   \bool_set_false:N \l_@@_final_open_bool

```

For the column, it's easy.

```
3535 \int_set:Nn \l_@@_initial_j_int { #2 }
3536 \int_set_eq:NN \l_@@_final_j_int \l_@@_initial_j_int
```

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

```
3537 \int_compare:nNnTF #1 = 1
3538 {
3539   \int_set:Nn \l_@@_initial_i_int 1
3540   \bool_set_true:N \l_@@_initial_open_bool
3541 }
3542 {
3543   \cs_if_exist:cTF
3544   {
3545     pgf @ sh @ ns @ \@@_env:
3546     - \int_eval:n { #1 - 1 }
3547     - \int_use:N \l_@@_initial_j_int
3548   }
3549   { \int_set:Nn \l_@@_initial_i_int { #1 - 1 } }
3550   {
3551     \int_set:Nn \l_@@_initial_i_int { #1 }
3552     \bool_set_true:N \l_@@_initial_open_bool
3553   }
3554 }
3555 \int_compare:nNnTF { #1 + #3 - 1 } = \c@iRow
3556 {
3557   \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }
3558   \bool_set_true:N \l_@@_final_open_bool
3559 }
3560 {
3561   \cs_if_exist:cTF
3562   {
3563     pgf @ sh @ ns @ \@@_env:
3564     - \int_eval:n { #1 + #3 }
3565     - \int_use:N \l_@@_final_j_int
3566   }
3567   { \int_set:Nn \l_@@_final_i_int { #1 + #3 } }
3568   {
3569     \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }
3570     \bool_set_true:N \l_@@_final_open_bool
3571   }
3572 }
3573 \group_begin:
3574 \int_compare:nNnTF { #2 } = 0
3575 { \color { nicematrix-first-col } }
3576 {
3577   \int_compare:nNnT { #2 } = \g_@@_col_total_int
3578   { \color { nicematrix-last-col } }
3579 }
3580 \keys_set:nn { NiceMatrix / xdots } { #4 }
3581 \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3582 \@@_actually_draw_Vdots:
3583 \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).

```
3584 \int_step_inline:nnn { #1 } { #1 + #3 - 1 }
3585 { \cs_set:cpn { @@ _ dotted _ ##1 - #2 } { } }
3586 }
```

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

```
3587 \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>58</sup>

```
3588 \cs_new:Npn \@@_double_int_eval:n #1-#2 \q_stop
3589 { \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).

```
3590 \AtBeginDocument
3591 {
3592   \tl_set:Nn \l_@@_argspec_tl { 0 { } m m ! 0 { } E { _ ^ } { { } { } } }
3593   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
3594   \exp_args:NNV \NewDocumentCommand \@@_line \l_@@_argspec_tl
3595   {
3596     \group_begin:
3597     \keys_set:nn { NiceMatrix / xdots } { #1 , #4 , down = #5 , up = #6 }
3598     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3599     \use:e
3600     {
3601       \@@_line_i:nn
3602       { \@@_double_int_eval:n #2 \q_stop }
3603       { \@@_double_int_eval:n #3 \q_stop }
3604     }
3605     \group_end:
3606   }
3607 }

3608 \cs_new_protected:Npn \@@_line_i:nn #1 #2
3609 {
3610   \bool_set_false:N \l_@@_initial_open_bool
3611   \bool_set_false:N \l_@@_final_open_bool
3612   \bool_if:nTF
3613   {
3614     \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #1 }
3615     ||
3616     \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #2 }
3617   }
3618   {
3619     \@@_error:nnn { unknown~cell~for~line~in~CodeAfter } { #1 } { #2 }
3620   }
3621   { \@@_draw_line_ii:nn { #1 } { #2 } }
3622 }

3623 \AtBeginDocument
3624 {
3625   \cs_new_protected:Npx \@@_draw_line_ii:nn #1 #2
3626   {
```

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>58</sup>Indeed, we want that the user may use the command `\line` in `\CodeAfter` with LaTeX counters in the arguments — with the command `\value`.

```

3627     \c_@@_pgfortikzpicture_tl
3628     \@@_draw_line_iii:nn { #1 } { #2 }
3629     \c_@@_endpgfortikzpicture_tl
3630   }
3631 }

```

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

```

3632 \cs_new_protected:Npn \@@_draw_line_iii:nn #1 #2
3633 {
3634   \pgfrememberpicturepositiononpagetrue
3635   \pgfpointshapeborder { \@@_env: - #1 } { \@@_qpoint:n { #2 } }
3636   \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
3637   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
3638   \pgfpointshapeborder { \@@_env: - #2 } { \@@_qpoint:n { #1 } }
3639   \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
3640   \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
3641   \@@_draw_line:
3642 }

```

The commands `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, and `\Iddots` don't use this command because they have to do other settings (for example, the diagonal lines must be parallelized).

## Colors of cells, rows and columns

We want to avoid the thin white lines that are shown in some PDF viewers (eg: with the engine MuPDF used by SumatraPDF). That's why we try to draw rectangles of the same color in the same instruction `\pgfusepath { fill }` (and they will be in the same instruction `fill`—coded `f`— in the resulting PDF).

The commands `\@@_rowcolor`, `\@@_columncolor` and `\@@_rectanglecolor` (which are linked to `\rowcolor`, `\columncolor` and `\rectanglecolor` before the execution of the `code-before`) don't directly draw the corresponding rectangles. Instead, they store their instructions color by color:

- A sequence `\g_@@_colors_seq` will be built containing all the colors used by at least one of these instructions. Each *color* may be prefixed by its color model (eg: `[gray]{0.5}`).
- For the color whose index in `\g_@@_colors_seq` is equal to *i*, a list of instructions which use that color will be constructed in the token list `\g_@@_color_i_tl`. In that token list, the instructions will be written using `\@@_rowcolor:n`, `\@@_columncolor:n` and `\@@_rectanglecolor:nn` (corresponding of `\@@_rowcolor`, `\@@_columncolor` and `\@@_rectanglecolor`).

`bigskip #1` is the color and `#2` is an instruction using that color. Despite its name, the command `\@@_add_to_color_seq` doesn't only add a color to `\g_@@_colors_seq`: it also updates the corresponding token list `\g_@@_color_i_tl`. We add in a global way because the final user may use the instructions such as `\cellcolor` in a loop of `pgffor` in the `\CodeBefore` (and we recall that a loop of `pgffor` is encapsulated in a group).

```

3643 \cs_new_protected:Npn \@@_add_to_colors_seq:nn #1 #2
3644 {

```

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.

```

3645   \int_zero:N \l_tmpa_int
3646   \seq_map_indexed_inline:Nn \g_@@_colors_seq
3647     { \tl_if_eq:nnT { #1 } { ##2 } { \int_set:Nn \l_tmpa_int { ##2 } } }
3648   \int_compare:nNnTF \l_tmpa_int = \c_zero_int

```

First, the case where the color is a *new* color (not in the sequence).

```

3649     {
3650       \seq_gput_right:Nn \g_@@_colors_seq { #1 }
3651       \tl_gset:cx { g_@@_color _ \seq_count:N \g_@@_colors_seq _ tl } { #2 }
3652     }

```

Now, the case where the color is *not* a new color (the color is in the sequence at the position `\l_tmpa_int`).

```

3653     { \tl_gput_right:cx { g_@@_color _ \int_use:N \l_tmpa_int _tl } { #2 } }
3654   }
3655 \cs_generate_variant:Nn \@@_add_to_colors_seq:nn { x n }

```

The macro `\@@_actually_color:` will actually fill all the rectangles, color by color (using the sequence `\l_@@_colors_seq` and all the token lists of the form `\l_@@_color_i_tl`).

```

3656 \cs_new_protected:Npn \@@_actually_color:
3657 {
3658   \pgfpicture
3659   \pgf@relevantforpicturesizefalse
3660   \seq_map_indexed_inline:Nn \g_@@_colors_seq
3661   {
3662     \color ##2
3663     \use:c { g_@@_color _ ##1 _tl }
3664     \tl_gclear:c { g_@@_color _ ##1 _tl }
3665     \pgfusepath { fill }
3666   }
3667   \endpgfpicture
3668 }
3669 \cs_set_protected:Npn \@@_cut_on_hyphen:w #1-#2\q_stop
3670 {
3671   \tl_set:Nn \l_tmpa_tl { #1 }
3672   \tl_set:Nn \l_tmpb_tl { #2 }
3673 }

```

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

```

3674 \NewDocumentCommand \@@_rowcolor { 0 { } m m }
3675 {
3676   \tl_if_blank:nF { #2 }
3677   {
3678     \@@_add_to_colors_seq:xn
3679     { \tl_if_blank:nF { #1 } { [ #1 ] } { #2 } }
3680     { \@@_rowcolor:n { #3 } }
3681   }
3682 }
3683 \cs_new_protected:Npn \@@_rowcolor:n #1
3684 {
3685   \tl_set:Nn \l_@@_rows_tl { #1 }
3686   \tl_set:Nn \l_@@_cols_tl { - }

```

The command `\@@_cartesian_path:` takes in two implicit arguments: `\l_@@_cols_tl` and `\l_@@_rows_tl`.

```

3687   \@@_cartesian_path:
3688 }

```

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

```

3689 \NewDocumentCommand \@@_columncolor { 0 { } m m }
3690 {
3691   \tl_if_blank:nF { #2 }
3692   {
3693     \@@_add_to_colors_seq:xn
3694     { \tl_if_blank:nF { #1 } { [ #1 ] } { #2 } }
3695     { \@@_columncolor:n { #3 } }
3696   }
3697 }

```



```

3698 \cs_new_protected:Npn \@@_columncolor:n #1
3699 {
3700   \tl_set:Nn \l_@@_rows_tl { - }
3701   \tl_set:Nn \l_@@_cols_tl { #1 }

```

The command `\@@_cartesian_path:` takes in two implicit arguments: `\l_@@_cols_tl` and `\l_@@_rows_tl`.

```

3702   \@@_cartesian_path:
3703 }

```

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

```

3704 \NewDocumentCommand \@@_rectanglecolor { 0 { } m m m }
3705 {
3706   \tl_if_blank:nF { #2 }
3707   {
3708     \@@_add_to_colors_seq:xn
3709     { \tl_if_blank:nF { #1 } { [ #1 ] } { #2 } }
3710     { \@@_rectanglecolor:nnn { #3 } { #4 } { 0 pt } }
3711   }
3712 }

```

The last argument is the radius of the corners of the rectangle.

```

3713 \NewDocumentCommand \@@_roundedrectanglecolor { 0 { } m m m m }
3714 {
3715   \tl_if_blank:nF { #2 }
3716   {
3717     \@@_add_to_colors_seq:xn
3718     { \tl_if_blank:nF { #1 } { [ #1 ] } { #2 } }
3719     { \@@_rectanglecolor:nnn { #3 } { #4 } { #5 } }
3720   }
3721 }

```

The last argument is the radius of the corners of the rectangle.

```

3722 \cs_new_protected:Npn \@@_rectanglecolor:nnn #1 #2 #3
3723 {
3724   \@@_cut_on_hyphen:w #1 \q_stop
3725   \tl_clear_new:N \l_tmpc_tl
3726   \tl_clear_new:N \l_tmpd_tl
3727   \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl
3728   \tl_set_eq:NN \l_tmpd_tl \l_tmpb_tl
3729   \@@_cut_on_hyphen:w #2 \q_stop
3730   \tl_set:Nx \l_@@_rows_tl { \l_tmpc_tl - \l_tmpa_tl }
3731   \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`.

```

3732   \@@_cartesian_path:n { #3 }
3733 }

```

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

```

3734 \NewDocumentCommand \@@_cellcolor { 0 { } m m }
3735 {
3736   \clist_map_inline:nn { #3 }
3737   { \@@_rectanglecolor [ #1 ] { #2 } { ##1 } { ##1 } }
3738 }

```

```

3739 \NewDocumentCommand \@@_chessboardcolors { 0 { } m m }
3740 {
3741   \int_step_inline:nn { \int_use:N \c@iRow }
3742   {
3743     \int_step_inline:nn { \int_use:N \c@jCol }

```

```

3744     {
3745         \int_if_even:nTF { ####1 + ##1 }
3746         { \@@_cellcolor [ #1 ] { #2 } }
3747         { \@@_cellcolor [ #1 ] { #3 } }
3748     { ##1 - ####1 }
3749     }
3750 }
3751 }

3752 \keys_define:nn { NiceMatrix / arraycolor }
3753 {
3754     except-corners .clist_set:N = \l_tmpa_clist ,
3755     except-corners .default:n = { NW , SW , NE , SE }
3756 }

```

The command `\@@_arraycolor` (linked to `\arraycolor` at the beginning of the `\CodeBefore`) will color the whole tabular (excepted the potential exterior rows and columns). The third argument is a optional argument which a list of pairs key-value. As for now, there is only one key: `except-corners`. When that key is used, the cells in the corners are not colored.

```

3757 \NewDocumentCommand \@@_arraycolor { 0 { } m 0 { } }
3758 {
3759     \keys_set:nn { NiceMatrix / arraycolor } { #3 }

```

If the key `except-corners` is not used, it's easy: we only have a rectangle fo fill.

```

3760     \clist_if_empty:NTF \l_tmpa_clist
3761     {
3762         \@@_rectanglecolor [ #1 ] { #2 }
3763         { 1 - 1 } { \int_use:N \c@iRow - \int_use:N \c@jCol }
3764     }

```

The interesting case is when the key `except-corners` is used. In that case, we can't fill now the tabular because we don't know the list of the cells of the corners. That's why we postpone the treatment in the `\CodeAfter`.

```

3765     {
3766         \tl_gput_left:Nx \g_@@_internal_code_after_tl
3767         {
3768             \@@_arraycolor_code_after:nnn
3769             { #1 }
3770             { \exp_not:n { #2 } }
3771             { \l_tmpa_clist }
3772         }
3773     }
3774 }

```

The following command will be used in the `\CodeAfter`. #1 is the color model, #2 the color and #3 the value of the key `except-corners` (a sublist of {NW,NE,SW,SE}).

```

3775 \cs_new_protected:Npn \@@_arraycolor_code_after:nnn #1 #2 #3
3776 {
3777     \group_begin:

```

First, we compute the corners.

```

3778     \clist_set:Nn \l_@@_corners_clist { #3 }
3779     \@@_compute_corners:

```

Now, for each cell of the array (excepted the potential exterior rows and columns) we check whether we have to fill that cell. When we have to fill the cell, we do the job by writing an instruction in the `\CodeBefore` (in fact, the "`\CodeBefore`" which will be written on the `aux` file for the next run).

```

3780     \int_step_inline:nn \c@iRow
3781     {
3782         \int_step_inline:nn \c@jCol
3783         {
3784             \seq_if_in:NnF \l_@@_empty_corner_cells_seq { ##1 - ####1 }

```

```

3785         {
3786         \tl_gput_left:Nx \g_nicematrix_code_before_tl
3787         { \@@_cellcolor [ #1 ] { \exp_not:n { #2 } } { ##1 - #####1 } }
3788         }
3789     }
3790 }
3791 \group_end:
3792 }

3793 \keys_define:nn { NiceMatrix / rowcolors }
3794 {
3795     respect-blocks .bool_set:N = \l_@@_respect_blocks_bool ,
3796     respect-blocks .default:n = true ,
3797     cols .tl_set:N = \l_@@_cols_tl ,
3798     restart .bool_set:N = \l_@@_rowcolors_restart_bool ,
3799     restart .default:n = true ,
3800     unknown .code:n = \@@_error:n { Unknown-key-for-rowcolors }
3801 }

```

The command `\rowcolors` (accessible in the code-before) is inspired by the command `\rowcolors` of the package `xcolor` (with the option `table`). However, the command `\rowcolors` of `nicematrix` has *not* the optional argument of the command `\rowcolors` of `xcolor`. Here is an example: `\rowcolors{1}{blue!10}{}[respect-blocks]`.

#1 (optional) is the color space ; #2 is a list of intervals of rows ; #3 is the first color ; #4 is the second color ; #5 is for the optional list of pairs key-value.

```

3802 \NewDocumentCommand \@@_rowcolors { 0 { } m m m 0 { } }
3803 {

```

The group is for the options.

```

3804     \group_begin:
3805     \tl_clear_new:N \l_@@_cols_tl
3806     \tl_set:Nn \l_@@_cols_tl { - }
3807     \keys_set:nn { NiceMatrix / rowcolors } { #5 }

```

The boolean `\l_tmpa_bool` will indicate whereas we are in a row of the first color or of the second color.

```

3808     \bool_set_true:N \l_tmpa_bool
3809     \bool_lazy_and:nnT
3810         \l_@@_respect_blocks_bool
3811         {
3812             \cs_if_exist_p:c
3813                 { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq } }
3814     }

```

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`).

```

3815     \seq_set_eq:Nc \l_tmpb_seq
3816         { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq }
3817     \seq_set_filter:NNn \l_tmpa_seq \l_tmpb_seq
3818         { \@@_not_in_exterior_p:nnnn ##1 }
3819 }
3820 \pgfpicture
3821 \pgf@relevantforpicturesizefalse
3822 \clist_map_inline:nn { #2 }
3823 {
3824     \tl_set:Nn \l_tmpa_tl { ##1 }
3825     \tl_if_in:NnTF \l_tmpa_tl { - }
3826     { \@@_cut_on_hyphen:w ##1 \q_stop }
3827     { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@iRow } }

```

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

```

3828     \int_set:Nn \l_tmpa_int \l_tmpa_tl
3829     \bool_if:NTF \l_@@_rowcolors_restart_bool
3830     { \bool_set_true:N \l_tmpa_bool }
3831     { \bool_set:Nn \l_tmpa_bool { \int_if_odd_p:n { \l_tmpa_tl } } }
3832     \int_zero_new:N \l_tmpc_int
3833     \int_set:Nn \l_tmpc_int \l_tmpb_tl
3834     \int_do_until:nNnn \l_tmpa_int > \l_tmpc_int
3835     {

```

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

```

3836         \int_set_eq:NN \l_tmpb_int \l_tmpa_int

```

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

```

3837         \bool_lazy_and:nnT
3838         \l_@@_respect_blocks_bool
3839         {
3840             \cs_if_exist_p:c
3841             { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq }
3842         }
3843         {
3844             \seq_set_filter:NNn \l_tmpb_seq \l_tmpa_seq
3845             { \@@_intersect_our_row_p:nnnn ###1 }
3846             \seq_map_inline:Nn \l_tmpb_seq { \@@_rowcolors_i:nnnn ###1 }

```

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

```

3847         }
3848         \tl_set:Nx \l_@@_rows_tl
3849         { \int_use:N \l_tmpa_int - \int_use:N \l_tmpb_int }
3850         \bool_if:NTF \l_tmpa_bool
3851         {
3852             \tl_if_blank:nF { #3 }
3853             {
3854                 \tl_if_empty:nTF { #1 }
3855                 \color
3856                 { \color [ #1 ] }
3857                 { #3 }

```

The command `\@@_cartesian_path:` takes in two implicit arguments: `\l_@@_cols_tl` and `\l_@@_rows_tl`.

```

3858             \@@_cartesian_path:
3859             \pgfusepath { fill }
3860         }
3861         \bool_set_false:N \l_tmpa_bool
3862     }
3863     {
3864         \tl_if_blank:nF { #4 }
3865         {
3866             \tl_if_empty:nTF { #1 }
3867             \color
3868             { \color [ #1 ] }
3869             { #4 }

```

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

```

3870             \@@_cartesian_path:
3871             \pgfusepath { fill }
3872         }
3873         \bool_set_true:N \l_tmpa_bool
3874     }
3875     \int_set:Nn \l_tmpa_int { \l_tmpb_int + 1 }
3876 }
3877 }
3878 \endpgfpicture
3879 \group_end:
3880 }

```

```

3881 \cs_new_protected:Npn \@@_rowcolors_i:nnnn #1 #2 #3 #4
3882 {
3883   \int_compare:nNt { #3 } > \l_tmpb_int
3884   { \int_set:Nn \l_tmpb_int { #3 } }
3885 }

3886 \prg_new_conditional:Nnn \@@_not_in_exterior:nnnn p
3887 {
3888   \bool_lazy_or:nnTF
3889   { \int_compare_p:nNn { #4 } = \c_zero_int }
3890   { \int_compare_p:nNn { #2 } = { \@@_succ:n { \c@jCol } } }
3891   \prg_return_false:
3892   \prg_return_true:
3893 }

```

The following command return true when the block intersects the row \l\_tmpa\_int.

```

3894 \prg_new_conditional:Nnn \@@_intersect_our_row:nnnn p
3895 {
3896   \bool_if:nTF
3897   {
3898     \int_compare_p:n { #1 <= \l_tmpa_int }
3899     &&
3900     \int_compare_p:n { \l_tmpa_int <= #3 }
3901   }
3902   \prg_return_true:
3903   \prg_return_false:
3904 }

3905 \cs_new:Npn \@@_cartesian_path: { \@@_cartesian_path:n { 0 pt } }

```

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.

```

3906 \cs_new_protected:Npn \@@_cartesian_path:n #1
3907 {

```

We begin the loop over the columns.

```

3908   \clist_map_inline:Nn \l_@@_cols_tl
3909   {
3910     \tl_set:Nn \l_tmpa_tl { ##1 }
3911     \tl_if_in:NnTF \l_tmpa_tl { - }
3912     { \@@_cut_on_hyphen:w ##1 \q_stop }
3913     { \@@_cut_on_hyphen:w ##1 - ##1 \q_stop }
3914     \bool_lazy_or:nnT
3915     { \tl_if_blank_p:V \l_tmpa_tl }
3916     { \str_if_eq_p:Vn \l_tmpa_tl { * } }
3917     { \tl_set:Nn \l_tmpa_tl { 1 } }
3918     \bool_lazy_or:nnT
3919     { \tl_if_blank_p:V \l_tmpb_tl }
3920     { \str_if_eq_p:Vn \l_tmpb_tl { * } }
3921     { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
3922     \int_compare:nNt \l_tmpb_tl > \c@jCol
3923     { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }

```

If we decide to provide the commands \cellcolor, \rectanglecolor, \rowcolor, \columncolor, \rowcolors and \chessboardcolors available 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 rows below in the same command \@@\_cartesian\_path:.

```

3924     \@@_qpoint:n { col - \l_tmpa_tl }
3925     \int_compare:nNtTF \l_@@_first_col_int = \l_tmpa_tl
3926     { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
3927     { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
3928     \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
3929     \dim_set:Nn \l_tmpa_dim { \pgf@x + 0.5 \arrayrulewidth }

```

We begin the loop over the rows.

```

3930 \clist_map_inline:Nn \l_@@_rows_tl
3931 {
3932   \tl_set:Nn \l_tmpa_tl { #####1 }
3933   \tl_if_in:NnTF \l_tmpa_tl { - }
3934     { \@@_cut_on_hyphen:w #####1 \q_stop }
3935     { \@@_cut_on_hyphen:w #####1 - #####1 \q_stop }
3936   \tl_if_empty:NT \l_tmpa_tl { \tl_set:Nn \l_tmpa_tl { 1 } }
3937   \tl_if_empty:NT \l_tmpb_tl
3938     { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@iRow } }
3939   \int_compare:nNnT \l_tmpb_tl > \c@iRow
3940     { \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`.

```

3941   \@@_qpoint:n { row - \@@_succ:n \l_tmpb_tl }
3942   \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
3943   \@@_qpoint:n { row - \l_tmpa_tl }
3944   \dim_set:Nn \l_tmpd_dim { \pgf@y + 0.5 \arrayrulewidth }
3945   \pgfsetcornersarced { \pgfpoint { #1 } { #1 } }
3946   \pgfpathrectanglecorners
3947     { \pgfpoint \l_tmpc_dim \l_tmpd_dim }
3948     { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3949   }
3950 }
3951 }

```

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

```

3952 \NewDocumentCommand \@@_cellcolor_tabular { 0 { } m }
3953 {
3954   \tl_gput_right:Nx \g_nicematrix_code_before_tl
3955   {

```

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`).

```

3956     \cellcolor [ #1 ] { \exp_not:n { #2 } }
3957     { \int_use:N \c@iRow - \int_use:N \c@jCol }
3958   }
3959 }

```

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

```

3960 \NewDocumentCommand \@@_rowcolor_tabular { 0 { } m }
3961 {
3962   \tl_gput_right:Nx \g_nicematrix_code_before_tl
3963   {
3964     \exp_not:N \rectanglecolor [ #1 ] { \exp_not:n { #2 } }
3965     { \int_use:N \c@iRow - \int_use:N \c@jCol }
3966     { \int_use:N \c@iRow - \exp_not:n { \int_use:N \c@jCol } }
3967   }
3968 }

```

```

3969 \NewDocumentCommand \@@_columncolor_preamble { 0 { } m }
3970 {

```

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).

```

3971   \int_compare:nNnT \c@jCol > \g_@@_col_total_int
3972   {

```

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).

```

3973      \tl_gput_left:Nx \g_nicematrix_code_before_tl
3974      {
3975          \exp_not:N \columncolor [ #1 ]
3976          { \exp_not:n { #2 } } { \int_use:N \c@jCol }
3977      }
3978  }
3979 }

```

## 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`.

```

3980 \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`.

```

3981 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix:n #1
3982 {
3983     \int_compare:nNnTF \l_@@_first_col_int = 0
3984     { \@@_OnlyMainNiceMatrix_i:n { #1 } }
3985     {
3986         \int_compare:nNnTF \c@jCol = 0
3987         {
3988             \int_compare:nNnF \c@iRow = { -1 }
3989             { \int_compare:nNnF \c@iRow = { \l_@@_last_row_int - 1 } { #1 } }
3990         }
3991         { \@@_OnlyMainNiceMatrix_i:n { #1 } }
3992     }
3993 }

```

This definition may seem complicated by we must remind that the number of row `\c@iRow` is incremented in the first cell of the row, *after* a potential vertical rule on the left side of the first cell.

The command `\@@_OnlyMainNiceMatrix_i:n` is only a short-cut which is used twice in the above command. This command must *not* be protected.

```

3994 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix_i:n #1
3995 {
3996     \int_compare:nNnF \c@iRow = 0
3997     { \int_compare:nNnF \c@iRow = \l_@@_last_row_int { #1 } }
3998 }

```

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 `|`.

```

3999 \cs_new_protected:Npn \@@_vline:nn #1 #2
4000 {

```

The following test is for the case where the user don't use all the columns specified in the preamble of the environment (for instance, a preamble of `|c|c|c|` but only two columns used).

```

4001   \int_compare:nNnT { #1 } < { \c@jCol + 2 }
4002   {
4003     \pgfpicture
4004     \@@_vline_i:nn { #1 } { #2 }
4005     \endpgfpicture
4006   }
4007 }
4008 \cs_new_protected:Npn \@@_vline_i:nn #1 #2
4009 {

```

`\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`.

```

4010   \tl_set:Nx \l_tmpb_tl { #1 }
4011   \tl_clear_new:N \l_tmpc_tl
4012   \int_step_variable:nNn \c@iRow \l_tmpa_tl
4013   {

```

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.

```

4014     \bool_gset_true:N \g_tmpa_bool
4015     \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
4016     { \@@_test_vline_in_block:nnnn ##1 }
4017     \seq_map_inline:Nn \g_@@_pos_of_xdots_seq
4018     { \@@_test_vline_in_block:nnnn ##1 }
4019     \seq_map_inline:Nn \g_@@_pos_of_stroken_blocks_seq
4020     { \@@_test_vline_in_stroken_block:nnnn ##1 }
4021     \clist_if_empty:NF \l_@@_corners_clist
4022     \@@_test_in_corner_v:
4023     \bool_if:NTF \g_tmpa_bool
4024     {
4025       \tl_if_empty:NT \l_tmpc_tl

```

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

```

4026       { \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl }
4027     }
4028   {
4029     \tl_if_empty:NF \l_tmpc_tl
4030     {
4031       \@@_vline_ii:nnnn
4032       { #1 }
4033       { #2 }
4034       \l_tmpc_tl
4035       { \int_eval:n { \l_tmpa_tl - 1 } }
4036       \tl_clear:N \l_tmpc_tl
4037     }
4038   }
4039 }
4040 \tl_if_empty:NF \l_tmpc_tl
4041 {
4042   \@@_vline_ii:nnnn
4043   { #1 }
4044   { #2 }
4045   \l_tmpc_tl
4046   { \int_use:N \c@iRow }
4047   \tl_clear:N \l_tmpc_tl
4048 }
4049 }

```

```

4050 \cs_new_protected:Npn \@@_test_in_corner_v:

```



```

4051 {
4052   \int_compare:nNnTF \l_tmpb_tl = { \@@_succ:n \c@jCol }
4053   {
4054     \seq_if_in:NxT
4055       \l_@@_empty_corner_cells_seq
4056       { \l_tmpa_tl - \@@_pred:n \l_tmpb_tl }
4057       { \bool_set_false:N \g_tmpa_bool }
4058   }
4059   {
4060     \seq_if_in:NxT
4061       \l_@@_empty_corner_cells_seq
4062       { \l_tmpa_tl - \l_tmpb_tl }
4063       {
4064         \int_compare:nNnTF \l_tmpb_tl = 1
4065         { \bool_set_false:N \g_tmpa_bool }
4066         {
4067           \seq_if_in:NxT
4068             \l_@@_empty_corner_cells_seq
4069             { \l_tmpa_tl - \@@_pred:n \l_tmpb_tl }
4070             { \bool_set_false:N \g_tmpa_bool }
4071         }
4072       }
4073   }
4074 }

```

#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.

```

4075 \cs_new_protected:Npn \@@_vline_ii:nnnn #1 #2 #3 #4
4076 {
4077   \pgfrememberpicturepositiononpagetrue
4078   \pgf@relevantforpicturesizefalse
4079   \@@_qpoint:n { row - #3 }
4080   \dim_set_eq:NN \l_tmpa_dim \pgf@y
4081   \@@_qpoint:n { col - #1 }
4082   \dim_set_eq:NN \l_tmpb_dim \pgf@x
4083   \@@_qpoint:n { row - \@@_succ:n { #4 } }
4084   \dim_set_eq:NN \l_tmpc_dim \pgf@y
4085   \bool_lazy_and:nnT
4086     { \int_compare_p:nNn { #2 } > 1 }
4087     { ! \tl_if_blank_p:V \CT@drsc@ }
4088     {
4089       \group_begin:
4090       \CT@drsc@
4091       \dim_add:Nn \l_tmpa_dim { 0.5 \arrayrulewidth }
4092       \dim_sub:Nn \l_tmpc_dim { 0.5 \arrayrulewidth }
4093       \dim_set:Nn \l_tmpd_dim
4094         { \l_tmpb_dim - ( \doublerulesep + \arrayrulewidth ) * ( #2 - 1 ) }
4095       \pgfpathrectanglecorners
4096         { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
4097         { \pgfpoint \l_tmpd_dim \l_tmpc_dim }
4098       \pgfusepath { fill }
4099       \group_end:
4100     }
4101   \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
4102   \pgfpathlineto { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
4103   \prg_replicate:nn { #2 - 1 }
4104   {
4105     \dim_sub:Nn \l_tmpb_dim \arrayrulewidth
4106     \dim_sub:Nn \l_tmpb_dim \doublerulesep
4107     \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
4108     \pgfpathlineto { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
4109   }
4110   \CT@arc@

```

```

4111 \pgfsetlinewidth { 1.1 \arrayrulewidth }
4112 \pgfsetrectcap
4113 \pgfusepathqstroke
4114 }

```

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).

```

4115 \cs_new_protected:Npn \@@_vline_i_complete:nn #1 #2
4116 { \@@_vline_ii:nnnn { #1 } { #2 } 1 { \int_use:N \c@iRow } }

```

The command **\@@\_draw\_hlines:** draws all the vertical rules excepted in the blocks, in the virtual blocks (determined by a command such as **\Cdots**) and in the corners (if the key **corners** is used).

```

4117 \cs_new_protected:Npn \@@_draw_vlines:
4118 {
4119   \int_step_inline:nnn
4120   { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
4121   { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@jCol } \c@jCol }
4122   {
4123     \tl_if_eq:NnF \l_@@_vlines_clist { all }
4124     { \clist_if_in:NnT \l_@@_vlines_clist { ##1 } }
4125     { \@@_vline:nn { ##1 } 1 }
4126   }
4127 }

```

## 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**.

```

4128 \cs_new_protected:Npn \@@_hline:nn #1 #2
4129 {
4130   \pgfpicture
4131   \@@_hline_i:nn { #1 } { #2 }
4132   \endpgfpicture
4133 }
4134 \cs_new_protected:Npn \@@_hline_i:nn #1 #2
4135 {

```

**\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**.

```

4136   \tl_set:Nn \l_tmpa_tl { #1 }
4137   \tl_clear_new:N \l_tmppc_tl
4138   \int_step_variable:nNn \c@jCol \l_tmpb_tl
4139   {

```

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.

```

4140   \bool_gset_true:N \g_tmpa_bool
4141   \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
4142   { \@@_test_hline_in_block:nnnn ##1 }
4143   \seq_map_inline:Nn \g_@@_pos_of_xdots_seq
4144   { \@@_test_hline_in_block:nnnn ##1 }
4145   \seq_map_inline:Nn \g_@@_pos_of_stroken_blocks_seq
4146   { \@@_test_hline_in_stroken_block:nnnn ##1 }
4147   \clist_if_empty:NF \l_@@_corners_clist \@@_test_in_corner_h:
4148   \bool_if:NTF \g_tmpa_bool
4149   {
4150     \tl_if_empty:NT \l_tmppc_tl

```

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

```

4151         { \tl_set_eq:NN \l_tmpc_tl \l_tmpb_tl }
4152     }
4153     {
4154         \tl_if_empty:NF \l_tmpc_tl
4155         {
4156             \@@_hline_ii:nnnn
4157             { #1 }
4158             { #2 }
4159             \l_tmpc_tl
4160             { \int_eval:n { \l_tmpb_tl - 1 } }
4161             \tl_clear:N \l_tmpc_tl
4162         }
4163     }
4164 }
4165 \tl_if_empty:NF \l_tmpc_tl
4166 {
4167     \@@_hline_ii:nnnn
4168     { #1 }
4169     { #2 }
4170     \l_tmpc_tl
4171     { \int_use:N \c@jCol }
4172     \tl_clear:N \l_tmpc_tl
4173 }
4174 }

4175 \cs_new_protected:Npn \@@_test_in_corner_h:
4176 {
4177     \int_compare:nNnTF \l_tmpa_tl = { \@@_succ:n \c@iRow }
4178     {
4179         \seq_if_in:NxT
4180         \l_@@_empty_corner_cells_seq
4181         { \@@_pred:n \l_tmpa_tl - \l_tmpb_tl }
4182         { \bool_set_false:N \g_tmpa_bool }
4183     }
4184     {
4185         \seq_if_in:NxT
4186         \l_@@_empty_corner_cells_seq
4187         { \l_tmpa_tl - \l_tmpb_tl }
4188         {
4189             \int_compare:nNnTF \l_tmpa_tl = 1
4190             { \bool_set_false:N \g_tmpa_bool }
4191             {
4192                 \seq_if_in:NxT
4193                 \l_@@_empty_corner_cells_seq
4194                 { \@@_pred:n \l_tmpa_tl - \l_tmpb_tl }
4195                 { \bool_set_false:N \g_tmpa_bool }
4196             }
4197         }
4198     }
4199 }

```

#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.

```

4200 \cs_new_protected:Npn \@@_hline_ii:nnnn #1 #2 #3 #4
4201 {
4202     \pgfrememberpicturepositiononpagetrue
4203     \pgf@relevantforpicturesizefalse
4204     \@@_qpoint:n { col - #3 }
4205     \dim_set_eq:NN \l_tmpa_dim \pgf@x
4206     \@@_qpoint:n { row - #1 }
4207     \dim_set_eq:NN \l_tmpb_dim \pgf@y

```

```

4208 \@@_qpoint:n { col - \@@_succ:n { #4 } }
4209 \dim_set_eq:NN \l_tmpc_dim \pgf@x
4210 \bool_lazy_and:nnT
4211 { \int_compare_p:nNn { #2 } > 1 }
4212 { ! \tl_if_blank_p:V \CT@drsc@ }
4213 {
4214   \group_begin:
4215   \CT@drsc@
4216   \dim_set:Nn \l_tmpd_dim
4217     { \l_tmpb_dim - ( \doublerulesep + \arrayrulewidth ) * ( #2 - 1 ) }
4218   \pgfpathrectanglecorners
4219     { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
4220     { \pgfpoint \l_tmpc_dim \l_tmpd_dim }
4221   \pgfusepathqfill
4222   \group_end:
4223 }
4224 \pgfpathmoveto { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
4225 \pgfpathlineto { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
4226 \prg_replicate:nn { #2 - 1 }
4227 {
4228   \dim_sub:Nn \l_tmpb_dim \arrayrulewidth
4229   \dim_sub:Nn \l_tmpb_dim \doublerulesep
4230   \pgfpathmoveto { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
4231   \pgfpathlineto { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
4232 }
4233 \CT@arc@
4234 \pgfsetlinewidth { 1.1 \arrayrulewidth }
4235 \pgfsetrectcap
4236 \pgfusepathqstroke
4237 }

4238 \cs_new_protected:Npn \@@_hline_i_complete:nn #1 #2
4239 { \@@_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).

```

4240 \cs_new_protected:Npn \@@_draw_hlines:
4241 {
4242   \int_step_inline:nnn
4243     { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
4244     { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@iRow } \c@iRow }
4245     {
4246       \tl_if_eq:NnF \l_@@_hlines_clist { all }
4247       { \clist_if_in:NnT \l_@@_hlines_clist { ##1 } }
4248       { \@@_hline:nn { ##1 } 1 }
4249     }
4250 }

```

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

```

4251 \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.

```

4252 \cs_set:Npn \@@_Hline_i:n #1
4253 {
4254   \peek_meaning_ignore_spaces:NTF \Hline
4255     { \@@_Hline_ii:nn { #1 + 1 } }
4256     { \@@_Hline_iii:n { #1 } }
4257 }
4258 \cs_set:Npn \@@_Hline_ii:nn #1 #2 { \@@_Hline_i:n { #1 } }

```

```

4259 \cs_set:Npn \@@_Hline_iii:n #1
4260 {
4261   \skip_vertical:n
4262   {
4263     \arrayrulewidth * ( #1 )
4264     + \doublerulesep * ( \int_max:nn 0 { #1 - 1 } )
4265   }
4266   \tl_gput_right:Nx \g_@@_internal_code_after_tl
4267   { \@@_hline:nn { \@@_succ:n { \c@iRow } } { #1 } }
4268   \ifnum 0 = `{ \fi }
4269 }

```

## 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.

```

4270 \cs_new_protected:Npn \@@_test_hline_in_block:nnnn #1 #2 #3 #4
4271 {
4272   \bool_lazy_all:nT
4273   {
4274     { \int_compare_p:nNn \l_tmpa_tl > { #1 } }
4275     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
4276     { \int_compare_p:nNn \l_tmpb_tl > { #2 - 1 } }
4277     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
4278   }
4279   { \bool_gset_false:N \g_tmpa_bool }
4280 }

```

The same for vertical rules.

```

4281 \cs_new_protected:Npn \@@_test_vline_in_block:nnnn #1 #2 #3 #4
4282 {
4283   \bool_lazy_all:nT
4284   {
4285     { \int_compare_p:nNn \l_tmpa_tl > { #1 - 1 } }
4286     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
4287     { \int_compare_p:nNn \l_tmpb_tl > { #2 } }
4288     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
4289   }
4290   { \bool_gset_false:N \g_tmpa_bool }
4291 }
4292 \cs_new_protected:Npn \@@_test_hline_in_stroken_block:nnnn #1 #2 #3 #4
4293 {
4294   \bool_lazy_all:nT
4295   {
4296     { \int_compare_p:nNn \l_tmpa_tl > { #1 - 1 } }
4297     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 2 } }
4298     { \int_compare_p:nNn \l_tmpb_tl > { #2 - 1 } }
4299     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
4300   }
4301   { \bool_gset_false:N \g_tmpa_bool }
4302 }
4303 \cs_new_protected:Npn \@@_test_vline_in_stroken_block:nnnn #1 #2 #3 #4
4304 {
4305   \bool_lazy_all:nT
4306   {
4307     { \int_compare_p:nNn \l_tmpa_tl > { #1 - 1 } }
4308     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
4309     { \int_compare_p:nNn \l_tmpb_tl > { #2 - 1 } }
4310     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 2 } }
4311   }

```

```

4312     { \bool_gset_false:N \g_tmpa_bool }
4313 }

```

## 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.

```

4314 \cs_new_protected:Npn \@@_compute_corners:
4315 {

```

The sequence `\l_@@_empty_corner_cells_seq` will be the sequence of all the empty cells (and not in a block) considered in the corners of the array.

```

4316     \seq_clear_new:N \l_@@_empty_corner_cells_seq
4317     \clist_map_inline:Nn \l_@@_corners_clist
4318     {
4319         \str_case:nnF { ##1 }
4320         {
4321             { NW }
4322             { \@@_compute_a_corner:nnnnnn 1 1 1 1 \c@iRow \c@jCol }
4323             { NE }
4324             { \@@_compute_a_corner:nnnnnn 1 \c@jCol 1 { -1 } \c@iRow 1 }
4325             { SW }
4326             { \@@_compute_a_corner:nnnnnn \c@iRow 1 { -1 } 1 1 \c@jCol }
4327             { SE }
4328             { \@@_compute_a_corner:nnnnnn \c@iRow \c@jCol { -1 } { -1 } 1 1 }
4329         }
4330     { \@@_error:nn { bad~corner } { ##1 } }
4331 }
4332 }

```

“Computing a corner” is determining all the empty cells (which are not in a block) that belong to that corner. These cells will be added to the sequence `\l_@@_empty_corner_cells_seq`.

The six arguments of `\@@_compute_a_corner:nnnnnn` are as follow:

- **#1** and **#2** are the number of row and column of the cell which is actually in the corner;
- **#3** and **#4** are the steps in rows and the step in columns when moving from the corner;
- **#5** is the number of the final row when scanning the rows from the corner;
- **#6** is the number of the final column when scanning the columns from the corner.

```

4333 \cs_new_protected:Npn \@@_compute_a_corner:nnnnnn #1 #2 #3 #4 #5 #6
4334 {

```

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.

```

4335     \bool_set_false:N \l_tmpa_bool
4336     \int_zero_new:N \l_@@_last_empty_row_int
4337     \int_set:Nn \l_@@_last_empty_row_int { #1 }
4338     \int_step_inline:nnnn { #1 } { #3 } { #5 }
4339     {
4340         \@@_test_if_cell_in_a_block:nn { ##1 } { \int_eval:n { #2 } }
4341         \bool_lazy_or:nnTF
4342         {
4343             \cs_if_exist_p:c
4344             { pgf @ sh @ ns @ \@@_env: - ##1 - \int_eval:n { #2 } }
4345         }
4346         \l_tmpb_bool

```

```

4347     { \bool_set_true:N \l_tmpa_bool }
4348     {
4349       \bool_if:NF \l_tmpa_bool
4350       { \int_set:Nn \l_@@_last_empty_row_int { ##1 } }
4351     }
4352   }

```

Now, you determine the last empty cell in the row of number 1.

```

4353   \bool_set_false:N \l_tmpa_bool
4354   \int_zero_new:N \l_@@_last_empty_column_int
4355   \int_set:Nn \l_@@_last_empty_column_int { #2 }
4356   \int_step_inline:nnnn { #2 } { #4 } { #6 }
4357   {
4358     \@@_test_if_cell_in_a_block:nn { \int_eval:n { #1 } } { ##1 }
4359     \bool_lazy_or:nnTF
4360       \l_tmpb_bool
4361       {
4362         \cs_if_exist_p:c
4363           { pgf @ sh @ ns @ \@@_env: - \int_eval:n { #1 } - ##1 }
4364       }
4365     { \bool_set_true:N \l_tmpa_bool }
4366     {
4367       \bool_if:NF \l_tmpa_bool
4368       { \int_set:Nn \l_@@_last_empty_column_int { ##1 } }
4369     }
4370   }

```

Now, we loop over the rows.

```

4371   \int_step_inline:nnnn { #1 } { #3 } { \l_@@_last_empty_row_int
4372     {

```

We treat the row number ##1 with another loop.

```

4373     \bool_set_false:N \l_tmpa_bool
4374     \int_step_inline:nnnn { #2 } { #4 } { \l_@@_last_empty_column_int
4375       {
4376         \@@_test_if_cell_in_a_block:nn { ##1 } { #####1 }
4377         \bool_lazy_or:nnTF
4378           \l_tmpb_bool
4379           {
4380             \cs_if_exist_p:c
4381               { pgf @ sh @ ns @ \@@_env: - ##1 - #####1 }
4382           }
4383         { \bool_set_true:N \l_tmpa_bool }
4384         {
4385           \bool_if:NF \l_tmpa_bool
4386           {
4387             \int_set:Nn \l_@@_last_empty_column_int { #####1 }
4388             \seq_put_right:Nn
4389               \l_@@_empty_corner_cells_seq
4390               { ##1 - #####1 }
4391           }
4392         }
4393       }
4394     }
4395   }

```

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`).

```

4396   \cs_new_protected:Npn \@@_test_if_cell_in_a_block:nn #1 #2
4397   {
4398     \int_set:Nn \l_tmpa_int { #1 }
4399     \int_set:Nn \l_tmpb_int { #2 }
4400     \bool_set_false:N \l_tmpb_bool

```

```

4401 \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
4402 { \@@_test_if_cell_in_block:nnnnnn \l_tmpa_int \l_tmpb_int ##1 }
4403 }
4404 \cs_new_protected:Npn \@@_test_if_cell_in_block:nnnnnn #1 #2 #3 #4 #5 #6
4405 {
4406   \int_compare:nNnT { #3 } < { \@@_succ:n { #1 } }
4407   {
4408     \int_compare:nNnT { #1 } < { \@@_succ:n { #5 } }
4409     {
4410       \int_compare:nNnT { #4 } < { \@@_succ:n { #2 } }
4411       {
4412         \int_compare:nNnT { #2 } < { \@@_succ:n { #6 } }
4413         { \bool_set_true:N \l_tmpb_bool }
4414       }
4415     }
4416   }
4417 }

```

## 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`.

```

4418 \cs_new:Npn \@@_hdottedline:
4419 {
4420   \noalign { \skip_vertical:N 2\l_@@_radius_dim }
4421   \@@_hdottedline_i:
4422 }

```

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

```

4423 \cs_new_protected:Npn \@@_hdottedline_i:
4424 {

```

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).

```

4425 \tl_gput_right:Nx \g_@@_internal_code_after_tl
4426 { \@@_hdottedline:n { \int_use:N \c@iRow } }
4427 }

```

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.

```

4428 \AtBeginDocument
4429 {

```

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}`).

```

4430 \cs_new_protected:Npx \@@_hdottedline:n #1
4431 {
4432   \bool_set_true:N \exp_not:N \l_@@_initial_open_bool
4433   \bool_set_true:N \exp_not:N \l_@@_final_open_bool
4434   \c_@@_pgfortikzpicture_tl
4435   \@@_hdottedline_i:n { #1 }
4436   \c_@@_endpgfortikzpicture_tl
4437 }
4438 }

```



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

```

4439 \cs_new_protected:Npn \@@_hdottedline_i:n #1
4440 {
4441   \pgfrememberpicturepositiononpagetrue
4442   \@@_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).

```

4443   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
4444   \dim_sub:Nn \l_@@_y_initial_dim \l_@@_radius_dim
4445   \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}

```

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

```

4446   \@@_qpoint:n { col - 1 }
4447   \dim_set:Nn \l_@@_x_initial_dim
4448   {
4449     \pgf@x +

```

We do a reduction by `\arraycolsep` for the environments with delimiters (and not for the other).

```

4450     \bool_if:NTF \l_@@_NiceArray_bool \c_zero_dim \arraycolsep
4451     - \l_@@_left_margin_dim
4452   }
4453   \@@_qpoint:n { col - \@@_succ:n \c_jCol }
4454   \dim_set:Nn \l_@@_x_final_dim
4455   {
4456     \pgf@x -
4457     \bool_if:NTF \l_@@_NiceArray_bool \c_zero_dim \arraycolsep
4458     + \l_@@_right_margin_dim
4459   }

```

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.

```

4460   \tl_if_eq:NnF \l_@@_left_delim_tl (
4461     { \dim_gadd:Nn \l_@@_x_initial_dim { 0.5 \l_@@_inter_dots_dim } }
4462   \tl_if_eq:NnF \l_@@_right_delim_tl )
4463   { \dim_gsub:Nn \l_@@_x_final_dim { 0.5 \l_@@_inter_dots_dim } }

```

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

```

4464   \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
4465   \@@_draw_line:
4466 }

```

## Vertical dotted lines

```

4467 \cs_new_protected:Npn \@@_vdottedline:n #1
4468 {
4469   \bool_set_true:N \l_@@_initial_open_bool
4470   \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”.

```

4471   \bool_if:NTF \c_@@_tikz_loaded_bool
4472   {
4473     \tikzpicture
4474     \@@_vdottedline_i:n { #1 }
4475     \endtikzpicture
4476   }
4477   {
4478     \pgfpicture
4479     \@@_vdottedline_i:n { #1 }
4480     \endpgfpicture
4481   }
4482 }

```

```

4483 \cs_new_protected:Npn \@@_vdottedline_i:n #1
4484 {

```

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.

```

4485   \CT@arc@
4486   \pgfrememberpicturepositiononpagetrue
4487   \@@_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).

```

4488   \dim_set:Nn \l_@@_x_initial_dim { \pgf@x - \l_@@_radius_dim }
4489   \dim_set:Nn \l_@@_x_final_dim { \pgf@x - \l_@@_radius_dim }
4490   \@@_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.

```

4491   \dim_set:Nn \l_@@_y_initial_dim { \pgf@y - 0.5 \l_@@_inter_dots_dim }
4492   \@@_qpoint:n { row - \@@_succ:n \c@iRow }
4493   \dim_set:Nn \l_@@_y_final_dim { \pgf@y + 0.5 \l_@@_inter_dots_dim }

```

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

```

4494   \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
4495   \@@_draw_line:
4496 }

```

## 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.

```

4497 \bool_new:N \l_@@_block_auto_columns_width_bool

```

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

```

4498 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
4499 {
4500   auto-columns-width .code:n =
4501   {

```

```

4502     \bool_set_true:N \l_@@_block_auto_columns_width_bool
4503     \dim_gzero_new:N \g_@@_max_cell_width_dim
4504     \bool_set_true:N \l_@@_auto_columns_width_bool
4505   }
4506 }

4507 \NewDocumentEnvironment { NiceMatrixBlock } { ! 0 { } }
4508 {
4509   \int_gincr:N \g_@@_NiceMatrixBlock_int
4510   \dim_zero:N \l_@@_columns_width_dim
4511   \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
4512   \bool_if:NT \l_@@_block_auto_columns_width_bool
4513   {
4514     \cs_if_exist:cT { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
4515     {
4516       \exp_args:NNc \dim_set:Nn \l_@@_columns_width_dim
4517       { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
4518     }
4519   }
4520 }

```

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).

```

4521 {
4522   \bool_if:NT \l_@@_block_auto_columns_width_bool
4523   {
4524     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
4525     \iow_shipout:Nx \@mainaux
4526     {
4527       \cs_gset:cpn
4528       { @@_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.

```

4529       { \dim_eval:n { \g_@@_max_cell_width_dim + \arrayrulewidth } }
4530     }
4531     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
4532   }
4533 }

```

## The extra nodes

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

```

4534 \cs_generate_variant:Nn \dim_min:nn { v n }
4535 \cs_generate_variant:Nn \dim_max:nn { v n }

```

The following command is called in \@@\_use\_arraybox\_with\_notes\_c: just before the construction of the blocks (if the creation of medium nodes is required, medium nodes are also created for the blocks dans that construction uses the standard medium nodes).

```

4536 \cs_new_protected:Npn \@@_create_extra_nodes:
4537 {
4538   \bool_if:nTF \l_@@_medium_nodes_bool
4539   {
4540     \bool_if:nTF \l_@@_large_nodes_bool
4541     \@@_create_medium_and_large_nodes:
4542     \@@_create_medium_nodes:
4543   }
4544   { \bool_if:NT \l_@@_large_nodes_bool \@@_create_large_nodes: }
4545 }

```

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`.

```

4546 \cs_new_protected:Npn \@@_computations_for_medium_nodes:
4547 {
4548   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4549   {
4550     \dim_zero_new:c { l_@@_row_\@@_i: _min_dim }
4551     \dim_set_eq:cN { l_@@_row_\@@_i: _min_dim } \c_max_dim
4552     \dim_zero_new:c { l_@@_row_\@@_i: _max_dim }
4553     \dim_set:cn { l_@@_row_\@@_i: _max_dim } { - \c_max_dim }
4554   }
4555   \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
4556   {
4557     \dim_zero_new:c { l_@@_column_\@@_j: _min_dim }
4558     \dim_set_eq:cN { l_@@_column_\@@_j: _min_dim } \c_max_dim
4559     \dim_zero_new:c { l_@@_column_\@@_j: _max_dim }
4560     \dim_set:cn { l_@@_column_\@@_j: _max_dim } { - \c_max_dim }
4561   }

```

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

```

4562   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4563   {
4564     \int_step_variable:nnNn
4565     \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.

```

4566     {
4567       \cs_if_exist:cT
4568       { 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`.

```

4569     {
4570       \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { south-west }
4571       \dim_set:cn { l_@@_row_\@@_i: _min_dim }
4572       { \dim_min:vn { l_@@_row _ \@@_i: _min_dim } \pgf@y }
4573       \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
4574       {
4575         \dim_set:cn { l_@@_column _ \@@_j: _min_dim }
4576         { \dim_min:vn { l_@@_column _ \@@_j: _min_dim } \pgf@x }
4577       }

```

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`.

```

4578       \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { north-east }
4579       \dim_set:cn { l_@@_row _ \@@_i: _max_dim }
4580       { \dim_max:vn { l_@@_row _ \@@_i: _max_dim } \pgf@y }

```

```

4581         \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
4582         {
4583             \dim_set:cn { l_@@_column _ \@@_j: _ max_dim }
4584             { \dim_max:vn { l_@@_column _ \@@_j: _ max_dim } \pgf@x }
4585         }
4586     }
4587 }
4588 }

```

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

```

4589 \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4590 {
4591     \dim_compare:nNnT
4592     { \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } } = \c_max_dim
4593     {
4594         \@@_qpoint:n { row - \@@_i: - base }
4595         \dim_set:cn { l_@@_row _ \@@_i: _ max _ dim } \pgf@y
4596         \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim } \pgf@y
4597     }
4598 }
4599 \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
4600 {
4601     \dim_compare:nNnT
4602     { \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } } = \c_max_dim
4603     {
4604         \@@_qpoint:n { col - \@@_j: }
4605         \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim } \pgf@x
4606         \dim_set:cn { l_@@_column _ \@@_j: _ min _ dim } \pgf@x
4607     }
4608 }
4609 }

```

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

```

4610 \cs_new_protected:Npn \@@_create_medium_nodes:
4611 {
4612     \pgfpicture
4613     \pgfrememberpicturepositiononpagetrue
4614     \pgf@relevantforpicturesizefalse
4615     \@@_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”.

```

4616     \tl_set:Nn \l_@@_suffix_tl { -medium }
4617     \@@_create_nodes:
4618     \endpgfpicture
4619 }

```

The command `\@@_create_large_nodes:` must be used when we want to create only the “large nodes” and not the medium ones<sup>59</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:`.

```

4620 \cs_new_protected:Npn \@@_create_large_nodes:
4621 {
4622     \pgfpicture
4623     \pgfrememberpicturepositiononpagetrue
4624     \pgf@relevantforpicturesizefalse
4625     \@@_computations_for_medium_nodes:

```

---

<sup>59</sup>If we want to create both, we have to use `\@@_create_medium_and_large_nodes:`

```

4626     \@@_computations_for_large_nodes:
4627     \tl_set:Nn \l_@@_suffix_tl { - large }
4628     \@@_create_nodes:
4629     \endpgfpicture
4630   }
4631 \cs_new_protected:Npn \@@_create_medium_and_large_nodes:
4632 {
4633   \pgfpicture
4634   \pgfrememberpicturepositiononpagetrue
4635   \pgf@relevantforpicturesizefalse
4636   \@@_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”.

```

4637     \tl_set:Nn \l_@@_suffix_tl { - medium }
4638     \@@_create_nodes:
4639     \@@_computations_for_large_nodes:
4640     \tl_set:Nn \l_@@_suffix_tl { - large }
4641     \@@_create_nodes:
4642     \endpgfpicture
4643   }

```

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.

```

4644 \cs_new_protected:Npn \@@_computations_for_large_nodes:
4645 {
4646   \int_set:Nn \l_@@_first_row_int 1
4647   \int_set:Nn \l_@@_first_col_int 1

```

We have to change the values of all the dimensions `l_@@_row_i_min_dim`, `l_@@_row_i_max_dim`, `l_@@_column_j_min_dim` and `l_@@_column_j_max_dim`.

```

4648   \int_step_variable:nNn { \c@iRow - 1 } \@@_i:
4649   {
4650     \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim }
4651     {
4652       (
4653         \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } +
4654         \dim_use:c { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
4655       )
4656       / 2
4657     }
4658     \dim_set_eq:cc { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
4659     { l_@@_row _ \@@_i: _ min _ dim }
4660   }
4661   \int_step_variable:nNn { \c@jCol - 1 } \@@_j:
4662   {
4663     \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim }
4664     {
4665       (
4666         \dim_use:c { l_@@_column _ \@@_j: _ max _ dim } +
4667         \dim_use:c
4668         { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
4669       )
4670       / 2
4671     }
4672     \dim_set_eq:cc { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
4673     { l_@@_column _ \@@_j: _ max _ dim }
4674   }

```

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

```

4675   \dim_sub:cn
4676   { l_@@_column _ 1 _ min _ dim }
4677   \l_@@_left_margin_dim

```

```

4678 \dim_add:cn
4679 { l_@@_column _ \int_use:N \c@jCol _ max _ dim }
4680 \l_@@_right_margin_dim
4681 }

```

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).

```

4682 \cs_new_protected:Npn \@@_create_nodes:
4683 {
4684   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4685   {
4686     \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
4687     {

```

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

```

4688       \@@_pgf_rect_node:nnnnn
4689       { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
4690       { \dim_use:c { l_@@_column _ \@@_j: _min_dim } }
4691       { \dim_use:c { l_@@_row _ \@@_i: _min_dim } }
4692       { \dim_use:c { l_@@_column _ \@@_j: _max_dim } }
4693       { \dim_use:c { l_@@_row _ \@@_i: _max_dim } }
4694       \str_if_empty:NF \l_@@_name_str
4695       {
4696         \pgfnodealias
4697         { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
4698         { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
4699       }
4700     }
4701   }

```

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  $n$ .

```

4702   \seq_mapthread_function:NNN
4703   \g_@@_multicolumn_cells_seq
4704   \g_@@_multicolumn_sizes_seq
4705   \@@_node_for_multicolumn:nn
4706 }

```

```

4707 \cs_new_protected:Npn \@@_extract_coords_values: #1 - #2 \q_stop
4708 {
4709   \cs_set_nopar:Npn \@@_i: { #1 }
4710   \cs_set_nopar:Npn \@@_j: { #2 }
4711 }

```

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”).

```

4712 \cs_new_protected:Npn \@@_node_for_multicolumn:nn #1 #2
4713 {
4714   \@@_extract_coords_values: #1 \q_stop
4715   \@@_pgf_rect_node:nnnnn
4716   { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
4717   { \dim_use:c { l_@@_column _ \@@_j: _min _ dim } }
4718   { \dim_use:c { l_@@_row _ \@@_i: _min _ dim } }
4719   { \dim_use:c { l_@@_column _ \int_eval:n { \@@_j: +#2-1 } _ max _ dim } }
4720   { \dim_use:c { l_@@_row _ \@@_i: _max _ dim } }
4721   \str_if_empty:NF \l_@@_name_str

```

```

4722 {
4723   \pgfnodealias
4724   { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
4725   { \int_use:N \g_@@_env_int - \@@_i: - \@@_j: \l_@@_suffix_tl}
4726 }
4727 }

```

## 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.

```

4728 \keys_define:nn { NiceMatrix / Block / FirstPass }
4729 {
4730   l .code:n = \tl_set:Nn \l_@@_hpos_of_block_tl l ,
4731   l .value_forbidden:n = true ,
4732   r .code:n = \tl_set:Nn \l_@@_hpos_of_block_tl r ,
4733   r .value_forbidden:n = true ,
4734   c .code:n = \tl_set:Nn \l_@@_hpos_of_block_tl c ,
4735   c .value_forbidden:n = true ,
4736   t .code:n = \tl_set:Nn \l_@@_vpos_of_block_tl t ,
4737   t .value_forbidden:n = true ,
4738   b .code:n = \tl_set:Nn \l_@@_vpos_of_block_tl b ,
4739   b .value_forbidden:n = true ,
4740   color .tl_set:N = \l_@@_color_tl ,
4741   color .value_required:n = true ,
4742 }

```

The following command `\@@_Block:` will be linked to `\Block` in the environments of `nicematrix`. We define it with `\NewExpandableDocumentCommand` because it has an optional argument between `<` and `>` (for TeX instructions put before the math mode of the label and before the beginning of the small array of the block). It's mandatory to use an expandable command.

```

4743 \NewExpandableDocumentCommand \@@_Block: { 0 { } m D < > { } m }
4744 {

```

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).

```

4745   \tl_if_blank:nTF { #2 } { \@@_Block_i 1-1 \q_stop } { \@@_Block_i #2 \q_stop }
4746   { #1 } { #3 } { #4 }
4747 }

```

With the following construction, we extract the values of  $i$  and  $j$  in the first mandatory argument of the command.

```

4748 \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.

```

4749 \cs_new_protected:Npn \@@_Block_ii:nnnnn #1 #2 #3 #4 #5
4750 {

```

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).

```

4751   \bool_lazy_or:nnTF
4752   { \tl_if_blank_p:n { #1 } }

```



```

4753 { \str_if_eq_p:nn { #1 } { * } }
4754 { \int_set:Nn \l_tmpa_int { 100 } }
4755 { \int_set:Nn \l_tmpa_int { #1 } }
4756 \bool_lazy_or:nnTF
4757 { \tl_if_blank_p:n { #2 } }
4758 { \str_if_eq_p:nn { #2 } { * } }
4759 { \int_set:Nn \l_tmpb_int { 100 } }
4760 { \int_set:Nn \l_tmpb_int { #2 } }

```

If the block is mono-column.

```

4761 \int_compare:nNnTF \l_tmpb_int = 1
4762 {
4763   \tl_if_empty:NTF \l_@@_cell_type_tl
4764   { \tl_set:Nn \l_@@_hpos_of_block_tl c }
4765   { \tl_set_eq:NN \l_@@_hpos_of_block_tl \l_@@_cell_type_tl }
4766 }
4767 { \tl_set:Nn \l_@@_hpos_of_block_tl c }

```

The value of `\l_@@_hpos_of_block_tl` may be modified by the keys of the command `\Block` that we will analyze now.

```

4768 \keys_set_known:nn { NiceMatrix / Block / FirstPass } { #3 }
4769 \tl_set:Nx \l_tmpa_tl
4770 {
4771   { \int_use:N \c@iRow }
4772   { \int_use:N \c@jCol }
4773   { \int_eval:n { \c@iRow + \l_tmpa_int - 1 } }
4774   { \int_eval:n { \c@jCol + \l_tmpb_int - 1 } }
4775 }

```

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 curryfication).

```

4776 \bool_lazy_or:nnTF
4777 { \int_compare_p:nNn { \l_tmpa_int } = 1 }
4778 { \int_compare_p:nNn { \l_tmpb_int } = 1 }
4779 { \exp_args:Nxx \@@_Block_iv:nnnnn }
4780 { \exp_args:Nxx \@@_Block_v:nnnnn }
4781 { \l_tmpa_int } { \l_tmpb_int } { #3 } { #4 } { #5 }
4782 }

```

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).

```

4783 \cs_new_protected:Npn \@@_Block_iv:nnnnn #1 #2 #3 #4 #5
4784 {
4785   \int_gincr:N \g_@@_block_box_int
4786   \cs_set_protected_nopar:Npn \diagbox ##1 ##2
4787   {
4788     \tl_gput_right:Nx \g_@@_internal_code_after_tl
4789     {
4790       \@@_actually_diagbox:nnnnnn
4791       { \int_use:N \c@iRow }
4792       { \int_use:N \c@jCol }
4793       { \int_eval:n { \c@iRow + #1 - 1 } }
4794       { \int_eval:n { \c@jCol + #2 - 1 } }
4795       { \exp_not:n { ##1 } } { \exp_not:n { ##2 } }

```

```

4796     }
4797   }
4798   \box_gclear_new:c
4799   { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
4800   \hbox_gset:cn
4801   { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
4802   {

```

For a mono-column block, if the user has specified a color for the column in the preamble of the array, we want to fix that color in the box we construct. We do that with `\set@color` and not `\color_ensure_current`: because that command seems to be bugged: it doesn't work in XeLaTeX when `fontspec` is loaded.

```

4803     \tl_if_empty:NTF \l_@@_color_tl
4804     { \int_compare:nNnT { #2 } = 1 \set@color }
4805     { \color { \l_@@_color_tl } }
4806   \group_begin:
4807   \cs_set:Npn \arraystretch { 1 }
4808   \dim_set_eq:NN \extrarowheight \c_zero_dim
4809   #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).

```

4810     \bool_if:NT \g_@@_rotate_bool { \tl_set:Nn \l_@@_hpos_of_block_tl c }
4811     \bool_if:NNT \l_@@_NiceTabular_bool
4812     {
4813       \use:x
4814       {
4815         \exp_not:N \begin { tabular } [ \l_@@_vpos_of_block_tl ]
4816         { @ { } \l_@@_hpos_of_block_tl @ { } }
4817       }
4818       #5
4819       \end { tabular }
4820     }
4821     {
4822       \c_math_toggle_token
4823       \use:x
4824       {
4825         \exp_not:N \begin { array } [ \l_@@_vpos_of_block_tl ]
4826         { @ { } \l_@@_hpos_of_block_tl @ { } }
4827       }
4828       #5
4829       \end { array }
4830       \c_math_toggle_token
4831     }
4832   \group_end:
4833 }
4834 \bool_if:NT \g_@@_rotate_bool
4835 {
4836   \box_grotate:cn
4837   { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
4838   { 90 }
4839   \bool_gset_false:N \g_@@_rotate_bool
4840 }

```

If we are in a mono-column block, we take into account the width of that block for the width of the column.

```

4841     \int_compare:nNnT { #2 } = 1
4842     {
4843       \dim_gset:Nn \g_@@_blocks_wd_dim
4844       {
4845         \dim_max:nn

```

```

4846         \g_@@_blocks_wd_dim
4847         {
4848             \box_wd:c
4849             { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
4850         }
4851     }
4852 }

```

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.

```

4853     \int_compare:nNnT { #1 } = 1
4854     {
4855         \dim_gset:Nn \g_@@_blocks_ht_dim
4856         {
4857             \dim_max:nn
4858             \g_@@_blocks_ht_dim
4859             {
4860                 \box_ht:c
4861                 { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
4862             }
4863         }
4864         \dim_gset:Nn \g_@@_blocks_dp_dim
4865         {
4866             \dim_max:nn
4867             \g_@@_blocks_dp_dim
4868             {
4869                 \box_dp:c
4870                 { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
4871             }
4872         }
4873     }
4874     \seq_gput_right:Nx \g_@@_blocks_seq
4875     {
4876         \l_tmpa_tl

```

In the list of options #3, maybe there is a key for the horizontal alignment (l, r or c). In that case, that key has been read and stored in \l\_@@\_hpos\_of\_block\_tl. However, maybe there were no key of the horizontal alignment and that's why we put a key corresponding to the value of \l\_@@\_hpos\_of\_block\_tl, which is fixed by the type of current column.

```

4877         { \exp_not:n { #3 } , \l_@@_hpos_of_block_tl }
4878         {
4879             \box_use_drop:c
4880             { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
4881         }
4882     }
4883 }

```

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.

```

4884 \cs_new_protected:Npn \@@_Block_v:nnnnn #1 #2 #3 #4 #5
4885 {
4886     \seq_gput_right:Nx \g_@@_blocks_seq
4887     {
4888         \l_tmpa_tl
4889         { \exp_not:n { #3 } }
4890         \exp_not:n
4891         {
4892             {
4893                 \bool_if:NTF \l_@@_NiceTabular_bool
4894                 {
4895                     \group_begin:
4896                     \cs_set:Npn \arraystretch { 1 }

```

```

4897         \dim_set_eq:NN \extrarowheight \c_zero_dim
4898         #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).

```

4899         \bool_if:NT \g_@@_rotate_bool
4900         { \tl_set:Nn \l_@@_hpos_of_block_tl c }
4901     \use:x
4902     {
4903         \exp_not:N \begin { tabular } [ \l_@@_vpos_of_block_tl ]
4904         { @ { } \l_@@_hpos_of_block_tl @ { } }
4905     }
4906     #5
4907     \end { tabular }
4908     \group_end:
4909 }
4910 {
4911     \group_begin:
4912     \cs_set:Npn \arraystretch { 1 }
4913     \dim_set_eq:NN \extrarowheight \c_zero_dim
4914     #4
4915     \bool_if:NT \g_@@_rotate_bool
4916     { \tl_set:Nn \l_@@_hpos_of_block_tl c }
4917     \c_math_toggle_token
4918     \use:x
4919     {
4920         \exp_not:N \begin { array } [ \l_@@_vpos_of_block_tl ]
4921         { @ { } \l_@@_hpos_of_block_tl @ { } }
4922     }
4923     #5
4924     \end { array }
4925     \c_math_toggle_token
4926     \group_end:
4927 }
4928 }
4929 }
4930 }
4931 }

```

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).

```

4932 \keys_define:nn { NiceMatrix / Block / SecondPass }
4933 {
4934     fill .tl_set:N = \l_@@_fill_tl ,
4935     fill .value_required:n = true ,
4936     draw .tl_set:N = \l_@@_draw_tl ,
4937     draw .default:n = default ,
4938     rounded-corners .dim_set:N = \l_@@_rounded_corners_dim ,
4939     rounded-corners .default:n = 4 pt ,
4940     color .code:n = \color { #1 } \tl_set:Nn \l_@@_draw_tl { #1 } ,
4941     color .value_required:n = true ,
4942     borders .clist_set:N = \l_@@_borders_clist ,
4943     borders .value_required:n = true ,
4944     line-width .dim_set:N = \l_@@_line_width_dim ,
4945     line-width .value_required:n = true ,
4946     l .code:n = \tl_set:Nn \l_@@_hpos_of_block_tl l ,
4947     l .value_forbidden:n = true ,
4948     r .code:n = \tl_set:Nn \l_@@_hpos_of_block_tl r ,

```

```

4949   r .value_forbidden:n = true ,
4950   c .code:n = \tl_set:Nn \l_@@_hpos_of_block_tl c ,
4951   c .value_forbidden:n = true ,
4952   t .code:n = \tl_set:Nn \l_@@_vpos_of_block_tl t ,
4953   t .value_forbidden:n = true ,
4954   b .code:n = \tl_set:Nn \l_@@_vpos_of_block_tl b ,
4955   b .value_forbidden:n = true ,
4956   unknown .code:n = \@@_error:n { Unknown-key-for-Block }
4957 }

```

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.

```

4958 \cs_new_protected:Npn \@@_draw_blocks:
4959 {
4960   \cs_set_eq:NN \ialign \@@_old_ialign:
4961   \seq_map_inline:Nn \g_@@_blocks_seq { \@@_Block_iv:nnnnnn #1 }
4962 }
4963 \cs_new_protected:Npn \@@_Block_iv:nnnnnn #1 #2 #3 #4 #5 #6
4964 {

```

The integer `\l_@@_last_row_int` will be the last row of the block and `\l_@@_last_col_int` its last column.

```

4965   \int_zero_new:N \l_@@_last_row_int
4966   \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.

```

4967   \int_compare:nNnTF { #3 } > { 99 }
4968   { \int_set_eq:NN \l_@@_last_row_int \c@iRow }
4969   { \int_set:Nn \l_@@_last_row_int { #3 } }
4970   \int_compare:nNnTF { #4 } > { 99 }
4971   { \int_set_eq:NN \l_@@_last_col_int \c@jCol }
4972   { \int_set:Nn \l_@@_last_col_int { #4 } }
4973   \int_compare:nNnTF \l_@@_last_col_int > \g_@@_col_total_int
4974   {
4975     \int_compare:nTF
4976     { \l_@@_last_col_int <= \g_@@_static_num_of_col_int }
4977     {
4978       \msg_error:nnnn { nicematrix } { Block-too-large~2 } { #1 } { #2 }
4979       \@@_msg_redirect_name:nn { Block-too-large~2 } { none }
4980       \group_begin:
4981       \globaldefs = 1
4982       \@@_msg_redirect_name:nn { columns-not-used } { none }
4983       \group_end:
4984     }
4985     { \msg_error:nnnn { nicematrix } { Block-too-large~1 } { #1 } { #2 } }
4986   }
4987   {
4988     \int_compare:nNnTF \l_@@_last_row_int > \g_@@_row_total_int
4989     { \msg_error:nnnn { nicematrix } { Block-too-large~1 } { #1 } { #2 } }
4990     { \@@_Block_v:nnnnnn { #1 } { #2 } { #3 } { #4 } { #5 } { #6 } }
4991   }
4992 }
4993 \cs_new_protected:Npn \@@_Block_v:nnnnnn #1 #2 #3 #4 #5 #6
4994 {

```

The sequence of the positions of the blocks will be used when drawing the rules (in fact, there is also the `\multicolumn` and the `\diagbox` in that sequence).

```
4995 \seq_gput_left:Nn \g_@@_pos_of_blocks_seq { { #1 } { #2 } { #3 } { #4 } }
```

The group is for the keys.

```
4996 \group_begin:
4997 \keys_set:nn { NiceMatrix / Block / SecondPass } { #5 }

4998 \tl_if_empty:NF \l_@@_draw_tl
4999 {
5000   \tl_gput_right:Nx \g_nicematrix_code_after_tl
5001   {
5002     \@@_stroke_block:nnn
5003     { \exp_not:n { #5 } }
5004     { #1 - #2 }
5005     { \int_use:N \l_@@_last_row_int - \int_use:N \l_@@_last_col_int }
5006   }
5007   \seq_gput_right:Nn \g_@@_pos_of_stroken_blocks_seq
5008   { { #1 } { #2 } { #3 } { #4 } }
5009 }

5010 \clist_if_empty:NF \l_@@_borders_clist
5011 {
5012   \tl_gput_right:Nx \g_nicematrix_code_after_tl
5013   {
5014     \@@_stroke_borders_block:nnn
5015     { \exp_not:n { #5 } }
5016     { #1 - #2 }
5017     { \int_use:N \l_@@_last_row_int - \int_use:N \l_@@_last_col_int }
5018   }
5019 }

5020 \tl_if_empty:NF \l_@@_fill_tl
5021 {
5022   \tl_gput_right:Nx \g_nicematrix_code_before_tl
5023   {
5024     \exp_not:N \roundedrectanglecolor
5025     { \exp_not:V \l_@@_fill_tl }
5026     { #1 - #2 }
5027     { \int_use:N \l_@@_last_row_int - \int_use:N \l_@@_last_col_int }
5028     { \dim_use:N \l_@@_rounded_corners_dim }
5029   }
5030 }

5031 \cs_set_protected_nopar:Npn \diagbox ##1 ##2
5032 {
5033   \tl_gput_right:Nx \g_@@_internal_code_after_tl
5034   {
5035     \@@_actually_diagbox:nnnnnn
5036     { #1 }
5037     { #2 }
5038     { \int_use:N \l_@@_last_row_int }
5039     { \int_use:N \l_@@_last_col_int }
5040     { \exp_not:n { ##1 } } { \exp_not:n { ##2 } }
5041   }
5042 }

5043 \hbox_set:Nn \l_@@_cell_box { \set@color #6 }
5044 \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
```

Let's consider the following `{NiceTabular}`. Because of the instruction `!\hspace{1cm}` in the preamble which increases the space between the columns (by adding, in fact, that space to the previous column, that is to say the second column of the tabular), we will create *two* nodes relative to the block: the node `1-1-block` and the node `1-1-block-short`. The latter will be used by `nicematrix` to put the label of the node. The first one won't be used explicitly.

```

\begin{NiceTabular}{cc!\hspace{1cm}}c}
\Block{2-2}{our block} &      & one    & \\
                        &      & two    & \\
three                  & four & five    & \\
six                    & seven & eight   & \\
\end{NiceTabular}

```

We highlight the node 1-1-block

our block		one
		two
three	four	five
six	seven	eight

We highlight the node 1-1-block-short

our block		one
		two
three	four	five
six	seven	eight

The construction of the node corresponding to the merged cells.

```

5045 \pgfpicture
5046 \pgfrememberpicturepositiononpagetrue
5047 \pgf@relevantforpicturesizefalse
5048 \@@_qpoint:n { row - #1 }
5049 \dim_set_eq:NN \l_tmpa_dim \pgf@y
5050 \@@_qpoint:n { col - #2 }
5051 \dim_set_eq:NN \l_tmpb_dim \pgf@x
5052 \@@_qpoint:n { row - \@@_succ:n { \l_@@_last_row_int } }
5053 \dim_set_eq:NN \l_tmpc_dim \pgf@y
5054 \@@_qpoint:n { col - \@@_succ:n { \l_@@_last_col_int } }
5055 \dim_set_eq:NN \l_tmpd_dim \pgf@x

```

We construct the node for the block with the name (#1-#2-block).

The function `\@@_pgf_rect_node:nnnnn` takes in as arguments the name of the node and the four coordinates of two opposite corner points of the rectangle.

```

5056 \begin { pgfscope }
5057 \@@_pgf_rect_node:nnnnn
5058 { \@@_env: - #1 - #2 - block }
5059 \l_tmpb_dim \l_tmpa_dim \l_tmpd_dim \l_tmpc_dim
5060 \end { pgfscope }

```

We construct the short node.

```

5061 \dim_set_eq:NN \l_tmpb_dim \c_max_dim
5062 \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
5063 {

```

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.

```

5064 \cs_if_exist:cT
5065 { pgf @ sh @ ns @ \@@_env: - ##1 - #2 }
5066 {
5067 \seq_if_in:NnF \g_@@_multicolumn_cells_seq { ##1 - #2 }
5068 {
5069 \pgfpointanchor { \@@_env: - ##1 - #2 } { west }
5070 \dim_set:Nn \l_tmpb_dim { \dim_min:nn \l_tmpb_dim \pgf@x }
5071 }
5072 }
5073 }

```

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.

```

5074 \dim_compare:nNnT \l_tmpb_dim = \c_max_dim
5075 {
5076 \@@_qpoint:n { col - #2 }
5077 \dim_set_eq:NN \l_tmpb_dim \pgf@x
5078 }
5079 \dim_set:Nn \l_tmpd_dim { - \c_max_dim }
5080 \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
5081 {

```

```

5082 \cs_if_exist:cT
5083 { pgf @ sh @ ns @ \@@_env: - ##1 - \int_use:N \l_@@_last_col_int }
5084 {
5085     \seq_if_in:NnF \g_@@_multicolumn_cells_seq { ##1 - #2 }
5086     {
5087         \pgfpointanchor
5088         { \@@_env: - ##1 - \int_use:N \l_@@_last_col_int }
5089         { east }
5090         \dim_set:Nn \l_tmpd_dim { \dim_max:nn \l_tmpd_dim \pgf@x }
5091     }
5092 }
5093 }
5094 \dim_compare:nNnT \l_tmpd_dim = { - \c_max_dim }
5095 {
5096     \@@_qpoint:n { col - \@@_succ:n { \l_@@_last_col_int } }
5097     \dim_set_eq:NN \l_tmpd_dim \pgf@x
5098 }
5099 \@@_pgf_rect_node:nnnnn
5100 { \@@_env: - #1 - #2 - block - short }
5101 \l_tmpb_dim \l_tmpa_dim \l_tmpd_dim \l_tmpe_dim

```

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

```

5102 \bool_if:NT \l_@@_medium_nodes_bool
5103 {
5104     \@@_pgf_rect_node:nnn
5105     { \@@_env: - #1 - #2 - block - medium }
5106     { \pgfpointanchor { \@@_env: - #1 - #2 - medium } { north-west } }
5107     {
5108         \pgfpointanchor
5109         { \@@_env:
5110             - \int_use:N \l_@@_last_row_int
5111             - \int_use:N \l_@@_last_col_int - medium
5112         }
5113         { south-east }
5114     }
5115 }

```

Now, we will put the label of the block beginning with the case of a `\Block` of one row.

```

5116 \int_compare:nNnTF { #1 } = { #3 }
5117 {

```

We take into account the case of a block of one row in the “first row” or the “last row”.

```

5118 \int_compare:nNnTF { #1 } = 0
5119 { \l_@@_code_for_first_row_tl }
5120 {
5121     \int_compare:nNnT { #1 } = \l_@@_last_row_int
5122     \l_@@_code_for_last_row_tl
5123 }

```

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`.

```

5124 \pgfextracty \l_tmpa_dim { \@@_qpoint:n { row - #1 - base } }

```

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

```

5125 \pgfpointanchor
5126 { \@@_env: - #1 - #2 - block - short }
5127 {
5128     \str_case:Vn \l_@@_hpos_of_block_tl
5129     {
5130         c { center }
5131         l { west }
5132         r { east }

```



```

5133     }
5134 }

```

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

```

5135 \pgftransformshift { \pgfpoint \pgf@x \l_tmpa_dim }
5136 \pgfset { inner-sep = \c_zero_dim }
5137 \pgfnode
5138 { rectangle }
5139 {
5140   \str_case:Vn \l_@@_hpos_of_block_tl
5141   {
5142     c { base }
5143     l { base-west }
5144     r { base-east }
5145   }
5146 }
5147 { \box_use_drop:N \l_@@_cell_box } { } { }
5148 }

```

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`).

```

5149 {

```

If we are in the first column, we must put the block as if it was with the key `r`.

```

5150 \int_compare:nNnT { #2 } = 0
5151 { \tl_set:Nn \l_@@_hpos_of_block_tl r }
5152 \bool_if:nT \g_@@_last_col_found_bool
5153 {
5154   \int_compare:nNnT { #2 } = \g_@@_col_total_int
5155   { \tl_set:Nn \l_@@_hpos_of_block_tl l }
5156 }
5157 \pgftransformshift
5158 {
5159   \pgfpointanchor
5160   { \@@_env: - #1 - #2 - block - short }
5161   {
5162     \str_case:Vn \l_@@_hpos_of_block_tl
5163     {
5164       c { center }
5165       l { west }
5166       r { east }
5167     }
5168   }
5169 }
5170 \pgfset { inner-sep = \c_zero_dim }
5171 \pgfnode
5172 { rectangle }
5173 {
5174   \str_case:Vn \l_@@_hpos_of_block_tl
5175   {
5176     c { center }
5177     l { west }
5178     r { east }
5179   }
5180 }
5181 { \box_use_drop:N \l_@@_cell_box } { } { }
5182 }
5183 \endpgfpicture
5184 \group_end:
5185 }

```

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).

```

5186 \cs_new_protected:Npn \@@_stroke_block:nnn #1 #2 #3
5187 {
5188   \group_begin:
5189   \tl_clear:N \l_@@_draw_tl
5190   \dim_set_eq:NN \l_@@_line_width_dim \arrayrulewidth
5191   \keys_set_known:nn { NiceMatrix / BlockStroke } { #1 }
5192   \pgfpicture
5193   \pgfrememberpicturepositiononpagetrue
5194   \pgf@relevantforpicturesizefalse
5195   \tl_if_empty:NF \l_@@_draw_tl
5196   {

```

If the user has used the key `color` of the command `\Block` without value, the color fixed by `\arrayrulecolor` is used.

```

5197     \str_if_eq:VnTF \l_@@_draw_tl { default }
5198     { \CT@arc@ }
5199     { \exp_args:NV \pgfsetstrokecolor \l_@@_draw_tl }
5200   }
5201   \pgfsetcornersarced
5202   {
5203     \pgfpoint
5204     { \dim_use:N \l_@@_rounded_corners_dim }
5205     { \dim_use:N \l_@@_rounded_corners_dim }
5206   }
5207   \@@_cut_on_hyphen:w #2 \q_stop
5208   \bool_lazy_and:nnT
5209   { \int_compare_p:n { \l_tmpa_tl <= \c@iRow } }
5210   { \int_compare_p:n { \l_tmpb_tl <= \c@jCol } }
5211   {
5212     \@@_qpoint:n { row - \l_tmpa_tl }
5213     \dim_set:Nn \l_tmpb_dim { \pgf@y }
5214     \@@_qpoint:n { col - \l_tmpb_tl }
5215     \dim_set:Nn \l_tmpc_dim { \pgf@x }
5216     \@@_cut_on_hyphen:w #3 \q_stop
5217     \int_compare:nNnT \l_tmpa_tl > \c@iRow
5218     { \tl_set:Nx \l_tmpa_tl { \int_use:N \c@iRow } }
5219     \int_compare:nNnT \l_tmpb_tl > \c@jCol
5220     { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
5221     \@@_qpoint:n { row - \@@_succ:n \l_tmpa_tl }
5222     \dim_set:Nn \l_tmpa_dim { \pgf@y }
5223     \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
5224     \dim_set:Nn \l_tmpd_dim { \pgf@x }
5225     \pgfpathrectanglecorners
5226     { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
5227     { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
5228     \pgfsetlinewidth { 1.1 \l_@@_line_width_dim }

```

We can't use `\pgfusepathqstroke` because of the key `rounded-corners`.

```

5229     \pgfusepath { stroke }
5230   }
5231   \endpgfpicture
5232   \group_end:
5233 }

```

Here is the set of keys for the command `\@@_stroke_block:nnn`.

```

5234 \keys_define:nn { NiceMatrix / BlockStroke }
5235 {
5236   color .tl_set:N = \l_@@_draw_tl ,
5237   draw .tl_set:N = \l_@@_draw_tl ,
5238   draw .default:n = default ,
5239   line-width .dim_set:N = \l_@@_line_width_dim ,
5240   rounded-corners .dim_set:N = \l_@@_rounded_corners_dim ,
5241   rounded-corners .default:n = 4 pt
5242 }

```

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).

```

5243 \cs_new_protected:Npn \@@_stroke_borders_block:nnn #1 #2 #3
5244 {
5245   \dim_set_eq:NN \l_@@_line_width_dim \arrayrulewidth
5246   \keys_set_known:nn { NiceMatrix / BlockBorders } { #1 }
5247   \dim_compare:nNnTF \l_@@_rounded_corners_dim > \c_zero_dim
5248     { \@@_error:n { borders~forbidden } }
5249     {
5250       \clist_map_inline:Nn \l_@@_borders_clist
5251       {
5252         \clist_if_in:nnF { top , bottom , left , right } { ##1 }
5253         { \@@_error:nn { bad~border } { ##1 } }
5254       }
5255       \@@_cut_on_hyphen:w #2 \q_stop
5256       \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl
5257       \tl_set_eq:NN \l_tmpd_tl \l_tmpb_tl
5258       \@@_cut_on_hyphen:w #3 \q_stop
5259       \tl_set:Nx \l_tmpa_tl { \int_eval:n { \l_tmpa_tl + 1 } }
5260       \tl_set:Nx \l_tmpb_tl { \int_eval:n { \l_tmpb_tl + 1 } }
5261       \pgfpicture
5262       \pgfrememberpicturepositiononpagetrue
5263       \pgf@relevantforpicturesizefalse
5264       \CT@arc@
5265       \pgfsetlinewidth { 1.1 \l_@@_line_width_dim }
5266       \clist_if_in:NnT \l_@@_borders_clist { right }
5267       { \@@_stroke_vertical:n \l_tmpb_tl }
5268       \clist_if_in:NnT \l_@@_borders_clist { left }
5269       { \@@_stroke_vertical:n \l_tmpd_tl }
5270       \clist_if_in:NnT \l_@@_borders_clist { bottom }
5271       { \@@_stroke_horizontal:n \l_tmpa_tl }
5272       \clist_if_in:NnT \l_@@_borders_clist { top }
5273       { \@@_stroke_horizontal:n \l_tmpc_tl }
5274       \endpgfpicture
5275     }
5276 }

```

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).

```

5277 \cs_new_protected:Npn \@@_stroke_vertical:n #1
5278 {
5279   \@@_qpoint:n \l_tmpc_tl
5280   \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \l_@@_line_width_dim }
5281   \@@_qpoint:n \l_tmpa_tl
5282   \dim_set:Nn \l_tmpc_dim { \pgf@y + 0.5 \l_@@_line_width_dim }
5283   \@@_qpoint:n { #1 }
5284   \pgfpathmoveto { \pgfpoint \pgf@x \l_tmpb_dim }
5285   \pgfpathlineto { \pgfpoint \pgf@x \l_tmpc_dim }
5286   \pgfusepathqstroke
5287 }

```

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).

```

5288 \cs_new_protected:Npn \@@_stroke_horizontal:n #1
5289 {
5290   \@@_qpoint:n \l_tmpd_tl
5291   \clist_if_in:NnTF \l_@@_borders_clist { left }
5292     { \dim_set:Nn \l_tmpa_dim { \pgf@x - 0.5 \l_@@_line_width_dim } }
5293     { \dim_set:Nn \l_tmpa_dim { \pgf@x + 0.5 \l_@@_line_width_dim } }
5294   \@@_qpoint:n \l_tmpb_tl
5295   \dim_set:Nn \l_tmpb_dim { \pgf@x + 0.5 \l_@@_line_width_dim }
5296   \@@_qpoint:n { #1 }

```

```

5297 \pgfpathmoveto { \pgfpoint \l_tmpa_dim \pgf@y }
5298 \pgfpathlineto { \pgfpoint \l_tmpb_dim \pgf@y }
5299 \pgfusepathqstroke
5300 }

```

Here is the set of keys for the command `\@@_stroke_borders_block:nnn`.

```

5301 \keys_define:nn { NiceMatrix / BlockBorders }
5302 {
5303   borders .clist_set:N = \l_@@_borders_clist ,
5304   rounded-corners .dim_set:N = \l_@@_rounded_corners_dim ,
5305   rounded-corners .default:n = 4 pt ,
5306   line-width .dim_set:N = \l_@@_line_width_dim
5307 }

```

## How to draw the dotted lines transparently

```

5308 \cs_set_protected:Npn \@@_renew_matrix:
5309 {
5310   \RenewDocumentEnvironment { pmatrix } { } {
5311     { \pNiceMatrix }
5312     { \endpNiceMatrix }
5313   \RenewDocumentEnvironment { vmatrix } { } {
5314     { \vNiceMatrix }
5315     { \endvNiceMatrix }
5316   \RenewDocumentEnvironment { Vmatrix } { } {
5317     { \VNiceMatrix }
5318     { \endVNiceMatrix }
5319   \RenewDocumentEnvironment { bmatrix } { } {
5320     { \bNiceMatrix }
5321     { \endbNiceMatrix }
5322   \RenewDocumentEnvironment { Bmatrix } { } {
5323     { \BNiceMatrix }
5324     { \endBNiceMatrix }
5325   }

```

## Automatic arrays

```

5326 \cs_new_protected:Npn \@@_set_size:n #1-#2 \q_stop
5327 {
5328   \int_set:Nn \l_@@_nb_rows_int { #1 }
5329   \int_set:Nn \l_@@_nb_cols_int { #2 }
5330 }
5331 \NewDocumentCommand \AutoNiceMatrixWithDelims { m m O { } m O { } m ! O { } }
5332 {
5333   \int_zero_new:N \l_@@_nb_rows_int
5334   \int_zero_new:N \l_@@_nb_cols_int
5335   \@@_set_size:n #4 \q_stop
5336   \begin { NiceArrayWithDelims } { #1 } { #2 }
5337     { * { \l_@@_nb_cols_int } { c } } [ #3 , #5 , #7 ]
5338   \int_compare:nNnT \l_@@_first_row_int = 0
5339     {
5340       \int_compare:nNnT \l_@@_first_col_int = 0 { & }
5341       \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
5342       \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
5343     }
5344   \prg_replicate:nn \l_@@_nb_rows_int
5345     {
5346       \int_compare:nNnT \l_@@_first_col_int = 0 { & }

```

You put `{ }` before `#6` to avoid a hasty expansion of a potential `\arabic{iRow}` at the beginning of the row which would result in an incorrect value of that `iRow` (since `iRow` is incremented in the first cell of the row of the `\halign`).

```

5347     \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { { } #6 & } #6
5348     \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\\
5349   }
5350   \int_compare:nNnT \l_@@_last_row_int > { -2 }
5351   {
5352     \int_compare:nNnT \l_@@_first_col_int = 0 { & }
5353     \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
5354     \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\\
5355   }
5356   \end { NiceArrayWithDelims }
5357 }
5358 \cs_set_protected:Npn \@@_define_com:nnn #1 #2 #3
5359 {
5360   \cs_set_protected:cpn { #1 AutoNiceMatrix }
5361   {
5362     \str_gset:Nx \g_@@_name_env_str { #1 AutoNiceMatrix }
5363     \AutoNiceMatrixWithDelims { #2 } { #3 }
5364   }
5365 }
5366 \@@_define_com:nnn p ( )
5367 \@@_define_com:nnn b [ ]
5368 \@@_define_com:nnn v | |
5369 \@@_define_com:nnn V \| \|
5370 \@@_define_com:nnn B \{ \}

```

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

```

5371 \NewDocumentCommand \AutoNiceMatrix { 0 { } m 0 { } m ! 0 { } }
5372 {
5373   \group_begin:
5374   \bool_set_true:N \l_@@_NiceArray_bool
5375   \AutoNiceMatrixWithDelims . . { #2 } { #4 } [ #1 , #3 , #5 ]
5376   \group_end:
5377 }

```

## The redefinition of the command `\dotfill`

```

5378 \cs_set_eq:NN \@@_old_dotfill \dotfill
5379 \cs_new_protected:Npn \@@_dotfill:
5380 {

```

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}`).

```

5381   \@@_old_dotfill
5382   \bool_if:NT \l_@@_NiceTabular_bool
5383   { \group_insert_after:N \@@_dotfill_ii: }
5384   { \group_insert_after:N \@@_dotfill_i: }
5385 }
5386 \cs_new_protected:Npn \@@_dotfill_i: { \group_insert_after:N \@@_dotfill_ii: }
5387 \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`.

```

5388 \cs_new_protected:Npn \@@_dotfill_iii:
5389 { \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`.

```

5390 \cs_new_protected:Npn \@@_diagbox:nn #1 #2
5391 {
5392   \tl_gput_right:Nx \g_@@_internal_code_after_tl

```

```

5393 {
5394   \@@_actually_diagbox:nnnnnn
5395   { \int_use:N \c@iRow }
5396   { \int_use:N \c@jCol }
5397   { \int_use:N \c@iRow }
5398   { \int_use:N \c@jCol }
5399   { \exp_not:n { #1 } }
5400   { \exp_not:n { #2 } }
5401 }

```

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`.

```

5402 \seq_gput_right:Nx \g_@@_pos_of_blocks_seq
5403 {
5404   { \int_use:N \c@iRow }
5405   { \int_use:N \c@jCol }
5406   { \int_use:N \c@iRow }
5407   { \int_use:N \c@jCol }
5408 }
5409 }

```

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.

```

5410 \cs_new_protected:Npn \@@_actually_diagbox:nnnnnn #1 #2 #3 #4 #5 #6
5411 {
5412   \pgfpicture
5413   \pgf@relevantforpicturesizefalse
5414   \pgfrememberpicturepositiononpagetrue
5415   \@@_qpoint:n { row - #1 }
5416   \dim_set_eq:NN \l_tmpa_dim \pgf@y
5417   \@@_qpoint:n { col - #2 }
5418   \dim_set_eq:NN \l_tmpb_dim \pgf@x
5419   \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
5420   \@@_qpoint:n { row - \@@_succ:n { #3 } }
5421   \dim_set_eq:NN \l_tmpc_dim \pgf@y
5422   \@@_qpoint:n { col - \@@_succ:n { #4 } }
5423   \dim_set_eq:NN \l_tmpd_dim \pgf@x
5424   \pgfpathlineto { \pgfpoint \l_tmpd_dim \l_tmpc_dim }
5425   {

```

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.

```

5426   \CT@arc@
5427   \pgfsetroundcap
5428   \pgfusepathqstroke
5429 }
5430 \pgfset { inner~sep = 1 pt }
5431 \pgfscope
5432 \pgftransformshift { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
5433 \pgfnode { rectangle } { south~west }
5434 { \@@_math_toggle_token: #5 \@@_math_toggle_token: } { } { }
5435 \endpgfscope
5436 \pgftransformshift { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
5437 \pgfnode { rectangle } { north~east }
5438 { \@@_math_toggle_token: #6 \@@_math_toggle_token: } { } { }
5439 \endpgfpicture
5440 }

```

## 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.

```

5441 \keys_define:nn { NiceMatrix }
5442 {
5443   CodeAfter / rules .inherit:n = NiceMatrix / rules ,
5444   CodeAfter / sub-matrix .inherit:n = NiceMatrix / sub-matrix
5445 }
5446 \keys_define:nn { NiceMatrix / CodeAfter }
5447 {
5448   sub-matrix .code:n = \keys_set:nn { NiceMatrix / sub-matrix } { #1 } ,
5449   sub-matrix .value_required:n = true ,
5450   delimiters / color .tl_set:N = \l_@@_delimiters_color_tl ,
5451   delimiters / color .value_required:n = true ,
5452   rules .code:n = \keys_set:nn { NiceMatrix / rules } { #1 } ,
5453   rules .value_required:n = true ,
5454   unknown .code:n = \@@_error:n { Unknown~key~for~CodeAfter }
5455 }
```

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. 98.

In the environments of `nicematrix`, `\CodeAfter` will be linked to `\@@_CodeAfter:`. That macro must *not* be protected since it begins with `\omit`.

```

5456 \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`.

```

5457 \cs_new_protected:Npn \@@_CodeAfter_i:n #1 \end
5458 {
5459   \tl_gput_right:Nn \g_nicematrix_code_after_tl { #1 }
5460   \@@_CodeAfter_ii:n
5461 }
```

We catch the argument of the command `\end` (in `#1`).

```

5462 \cs_new_protected:Npn \@@_CodeAfter_ii:n #1
5463 {
```

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.

```

5464   \str_if_eq:eeTF \@currenvir { #1 } { \end { #1 } }
```

If this is not the `\end` we are looking for, we put those tokens in `\g_nicematrix_code_after_tl` and we go on searching for the next command `\end` with a recursive call to the command `\@@_CodeAfter:n`.

```

5465   {
5466     \tl_gput_right:Nn \g_nicematrix_code_after_tl { \end { #1 } }
5467     \@@_CodeAfter_i:n
5468   }
5469 }
```

## 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_t1` (and also potentially add instructions in the preamble provided to `\array` in order to add spaces between columns).

The first argument is the type of delimiter (`(`, `[`, `\{`, `)`, `]` ou `\}`). 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.

```

5470 \cs_new_protected:Npn \@@_delimiter:nnn #1 #2 #3
5471 {
5472   \pgfpicture
5473   \pgfrememberpicturepositiononpagetrue
5474   \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.

```

5475   \@@_qpoint:n { row - 1 }
5476   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
5477   \@@_qpoint:n { row - \@@_succ:n \c@iRow }
5478   \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).

```

5479   \bool_if:nTF { #3 }
5480   { \dim_set_eq:NN \l_tmpa_dim \c_max_dim }
5481   { \dim_set:Nn \l_tmpa_dim { - \c_max_dim } }
5482   \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
5483   {
5484     \cs_if_exist:cT
5485     { pgf @ sh @ ns @ \@@_env: - ##1 - #2 }
5486     {
5487       \pgfpointanchor
5488       { \@@_env: - ##1 - #2 }
5489       { \bool_if:nTF { #3 } { west } { east } }
5490       \dim_set:Nn \l_tmpa_dim
5491       { \bool_if:nTF { #3 } \dim_min:nn \dim_max:nn \l_tmpa_dim \pgf@x }
5492     }
5493   }

```

Now we can put the delimiter with a node of PGF.

```

5494   \pgfset { inner~sep = \c_zero_dim }
5495   \dim_zero:N \nulldelimiterspace
5496   \pgftransformshift
5497   {
5498     \pgfpoint
5499     { \l_tmpa_dim }
5500     { ( \l_@@_y_initial_dim + \l_@@_y_final_dim + \arrayrulewidth ) / 2 }
5501   }
5502   \pgfnode
5503   { rectangle }
5504   { \bool_if:nTF { #3 } { east } { west } }
5505   {

```

Here is the content of the PGF node, that is to say the delimiter, constructed with its right size.

```

5506   \nullfont
5507   \c_math_toggle_token
5508   \tl_if_empty:NF \l_@@_delimiters_color_tl
5509   { \color { \l_@@_delimiters_color_tl } }
5510   \bool_if:nTF { #3 } { \left #1 } { \left . }
5511   \vcenter
5512   {
5513     \nullfont
5514     \hrule \@height
5515     \dim_eval:n { \l_@@_y_initial_dim - \l_@@_y_final_dim }
5516     \@depth \c_zero_dim

```



```

5517         \@width \c_zero_dim
5518     }
5519     \bool_if:nTF { #3 } { \right . } { \right #1 }
5520     \c_math_toggle_token
5521 }
5522 { }
5523 { }
5524 \endpgfpicture
5525 }

```

## The command `\SubMatrix`

```

5526 \keys_define:nn { NiceMatrix / sub-matrix }
5527 {
5528     extra-height .dim_set:N = \l_@@_submatrix_extra_height_dim ,
5529     extra-height .value_required:n = true ,
5530     left-xshift .dim_set:N = \l_@@_submatrix_left_xshift_dim ,
5531     left-xshift .value_required:n = true ,
5532     right-xshift .dim_set:N = \l_@@_submatrix_right_xshift_dim ,
5533     right-xshift .value_required:n = true ,
5534     xshift .meta:n = { left-xshift = #1, right-xshift = #1 } ,
5535     xshift .value_required:n = true ,
5536     delimiters / color .tl_set:N = \l_@@_delimiters_color_tl ,
5537     delimiters / color .value_required:n = true ,
5538     slim .bool_set:N = \l_@@_submatrix_slim_bool ,
5539     slim .default:n = true ,
5540     hlines .clist_set:N = \l_@@_submatrix_hlines_clist ,
5541     hlines .default:n = all ,
5542     vlines .clist_set:N = \l_@@_submatrix_vlines_clist ,
5543     vlines .default:n = all ,
5544     hvlines .meta:n = { hlines, vlines } ,
5545     hvlines .value_forbidden:n = true ,
5546 }
5547 \keys_define:nn { NiceMatrix }
5548 {
5549     SubMatrix .inherit:n = NiceMatrix / sub-matrix ,
5550     CodeAfter / sub-matrix .inherit:n = NiceMatrix / sub-matrix ,
5551     NiceMatrix / sub-matrix .inherit:n = NiceMatrix / sub-matrix ,
5552     NiceArray / sub-matrix .inherit:n = NiceMatrix / sub-matrix ,
5553     pNiceArray / sub-matrix .inherit:n = NiceMatrix / sub-matrix ,
5554     NiceMatrixOptions / sub-matrix .inherit:n = NiceMatrix / sub-matrix ,
5555 }

```

The following keys set is for the command `\SubMatrix` itself (not the tuning of `\SubMatrix` that can be done elsewhere).

```

5556 \keys_define:nn { NiceMatrix / SubMatrix }
5557 {
5558     hlines .clist_set:N = \l_@@_submatrix_hlines_clist ,
5559     hlines .default:n = all ,
5560     vlines .clist_set:N = \l_@@_submatrix_vlines_clist ,
5561     vlines .default:n = all ,
5562     hvlines .meta:n = { hlines, vlines } ,
5563     hvlines .value_forbidden:n = true ,
5564     name .code:n =
5565         \tl_if_empty:nTF { #1 }
5566         { \@@_error:n { Invalid-name-format } }
5567         {
5568             \regex_match:nnTF { \A[A-Za-z][A-Za-z0-9]*\Z } { #1 }
5569             {
5570                 \seq_if_in:NnTF \g_@@_submatrix_names_seq { #1 }
5571                 { \@@_error:nn { Duplicate-name-for-SubMatrix } { #1 } }
5572                 {
5573                     \str_set:Nn \l_@@_submatrix_name_str { #1 }

```

```

5574         \seq_gput_right:Nn \g_@@_submatrix_names_seq { #1 }
5575     }
5576 }
5577 { \@@_error:n { Invalid~name~format } }
5578 } ,
5579 rules .code:n = \keys_set:nn { NiceMatrix / rules } { #1 } ,
5580 rules .value_required:n = true ,
5581 code .tl_set:N = \l_@@_code_tl ,
5582 code .value_required:n = true ,
5583 name .value_required:n = true ,
5584 unknown .code:n = \@@_error:n { Unknown~key~for~SubMatrix }
5585 }

5586 \NewDocumentCommand \@@_SubMatrix_in_code_before { m m m m ! O { } }
5587 {
5588     \@@_cut_on_hyphen:w #3 \q_stop
5589     \tl_clear_new:N \l_tmpc_tl
5590     \tl_clear_new:N \l_tmpd_tl
5591     \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl
5592     \tl_set_eq:NN \l_tmpd_tl \l_tmpb_tl
5593     \@@_cut_on_hyphen:w #2 \q_stop
5594     \seq_gput_right:Nx \g_@@_submatrix_seq
5595         { { \l_tmpa_tl } { \l_tmpb_tl } { \l_tmpc_tl } { \l_tmpd_tl } }
5596     \tl_gput_right:Nn \g_@@_internal_code_after_tl
5597         { \SubMatrix { #1 } { #2 } { #3 } { #4 } [ #5 ] }
5598 }

```

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.

```

5599 \NewDocumentCommand \@@_SubMatrix { m m m m O { } }
5600 {
5601     \group_begin:

```

The four following token lists correspond to the position of the `\SubMatrix`.

```

5602     \tl_clear_new:N \l_@@_first_i_tl
5603     \tl_clear_new:N \l_@@_first_j_tl
5604     \tl_clear_new:N \l_@@_last_i_tl
5605     \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`.

```

5606     \@@_cut_on_hyphen:w #2 \q_stop
5607     \tl_set_eq:NN \l_@@_first_i_tl \l_tmpa_tl
5608     \tl_set_eq:NN \l_@@_first_j_tl \l_tmpb_tl
5609     \@@_cut_on_hyphen:w #3 \q_stop
5610     \tl_set_eq:NN \l_@@_last_i_tl \l_tmpa_tl
5611     \tl_set_eq:NN \l_@@_last_j_tl \l_tmpb_tl
5612     \bool_lazy_or:nnTF
5613         { \int_compare_p:nNn \l_@@_last_i_tl > \g_@@_row_total_int }
5614         { \int_compare_p:nNn \l_@@_last_j_tl > \g_@@_col_total_int }
5615         { \@@_error:n { SubMatrix~too~large } }
5616     {
5617         \str_clear_new:N \l_@@_submatrix_name_str
5618         \keys_set:nn { NiceMatrix / SubMatrix } { #5 }

```

```

5619 \pgfpicture
5620 \pgfrememberpicturerepositiononpagetrue
5621 \pgf@relevantforpicturesizefalse
5622 \pgfset { inner-sep = \c_zero_dim }
5623 \dim_set_eq:NN \l_@@_x_initial_dim \c_max_dim
5624 \dim_set:Nn \l_@@_x_final_dim { - \c_max_dim }

```

The last value of \int\_step\_inline:nnn is provided by currfication.

```

5625 \bool_if:NTF \l_@@_submatrix_slim_bool
5626 { \int_step_inline:nnn \l_@@_first_i_tl \l_@@_last_i_tl }
5627 { \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int }
5628 {
5629   \cs_if_exist:cT
5630   { pgf @ sh @ ns @ \@@_env: - ##1 - \l_@@_first_j_tl }
5631   {
5632     \pgfpointanchor { \@@_env: - ##1 - \l_@@_first_j_tl } { west }
5633     \dim_set:Nn \l_@@_x_initial_dim
5634     { \dim_min:nn \l_@@_x_initial_dim \pgf@x }
5635   }
5636   \cs_if_exist:cT
5637   { pgf @ sh @ ns @ \@@_env: - ##1 - \l_@@_last_j_tl }
5638   {
5639     \pgfpointanchor { \@@_env: - ##1 - \l_@@_last_j_tl } { east }
5640     \dim_set:Nn \l_@@_x_final_dim
5641     { \dim_max:nn \l_@@_x_final_dim \pgf@x }
5642   }
5643 }
5644 \@@_qpoint:n { row - \l_@@_first_i_tl - base }
5645 \dim_set:Nn \l_@@_y_initial_dim
5646 { \pgf@y + ( \box_ht:N \strutbox + \extrarowheight ) * \arraystretch }
5647 \@@_qpoint:n { row - \l_@@_last_i_tl - base }
5648 \dim_set:Nn \l_@@_y_final_dim
5649 { \pgf@y - ( \box_dp:N \strutbox ) * \arraystretch }
5650 \int_step_inline:nnn \l_@@_first_col_int \g_@@_col_total_int
5651 {
5652   \cs_if_exist:cT
5653   { pgf @ sh @ ns @ \@@_env: - \l_@@_first_i_tl - ##1 }
5654   {
5655     \pgfpointanchor { \@@_env: - \l_@@_first_i_tl - ##1 } { north }
5656     \dim_set:Nn \l_@@_y_initial_dim
5657     { \dim_max:nn \l_@@_y_initial_dim \pgf@y }
5658   }
5659   \cs_if_exist:cT
5660   { pgf @ sh @ ns @ \@@_env: - \l_@@_last_i_tl - ##1 }
5661   {
5662     \pgfpointanchor { \@@_env: - \l_@@_last_i_tl - ##1 } { south }
5663     \dim_set:Nn \l_@@_y_final_dim
5664     { \dim_min:nn \l_@@_y_final_dim \pgf@y }
5665   }
5666 }
5667 \dim_set:Nn \l_tmpa_dim
5668 {
5669   \l_@@_y_initial_dim - \l_@@_y_final_dim +
5670   \l_@@_submatrix_extra_height_dim - \arrayrulewidth
5671 }
5672 \dim_set_eq:NN \nulldelimiterspace \c_zero_dim

```

We will draw the rules in the \SubMatrix.

```

5673 \group_begin:
5674 \pgfsetlinewidth { 1.1 \arrayrulewidth }
5675 \tl_if_empty:NF \l_@@_rules_color_tl
5676 { \exp_after:wN \@@_set_CT@arc@: \l_@@_rules_color_tl \q_stop }
5677 \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`.

```

5678     \seq_map_inline:Nn \g_@@_cols_vlism_seq
5679     {
5680         \int_compare:nNnT \l_@@_first_j_tl < { ##1 }
5681         {
5682             \int_compare:nNnT
5683             { ##1 } < { \int_eval:n { \l_@@_last_j_tl + 1 } }
5684             {

```

First, we extract the value of the abscissa of the rule we have to draw.

```

5685                 \@@_qpoint:n { col - ##1 }
5686                 \pgfpathmoveto { \pgfpoint \pgf@x \l_@@_y_initial_dim }
5687                 \pgfpathlineto { \pgfpoint \pgf@x \l_@@_y_final_dim }
5688                 \pgfusepathqstroke
5689             }
5690         }
5691     }

```

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.

```

5692     \tl_if_eq:NnTF \l_@@_submatrix_vlines_clist { all }
5693     { \int_step_inline:nn { \l_@@_last_j_tl - \l_@@_first_j_tl } }
5694     { \clist_map_inline:Nn \l_@@_submatrix_vlines_clist }
5695     {
5696         \bool_lazy_and:nnTF
5697         { \int_compare_p:nNn { ##1 } > 0 }
5698         {
5699             \int_compare_p:nNn
5700             { ##1 } < { \l_@@_last_j_tl - \l_@@_first_j_tl + 1 } }
5701         {
5702             \@@_qpoint:n { col - \int_eval:n { ##1 + \l_@@_first_j_tl } }
5703             \pgfpathmoveto { \pgfpoint \pgf@x \l_@@_y_initial_dim }
5704             \pgfpathlineto { \pgfpoint \pgf@x \l_@@_y_final_dim }
5705             \pgfusepathqstroke
5706         }
5707         { \@@_error:nnn { Wrong-line-in-SubMatrix } { vertical } { ##1 } }
5708     }

```

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.

```

5709     \tl_if_eq:NnTF \l_@@_submatrix_hlines_clist { all }
5710     { \int_step_inline:nn { \l_@@_last_i_tl - \l_@@_first_i_tl } }
5711     { \clist_map_inline:Nn \l_@@_submatrix_hlines_clist }
5712     {
5713         \bool_lazy_and:nnTF
5714         { \int_compare_p:nNn { ##1 } > 0 }
5715         {
5716             \int_compare_p:nNn
5717             { ##1 } < { \l_@@_last_i_tl - \l_@@_first_i_tl + 1 } }
5718         {
5719             \@@_qpoint:n { row - \int_eval:n { ##1 + \l_@@_first_i_tl } }

```

We use a group to protect `\l_tmpa_dim` and `\l_tmpb_dim`.

```

5720     \group_begin:

```

We compute in `\l_tmpa_dim` the  $x$ -value of the left end of the rule.

```

5721         \dim_set:Nn \l_tmpa_dim
5722         { \l_@@_x_initial_dim - \l_@@_submatrix_left_xshift_dim }
5723         \str_case:nn { #1 }
5724         {
5725             ( { \dim_sub:Nn \l_tmpa_dim { 0.9 mm } }
5726             [ { \dim_sub:Nn \l_tmpa_dim { 0.2 mm } }

```

```

5727         \{ { \dim_sub:Nn \l_tmpa_dim { 0.9 mm } }
5728     }
5729     \pgfpathmoveto { \pgfpoint \l_tmpa_dim \pgf@y }
We compute in \l_tmpb_dim the  $x$ -value of the right end of the rule.
5730     \dim_set:Nn \l_tmpb_dim
5731     { \l_@@_x_final_dim + \l_@@_submatrix_right_xshift_dim }
5732     \str_case:nn { #4 }
5733     {
5734         ) { \dim_add:Nn \l_tmpb_dim { 0.9 mm } }
5735         ] { \dim_add:Nn \l_tmpb_dim { 0.2 mm } }
5736         \} { \dim_add:Nn \l_tmpb_dim { 0.9 mm } }
5737     }
5738     \pgfpathlineto { \pgfpoint \l_tmpb_dim \pgf@y }
5739     \pgfusepathqstroke
5740     \group_end:
5741 }
5742 { \@@_error:nnn { Wrong-line-in-SubMatrix } { horizontal } { ##1 } }
5743 }

```

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).

```

5744     \str_if_empty:NF \l_@@_submatrix_name_str
5745     {
5746         \@@_pgf_rect_node:nnnnn \l_@@_submatrix_name_str
5747         \l_@@_x_initial_dim \l_@@_y_initial_dim
5748         \l_@@_x_final_dim \l_@@_y_final_dim
5749     }
5750     \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`.

```

5751     \begin { pgfscope }
5752     \pgftransformshift
5753     {
5754         \pgfpoint
5755         { \l_@@_x_initial_dim - \l_@@_submatrix_left_xshift_dim }
5756         { ( \l_@@_y_initial_dim + \l_@@_y_final_dim ) / 2 }
5757     }
5758     \str_if_empty:NTF \l_@@_submatrix_name_str
5759     { \@@_node_left:nn #1 { } }
5760     { \@@_node_left:nn #1 { \@@_env: - \l_@@_submatrix_name_str - left } }
5761     \end { pgfscope }

```

Now, we deal with the right delimiter.

```

5762     \pgftransformshift
5763     {
5764         \pgfpoint
5765         { \l_@@_x_final_dim + \l_@@_submatrix_right_xshift_dim }
5766         { ( \l_@@_y_initial_dim + \l_@@_y_final_dim ) / 2 }
5767     }
5768     \str_if_empty:NTF \l_@@_submatrix_name_str
5769     { \@@_node_right:nn #4 { } }
5770     {
5771         \@@_node_right:nn #4 { \@@_env: - \l_@@_submatrix_name_str - right }
5772     }
5773     \endpgfpicture
5774     \cs_set_eq:NN \pgfpointanchor \@@_pgfpointanchor:n
5775     \flag_clear_new:n { nicematrix }
5776     \l_@@_code_tl
5777 }
5778 \group_end:
5779 }

```

The group was a group for the whole `\SubMatrix`.

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  $i-j$ , `row- $i$` , `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`.

```
5780 \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.

```
5781 \cs_new_protected:Npn \@@_pgfpntanchor:n #1
5782 {
5783   \use:e
5784   { \exp_not:N \@@_old_pgfpntanchor { \@@_pgfpntanchor_i:nn #1 } }
5785 }
```

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 -.

```
5786 \cs_new:Npn \@@_pgfpntanchor_i:nn #1 #2
5787 { #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.

```
5788 \tl_const:Nn \c_@@_integers_alist_tl
5789 {
5790   { 1 } { } { 2 } { } { 3 } { } { 4 } { } { 5 } { }
5791   { 6 } { } { 7 } { } { 8 } { } { 9 } { } { 10 } { }
5792   { 11 } { } { 12 } { } { 13 } { } { 14 } { } { 15 } { }
5793   { 16 } { } { 17 } { } { 18 } { } { 19 } { } { 20 } { }
5794 }
```

```
5795 \cs_new:Npn \@@_pgfpntanchor_ii:w #1-#2\q_stop
5796 {
```

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`.

```
5797   \tl_if_empty:nTF { #2 }
5798   {
5799     \str_case:nVTF { #1 } \c_@@_integers_alist_tl
5800     {
5801       \flag_raise:n { nicematrix }
5802       \int_if_even:nTF { \flag_height:n { nicematrix } }
5803       { \int_eval:n { #1 + \l_@@_first_i_tl - 1 } }
5804       { \int_eval:n { #1 + \l_@@_first_j_tl - 1 } }
5805     }
5806     { #1 }
5807   }
```

If there is an hyphen, we have to see whether we have a node of the form  $i-j$ , `row- $i$`  or `col- $j$` .

```
5808   { \@@_pgfpntanchor_iii:w { #1 } #2 }
5809 }
```

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. `\@@_pgfpointanchor_i:nn`).

```

5810 \cs_new:Npn \@@_pgfpointanchor_iii:w #1 #2 -
5811 {
5812   \str_case:nnF { #1 }
5813   {
5814     { row } { row - \int_eval:n { #2 + \l_@@_first_i_tl - 1 } }
5815     { col } { col - \int_eval:n { #2 + \l_@@_first_j_tl - 1 } }
5816   }

```

Now the case of a node of the form  $i-j$ .

```

5817 {
5818   \int_eval:n { #1 + \l_@@_first_i_tl - 1 }
5819   - \int_eval:n { #2 + \l_@@_first_j_tl - 1 }
5820 }
5821 }

```

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`).

```

5822 \cs_new_protected:Npn \@@_node_left:nn #1 #2
5823 {
5824   \pgfnode
5825   { rectangle }
5826   { east }
5827   {
5828     \nullfont
5829     \c_math_toggle_token
5830     \tl_if_empty:NF \l_@@_delimiters_color_tl
5831     { \color { \l_@@_delimiters_color_tl } }
5832     \left #1
5833     \vcenter
5834     {
5835       \nullfont
5836       \hrule \@height \l_tmpa_dim
5837       \@depth \c_zero_dim
5838       \@width \c_zero_dim
5839     }
5840     \right .
5841     \c_math_toggle_token
5842   }
5843   { #2 }
5844   { }
5845 }

```

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`).

```

5846 \cs_new_protected:Npn \@@_node_right:nn #1 #2
5847 {
5848   \pgfnode
5849   { rectangle }
5850   { west }
5851   {
5852     \nullfont
5853     \c_math_toggle_token
5854     \tl_if_empty:NF \l_@@_delimiters_color_tl
5855     { \color { \l_@@_delimiters_color_tl } }
5856     \left .
5857     \vcenter
5858     {
5859       \nullfont
5860       \hrule \@height \l_tmpa_dim

```

```

5861             \@depth \c_zero_dim
5862             \@width \c_zero_dim
5863         }
5864         \right #1
5865         \c_math_toggle_token
5866     }
5867     { #2 }
5868     { }
5869 }

```

## 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.

```

5870 \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.

```

5871 \bool_new:N \c_@@_footnote_bool
5872 \@@_msg_new:nnn { Unknown~option~for~package }
5873 {
5874     The~key~'\l_keys_key_str'~is~unknown.  \
5875     If~you~go~on,~it~will~be~ignored.  \
5876     For~a~list~of~the~available~keys,~type~H~<return>.
5877 }
5878 {
5879     The~available~keys~are~(in~alphabetic~order):~
5880     define-L-C-R,~
5881     footnote,~
5882     footnotehyper,~
5883     renew-dots,~and
5884     renew-matrix.
5885 }
5886 \@@_msg_new:nn { Key~transparent }
5887 {
5888     The~key~'transparent'~is~now~obsolete~(because~it's~name~
5889     is~not~clear).~You~should~use~the~conjunction~of~'renew-dots'~
5890     and~'renew-matrix'.~However,~you~can~go~on.
5891 }
5892 \keys_define:nn { NiceMatrix / Package }
5893 {
5894     define-L-C-R .bool_set:N = \c_@@_define_L_C_R_bool ,
5895     define-L-C-R .default:n = true ,
5896     renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
5897     renew-dots .value_forbidden:n = true ,
5898     renew-matrix .code:n = \@@_renew_matrix: ,
5899     renew-matrix .value_forbidden:n = true ,
5900     transparent .code:n =
5901     {
5902         \@@_renew_matrix:
5903         \bool_set_true:N \l_@@_renew_dots_bool
5904         \@@_error:n { Key~transparent }
5905     } ,
5906     transparent .value_forbidden:n = true,
5907     footnote .bool_set:N = \c_@@_footnote_bool ,

```



```

5908     footnotehyper .bool_set:N = \c_@@_footnotehyper_bool ,
5909     unknown .code:n = \@@_error:n { Unknown-option-for-package }
5910   }
5911   \ProcessKeysOptions { NiceMatrix / Package }

5912   \@@_msg_new:nn { footnote-with-footnotehyper-package }
5913   {
5914     You-can't-use-the-option-'footnote'~because-the-package~
5915     footnotehyper-has-already-been-loaded.~
5916     If-you-want,~you-can-use-the-option-'footnotehyper'~and-the-footnotes~
5917     within-the-environments-of-nicematrix-will-be-extracted-with-the-tools~
5918     of-the-package-footnotehyper.\\
5919     If-you-go-on,~the-package-footnote-won't-be-loaded.
5920   }

5921   \@@_msg_new:nn { footnotehyper-with-footnote-package }
5922   {
5923     You-can't-use-the-option-'footnotehyper'~because-the-package~
5924     footnote-has-already-been-loaded.~
5925     If-you-want,~you-can-use-the-option-'footnote'~and-the-footnotes~
5926     within-the-environments-of-nicematrix-will-be-extracted-with-the-tools~
5927     of-the-package-footnote.\\
5928     If-you-go-on,~the-package-footnotehyper-won't-be-loaded.
5929   }

5930   \bool_if:NT \c_@@_footnote_bool
5931   {

```

The class `beamer` has its own system to extract footnotes and that's why we have nothing to do if `beamer` is used.

```

5932     \@ifclassloaded { beamer }
5933     { \bool_set_false:N \c_@@_footnote_bool }
5934     {
5935       \@ifpackageloaded { footnotehyper }
5936       { \@@_error:n { footnote-with-footnotehyper-package } }
5937       { \usepackage { footnote } }
5938     }
5939   }

5940   \bool_if:NT \c_@@_footnotehyper_bool
5941   {

```

The class `beamer` has its own system to extract footnotes and that's why we have nothing to do if `beamer` is used.

```

5942     \@ifclassloaded { beamer }
5943     { \bool_set_false:N \c_@@_footnote_bool }
5944     {
5945       \@ifpackageloaded { footnote }
5946       { \@@_error:n { footnotehyper-with-footnote-package } }
5947       { \usepackage { footnotehyper } }
5948     }
5949     \bool_set_true:N \c_@@_footnote_bool
5950   }

```

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

```

5951 \seq_new:N \c_@@_types_of_matrix_seq
5952 \seq_set_from_clist:Nn \c_@@_types_of_matrix_seq
5953 {
5954   NiceMatrix ,
5955   pNiceMatrix , bNiceMatrix , vNiceMatrix, BNiceMatrix, VNiceMatrix
5956 }
5957 \seq_set_map_x:NNn \c_@@_types_of_matrix_seq \c_@@_types_of_matrix_seq
5958 { \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`.

```

5959 \cs_new_protected:Npn \@@_error_too_much_cols:
5960 {
5961   \seq_if_in:NVTF \c_@@_types_of_matrix_seq \g_@@_name_env_str
5962   {
5963     \int_compare:nNnTF \l_@@_last_col_int = { -2 }
5964     { \@@_fatal:n { too-much-cols-for-matrix } }
5965     {
5966       \bool_if:NF \l_@@_last_col_without_value_bool
5967       { \@@_fatal:n { too-much-cols-for-matrix-with-last-col } }
5968     }
5969   }
5970   { \@@_fatal:n { too-much-cols-for-array } }
5971 }

```

The following command must *not* be protected since it's used in an error message.

```

5972 \cs_new:Npn \@@_message_hdotsfor:
5973 {
5974   \tl_if_empty:VF \g_@@_HVdotsfor_lines_tl
5975   { ~Maybe-your-use-of~\token_to_str:N \Hdotsfor\ is-incorrect.}
5976 }
5977 \@@_msg_new:nn { too-much-cols-for-matrix-with-last-col }
5978 {
5979   You-try-to-use-more-columns-than-allowed-by-your~
5980   \@@_full_name_env:.\@@_message_hdotsfor:\ The-maximal-number-of~
5981   columns-is~\int_eval:n { \l_@@_last_col_int - 1 }~(plus-the~
5982   exterior-columns).~This-error-is-fatal.
5983 }
5984 \@@_msg_new:nn { too-much-cols-for-matrix }
5985 {
5986   You-try-to-use-more-columns-than-allowed-by-your~
5987   \@@_full_name_env:.\@@_message_hdotsfor:\ Recall-that-the-maximal~
5988   number-of-columns-for-a-matrix-is-fixed-by-the-LaTeX-counter~
5989   'MaxMatrixCols'.~Its-actual-value-is~\int_use:N \c@MaxMatrixCols.~
5990   This-error-is-fatal.
5991 }

```

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`.

```

5992 \@@_msg_new:nn { too-much-cols-for-array }
5993 {
5994   You-try-to-use-more-columns-than-allowed-by-your~
5995   \@@_full_name_env:.\@@_message_hdotsfor:\ The-maximal-number-of-columns-is~
5996   \int_use:N \g_@@_static_num_of_col_int\
5997   ~(plus-the-potential-exterior-ones).~
5998   This-error-is-fatal.
5999 }
6000 \@@_msg_new:nn { last-col-not-used }
6001 {

```

```

6002 The~key~'last~col'~is~in~force~but~you~have~not~used~that~last~column~
6003 in~your~\@@_full_name_env:.~However,~you~can~go~on.
6004 }

6005 \@@_msg_new:nn { columns~not~used }
6006 {
6007 The~preamble~of~your~\@@_full_name_env:\ announces~\int_use:N
6008 \g_@@_static_num_of_col_int\ columns~but~you~use~only~\int_use:N \c@jCol.\
6009 You~can~go~on~but~the~columns~you~did~not~used~won't~be~created.
6010 }

6011 \@@_msg_new:nn { in~first~col }
6012 {
6013 You~can't~use~the~command~#1 in~the~first~column~(number~0)~of~the~array.\
6014 If~you~go~on,~this~command~will~be~ignored.
6015 }

6016 \@@_msg_new:nn { in~last~col }
6017 {
6018 You~can't~use~the~command~#1 in~the~last~column~(exterior)~of~the~array.\
6019 If~you~go~on,~this~command~will~be~ignored.
6020 }

6021 \@@_msg_new:nn { in~first~row }
6022 {
6023 You~can't~use~the~command~#1 in~the~first~row~(number~0)~of~the~array.\
6024 If~you~go~on,~this~command~will~be~ignored.
6025 }

6026 \@@_msg_new:nn { in~last~row }
6027 {
6028 You~can't~use~the~command~#1 in~the~last~row~(exterior)~of~the~array.\
6029 If~you~go~on,~this~command~will~be~ignored.
6030 }

6031 \@@_msg_new:nn { bad~option~for~line~style }
6032 {
6033 Since~you~haven't~loaded~Tikz,~the~only~value~you~can~give~to~'line~style'~
6034 is~'standard'.~If~you~go~on,~this~key~will~be~ignored.
6035 }

6036 \@@_msg_new:nn { Unknown~key~for~xdots }
6037 {
6038 As~for~now,~there~is~only~three~key~available~here:~'color',~'line~style'~
6039 and~'shorten'~(and~you~try~to~use~'\l_keys_key_str').~If~you~go~on,~
6040 this~key~will~be~ignored.
6041 }

6042 \@@_msg_new:nn { Unknown~key~for~rowcolors }
6043 {
6044 As~for~now,~there~is~only~two~keys~available~here:~'cols'~and~'respect~blocks'~
6045 (and~you~try~to~use~'\l_keys_key_str').~If~you~go~on,~
6046 this~key~will~be~ignored.
6047 }

6048 \@@_msg_new:nn { ampersand~in~light~syntax }
6049 {
6050 You~can't~use~an~ampersand~(\token_to_str:N &)~to~separate~columns~because~
6051 ~you~have~used~the~key~'light~syntax'.~This~error~is~fatal.
6052 }

6053 \@@_msg_new:nn { SubMatrix~too~large }
6054 {
6055 Your~command~\token_to_str:N \SubMatrix\
6056 can't~be~drawn~because~your~matrix~is~too~small.\
6057 If~you~go~on,~this~command~will~be~ignored.
6058 }

6059 \@@_msg_new:nn { double~backslash~in~light~syntax }
6060 {

```

```

6061     You~can't~use~\token_to_str:N \~-to~separate~rows~because~you~have~used~
6062     the~key~'light-syntax'.~You~must~use~the~character~'\l_@@_end_of_row_tl'~
6063     (set~by~the~key~'end-of-row').~This~error~is~fatal.
6064 }

6065 \@@_msg_new:nn { standard-cline~in~document }
6066 {
6067     The~key~'standard-cline'~is~available~only~in~the~preamble.\\
6068     If~you~go~on~this~command~will~be~ignored.
6069 }

6070 \@@_msg_new:nn { old~column~type }
6071 {
6072     The~column~type~'#1'~is~no~longer~defined~in~'nicematrix'.~
6073     Since~version~5.0,~you~have~to~use~'l',~'c'~and~'r'~instead~of~'L',~
6074     'C'~and~'R'.~You~can~also~use~the~key~'define-L-C-R'.\\
6075     This~error~is~fatal.
6076 }

6077 \@@_msg_new:nn { bad~value~for~baseline }
6078 {
6079     The~value~given~to~'baseline'~(\int_use:N \l_tmpa_int)~is~not~
6080     valid.~The~value~must~be~between~\int_use:N \l_@@_first_row_int\ and~
6081     \int_use:N \g_@@_row_total_int\ or~equal~to~'t',~'c'~or~'b'.\\
6082     If~you~go~on,~a~value~of~1~will~be~used.
6083 }

6084 \@@_msg_new:nn { Invalid~name~format }
6085 {
6086     You~can't~give~the~name~'\l_keys_value_tl'~to~a~\token_to_str:N
6087     \SubMatrix.\\
6088     A~name~must~be~accepted~by~the~regular~expression~[A-Za-z][A-Za-z0-9]*.\\
6089     If~you~go~on,~this~key~will~be~ignored.
6090 }

6091 \@@_msg_new:nn { Wrong~line~in~SubMatrix }
6092 {
6093     You~try~to~draw~a~#1~line~of~number~'#2'~in~a~
6094     \token_to_str:N \SubMatrix\ of~your~\@@_full_name_env:\ but~that~
6095     number~is~not~valid.~If~you~go~on,~it~will~be~ignored.
6096 }

6097 \@@_msg_new:nn { empty~environment }
6098 { Your~\@@_full_name_env:\ is~empty.~This~error~is~fatal. }

6099 \@@_msg_new:nn { Delimiter~with~small }
6100 {
6101     You~can't~put~a~delimiter~in~the~preamble~of~your~\@@_full_name_env:\
6102     because~the~key~'small'~is~in~force.\\
6103     This~error~is~fatal.
6104 }

6105 \@@_msg_new:nn { unknown~cell~for~line~in~CodeAfter }
6106 {
6107     Your~command~\token_to_str:N\line\{#1\}\{#2\}~in~the~'code~after'~
6108     can't~be~executed~because~a~cell~doesn't~exist.\\
6109     If~you~go~on~this~command~will~be~ignored.
6110 }

6111 \@@_msg_new:nnn { Duplicate~name~for~SubMatrix }
6112 {
6113     The~name~'#1'~is~already~used~for~a~\token_to_str:N \SubMatrix\
6114     in~this~\@@_full_name_env:.\
6115     If~you~go~on,~this~key~will~be~ignored.\\
6116     For~a~list~of~the~names~already~used,~type~H~<return>.
6117 }
6118 {
6119     The~names~already~defined~in~this~\@@_full_name_env:\ are:~
6120     \seq_use:Nnnn \g_@@_submatrix_names_seq { ~and~ } { ,~ } { ~and~ }.

```

```

6121     }
6122 \@@_msg_new:nn { r~or~l~with~preamble }
6123 {
6124     You~can't~use~the~key~'\l_keys_key_str'~in~your~\@@_full_name_env:~.~
6125     You~must~specify~the~alignment~of~your~columns~with~the~preamble~of~
6126     your~\@@_full_name_env:~.~\
6127     If~you~go~on,~this~key~will~be~ignored.
6128 }
6129 \@@_msg_new:nn { Hdotsfor~in~col~0 }
6130 {
6131     You~can't~use~\token_to_str:N \Hdotsfor\ in~an~exterior~column~of~
6132     the~array.~This~error~is~fatal.
6133 }
6134 \@@_msg_new:nn { bad~corner }
6135 {
6136     #1~is~an~incorrect~specification~for~a~corner~(in~the~keys~
6137     'corners'~and~'except-corners').~The~available~
6138     values~are:~NW,~SW,~NE~and~SE.~\
6139     If~you~go~on,~this~specification~of~corner~will~be~ignored.
6140 }
6141 \@@_msg_new:nn { bad~border }
6142 {
6143     #1~is~an~incorrect~specification~for~a~border~(in~the~key~
6144     'borders'~of~the~command~\token_to_str:N \Block).~The~available~
6145     values~are:~left,~right,~top~and~bottom.~\
6146     If~you~go~on,~this~specification~of~border~will~be~ignored.
6147 }
6148 \@@_msg_new:nn { last~col~non~empty~for~NiceArray }
6149 {
6150     In~the~\@@_full_name_env:,~you~must~use~the~key~
6151     'last-col'~without~value.~\
6152     However,~you~can~go~on~for~this~time~
6153     (the~value~'\l_keys_value_tl'~will~be~ignored).
6154 }
6155 \@@_msg_new:nn { last~col~non~empty~for~NiceMatrixOptions }
6156 {
6157     In~\NiceMatrixoptions,~you~must~use~the~key~
6158     'last-col'~without~value.~\
6159     However,~you~can~go~on~for~this~time~
6160     (the~value~'\l_keys_value_tl'~will~be~ignored).
6161 }
6162 \@@_msg_new:nn { Block~too~large~1 }
6163 {
6164     You~try~to~draw~a~block~in~the~cell~#1~#2~of~your~matrix~but~the~matrix~is~
6165     too~small~for~that~block.~\
6166 }
6167 \@@_msg_new:nn { Block~too~large~2 }
6168 {
6169     The~preamble~of~your~\@@_full_name_env:\ announces~\int_use:N
6170     \g_@@_static_num_of_col_int\
6171     columns~but~you~use~only~\int_use:N \c@jCol\ and~that's~why~a~block~
6172     specified~in~the~cell~#1~#2~can't~be~drawn.~You~should~add~some~ampersands~
6173     (&)~at~the~end~of~the~first~row~of~your~
6174     \@@_full_name_env:~.~\
6175     If~you~go~on,~this~block~and~maybe~others~will~be~ignored.
6176 }
6177 \@@_msg_new:nn { unknown~column~type }
6178 {
6179     The~column~type~'#1'~in~your~\@@_full_name_env:\
6180     is~unknown.~\

```

```

6181   This~error~is~fatal.
6182 }

6183 \@@_msg_new:nn { tabularnote~forbidden }
6184 {
6185   You~can't~use~the~command~\token_to_str:N\tabularnote\
6186   ~in~a~\@@_full_name_env:~.~This~command~is~available~only~in~
6187   \{NiceTabular\},~\{NiceArray\}~and~\{NiceMatrix\}. \\
6188   If~you~go~on,~this~command~will~be~ignored.
6189 }

6190 \@@_msg_new:nn { borders~forbidden }
6191 {
6192   You~can't~use~the~key~'borders'~of~the~command~\token_to_str:N \Block\
6193   because~the~option~'rounded~corners'~
6194   is~in~force~with~a~non~zero~value. \\
6195   If~you~go~on,~this~key~will~be~ignored.
6196 }

6197 \@@_msg_new:nn { bottomrule~without~booktabs }
6198 {
6199   You~can't~use~the~key~'tabular/bottomrule'~because~you~haven't~
6200   loaded~'booktabs'. \\
6201   If~you~go~on,~this~key~will~be~ignored.
6202 }

6203 \@@_msg_new:nn { enumitem~not~loaded }
6204 {
6205   You~can't~use~the~command~\token_to_str:N\tabularnote\
6206   ~because~you~haven't~loaded~'enumitem'. \\
6207   If~you~go~on,~this~command~will~be~ignored.
6208 }

6209 \@@_msg_new:nn { Wrong~last~row }
6210 {
6211   You~have~used~'last~row=\int_use:N \l_@@_last_row_int'~but~your~
6212   \@@_full_name_env:\ seems~to~have~\int_use:N \c@iRow \ rows.~
6213   If~you~go~on,~the~value~of~\int_use:N \c@iRow \ will~be~used~for~
6214   last~row.~You~can~avoid~this~problem~by~using~'last~row'~
6215   without~value~(more~compilations~might~be~necessary).
6216 }

6217 \@@_msg_new:nn { Yet~in~env }
6218 { Environments~of~nicematrix~can't~be~nested. \\ This~error~is~fatal. }

6219 \@@_msg_new:nn { Outside~math~mode }
6220 {
6221   The~\@@_full_name_env:\ can~be~used~only~in~math~mode~
6222   (and~not~in~\token_to_str:N \vcenter). \\
6223   This~error~is~fatal.
6224 }

6225 \@@_msg_new:nn { One~letter~allowed }
6226 {
6227   The~value~of~key~'\l_keys_key_str'~must~be~of~length~1. \\
6228   If~you~go~on,~it~will~be~ignored.
6229 }

6230 \@@_msg_new:nnn { Unknown~key~for~Block }
6231 {
6232   The~key~'\l_keys_key_str'~is~unknown~for~the~command~\token_to_str:N
6233   \Block. \\ If~you~go~on,~it~will~be~ignored. \\
6234   For~a~list~of~the~available~keys,~type~H<return>.
6235 }
6236 {
6237   The~available~keys~are~(in~alphabetic~order):~borders,~c,~draw,~fill,~l,~
6238   line~width,~rounded~corners~and~r.
6239 }

```

```

6240 \@@_msg_new:nnn { Unknown-key-for-CodeAfter }
6241 {
6242   The~key~'\l_keys_key_str'~is-unknown.\\
6243   If~you~go~on,~it~will~be~ignored. \\
6244   For~a~list~of~the~available~keys~in~\token_to_str:N
6245   \CodeAfter,~type~H~<return>.
6246 }
6247 {
6248   The~available~keys~are~(in~alphabetic~order):~
6249   delimiters/color,~
6250   rules~(with~the~subkeys~'color'~and~'width'),~
6251   sub-matrix~(several~subkeys)~
6252   and~xdots~(several~subkeys).~
6253   The~latter~is~for~the~command~\token_to_str:N \line.
6254 }
6255 \@@_msg_new:nnn { Unknown-key-for-SubMatrix }
6256 {
6257   The~key~'\l_keys_key_str'~is-unknown.\\
6258   If~you~go~on,~this~key~will~be~ignored. \\
6259   For~a~list~of~the~available~keys~in~\token_to_str:N
6260   \SubMatrix,~type~H~<return>.
6261 }
6262 {
6263   The~available~keys~are~(in~alphabetic~order):~
6264   'delimiters/color',~
6265   'extra-height',~
6266   'hlines',~
6267   'hvlines',~
6268   'left-xshift',~
6269   'name',~
6270   'right-xshift',~
6271   'rules'~(with~the~subkeys~'color'~and~'width'),~
6272   'slim',~
6273   'vlines'~and~'xshift'~(which~sets~both~'left-xshift'~
6274   and~'right-xshift').\\
6275 }
6276 \@@_msg_new:nnn { Unknown-key-for-notes }
6277 {
6278   The~key~'\l_keys_key_str'~is-unknown.\\
6279   If~you~go~on,~it~will~be~ignored. \\
6280   For~a~list~of~the~available~keys~about~notes,~type~H~<return>.
6281 }
6282 {
6283   The~available~keys~are~(in~alphabetic~order):~
6284   bottomrule,~
6285   code-after,~
6286   code-before,~
6287   enumitem-keys,~
6288   enumitem-keys-para,~
6289   para,~
6290   label-in-list,~
6291   label-in-tabular~and~
6292   style.
6293 }
6294 \@@_msg_new:nnn { Unknown-key-for-NiceMatrixOptions }
6295 {
6296   The~key~'\l_keys_key_str'~is-unknown~for~the~command~
6297   \token_to_str:N \NiceMatrixOptions. \\
6298   If~you~go~on,~it~will~be~ignored. \\
6299   For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
6300 }
6301 {
6302   The~available~keys~are~(in~alphabetic~order):~

```

```

6303     allow-duplicate-names,~
6304     cell-space-bottom-limit,~
6305     cell-space-limits,~
6306     cell-space-top-limit,~
6307     code-for-first-col,~
6308     code-for-first-row,~
6309     code-for-last-col,~
6310     code-for-last-row,~
6311     corners,~
6312     create-extra-nodes,~
6313     create-medium-nodes,~
6314     create-large-nodes,~
6315     delimiters/color,~
6316     end-of-row,~
6317     first-col,~
6318     first-row,~
6319     hlines,~
6320     hvlines,~
6321     last-col,~
6322     last-row,~
6323     left-margin,~
6324     letter-for-dotted-lines,~
6325     light-syntax,~
6326     notes~(several~subkeys),~
6327     nullify-dots,~
6328     renew-dots,~
6329     renew-matrix,~
6330     right-margin,~
6331     rules~(with~the~subkeys~'color'~and~'width'),~
6332     small,~
6333     sub-matrix~(several~subkeys),
6334     vlines,~
6335     xdots~(several~subkeys).
6336 }

6337 \@@_msg_new:nnn { Unknown~option~for~NiceArray }
6338 {
6339     The~key~'\l_keys_key_str'~is~unknown~for~the~environment~
6340     \{NiceArray\}. \\
6341     If~you~go~on,~it~will~be~ignored. \\
6342     For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
6343 }
6344 {
6345     The~available~keys~are~(in~alphabetic~order):~
6346     b,~
6347     baseline,~
6348     c,~
6349     cell-space-bottom-limit,~
6350     cell-space-limits,~
6351     cell-space-top-limit,~
6352     code-after,~
6353     code-for-first-col,~
6354     code-for-first-row,~
6355     code-for-last-col,~
6356     code-for-last-row,~
6357     colortbl-like,~
6358     columns-width,~
6359     corners,~
6360     create-extra-nodes,~
6361     create-medium-nodes,~
6362     create-large-nodes,~
6363     delimiters/color,~
6364     extra-left-margin,~
6365     extra-right-margin,~

```



```

6366 first-col,~
6367 first-row,~
6368 hlines,~
6369 hvlines,~
6370 last-col,~
6371 last-row,~
6372 left-margin,~
6373 light-syntax,~
6374 name,~
6375 notes/bottomrule,~
6376 notes/para,~
6377 nullify-dots,~
6378 renew-dots,~
6379 right-margin,~
6380 rules~(with~the~subkeys~'color'~and~'width'),~
6381 small,~
6382 t,~
6383 tabularnote,~
6384 vlines,~
6385 xdots/color,~
6386 xdots/shorten~and~
6387 xdots/line-style.
6388 }

```

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).

```

6389 \@@_msg_new:nnn { Unknown~option~for~NiceMatrix }
6390 {
6391   The~key~'\l_keys_key_str'~is~unknown~for~the~
6392   \@@_full_name_env:. \\\
6393   If~you~go~on,~it~will~be~ignored. \\\
6394   For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
6395 }
6396 {
6397   The~available~keys~are~(in~alphabetic~order):~
6398   b,~
6399   baseline,~
6400   c,~
6401   cell-space-bottom-limit,~
6402   cell-space-limits,~
6403   cell-space-top-limit,~
6404   code-after,~
6405   code-for-first-col,~
6406   code-for-first-row,~
6407   code-for-last-col,~
6408   code-for-last-row,~
6409   colortbl-like,~
6410   columns-width,~
6411   corners,~
6412   create-extra-nodes,~
6413   create-medium-nodes,~
6414   create-large-nodes,~
6415   delimiters/color,~
6416   extra-left-margin,~
6417   extra-right-margin,~
6418   first-col,~
6419   first-row,~
6420   hlines,~
6421   hvlines,~
6422   l,~
6423   last-col,~
6424   last-row,~
6425   left-margin,~
6426   light-syntax,~

```

```

6427     name,~
6428     nullify-dots,~
6429     r,~
6430     renew-dots,~
6431     right-margin,~
6432     rules~(with~the~subkeys~'color'~and~'width'),~
6433     small,~
6434     t,~
6435     vl原因,~
6436     xdots/color,~
6437     xdots/shorten~and~
6438     xdots/line-style.
6439 }
6440 \@@_msg_new:nnn { Unknown~option~for~NiceTabular }
6441 {
6442     The~key~'\l_keys_key_str'~is~unknown~for~the~environment~
6443     \{NiceTabular\}. \\
6444     If~you~go~on,~it~will~be~ignored. \\
6445     For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
6446 }
6447 {
6448     The~available~keys~are~(in~alphabetic~order):~
6449     b,~
6450     baseline,~
6451     c,~
6452     cell-space-bottom-limit,~
6453     cell-space-limits,~
6454     cell-space-top-limit,~
6455     code-after,~
6456     code-for-first-col,~
6457     code-for-first-row,~
6458     code-for-last-col,~
6459     code-for-last-row,~
6460     colortbl-like,~
6461     columns-width,~
6462     corners,~
6463     create-extra-nodes,~
6464     create-medium-nodes,~
6465     create-large-nodes,~
6466     extra-left-margin,~
6467     extra-right-margin,~
6468     first-col,~
6469     first-row,~
6470     hlines,~
6471     hvlines,~
6472     last-col,~
6473     last-row,~
6474     left-margin,~
6475     light-syntax,~
6476     name,~
6477     notes/bottomrule,~
6478     notes/para,~
6479     nullify-dots,~
6480     renew-dots,~
6481     right-margin,~
6482     rules~(with~the~subkeys~'color'~and~'width'),~
6483     t,~
6484     tabularnote,~
6485     vl原因,~
6486     xdots/color,~
6487     xdots/shorten~and~
6488     xdots/line-style.
6489 }

```

```

6490 \@@_msg_new:nnn { Duplicate-name }
6491 {
6492   The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
6493   the~same~environment~name~twice.~You~can~go~on,~but,~
6494   maybe,~you~will~have~incorrect~results~especially~
6495   if~you~use~'columns-width=auto'.~If~you~don't~want~to~see~this~
6496   message~again,~use~the~key~'allow-duplicate-names'~in~
6497   '\token_to_str:N \NiceMatrixOptions'.\\
6498   For~a~list~of~the~names~already~used,~type~H~<return>. \\
6499 }
6500 {
6501   The~names~already~defined~in~this~document~are:~
6502   \seq_use:Nnnn \g_@@_names_seq { ~and~ } { ,~ } { ~and~ }.
6503 }
6504 \@@_msg_new:nn { Option-auto-for-columns-width }
6505 {
6506   You~can't~give~the~value~'auto'~to~the~key~'columns-width'~here.~
6507   If~you~go~on,~the~key~will~be~ignored.
6508 }

```

## 18 History

The successive versions of the file `nicematrix.sty` provided by TeXLive are available on the SVN server of TeXLive:

<https://www.tug.org/svn/texlive/trunk/Master/texmf-dist/tex/latex/nicematrix/nicematrix.sty>

### Changes between versions 1.0 and 1.1

The dotted lines are no longer drawn with Tikz nodes but with Tikz circles (for efficiency).  
Modification of the code which is now twice faster.

### Changes between versions 1.1 and 1.2

New environment `{NiceArray}` with column types L, C and R.

### Changes between version 1.2 and 1.3

New environment `{pNiceArrayC}` and its variants.

Correction of a bug in the definition of `{BNiceMatrix}`, `{vNiceMatrix}` and `{VNiceMatrix}` (in fact, it was a typo).

Options are now available locally in `{pNiceMatrix}` and its variants.

The names of the options are changed. The old names were names in “camel style”.

### Changes between version 1.3 and 1.4

The column types `w` and `W` can now be used in the environments `{NiceArray}`, `{pNiceArrayC}` and its variants with the same meaning as in the package `array`.

New option `columns-width` to fix the same width for all the columns of the array.

### Changes between version 1.4 and 2.0

The versions 1.0 to 1.4 of `nicematrix` were focused on the continuous dotted lines whereas the version 2.0 of `nicematrix` provides different features to improve the typesetting of mathematical matrices.

## Changes between version 2.0 and 2.1

New implementation of the environment `{pNiceArrayRC}`. With this new implementation, there is no restriction on the width of the columns.

The package `nicematrix` no longer loads `mathtools` but only `amsmath`.

Creation of “medium nodes” and “large nodes”.

## Changes between version 2.1 and 2.1.1

Small corrections: for example, the option `code-for-first-row` is now available in the command `\NiceMatrixOptions`.

Following a discussion on TeX StackExchange<sup>60</sup>, Tikz externalization is now deactivated in the environments of the package `nicematrix`.<sup>61</sup>

## Changes between version 2.1.2 and 2.1.3

When searching the end of a dotted line from a command like `\Cdots` issued in the “main matrix” (not in the exterior column), the cells in the exterior column are considered as outside the matrix. That means that it’s possible to do the following matrix with only a `\Cdots` command (and a single `\Vdots`).

$$\begin{pmatrix} & C_j & \\ 0 & \vdots & 0 \\ & a \cdots \cdots & \\ 0 & & 0 \end{pmatrix}_{L_i}$$

## 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`.

---

<sup>60</sup>cf. [tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package](https://tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package)

<sup>61</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 2.2.1 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>62</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.

---

<sup>62</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 3.5 and 3.6

LaTeX counters `iRow` and `jCol` available in the cells of the array.

Addition of `\normalbaselines` before the construction of the array: in environments like `{align}` of `amsmath` the value of `\baselineskip` is changed and if the options `first-row` and `last-row` were used in an environment of `nicematrix`, the position of the delimiters was wrong.

A warning is written in the `.log` file if an obsolete environment is used.

There is no longer artificial errors `Duplicate~name` in the environments of `amsmath`.

## Changes between version 3.6 and 3.7

The four “corners” of the matrix are correctly protected against the four codes: `code-for-first-col`, `code-for-last-col`, `code-for-first-row` and `code-for-last-row`.

New command `\pAutoNiceMatrix` and its variants (suggestion of Christophe Bal).

## Changes between version 3.7 and 3.8

New programming for the command `\Block` when the block has only one row. With this programming, the vertical rules drawn by the specifier “|” at the end of the block is actually drawn. In previous versions, they were not because the block of one row was constructed with `\multicolumn`. An error is raised when an obsolete environment is used.

## Changes between version 3.8 and 3.9

New commands `\NiceMatrixLastEnv` and `\OnlyMainNiceMatrix`.

New options `create-medium-nodes` and `create-large-nodes`.

## Changes between version 3.9 and 3.10

New option `light-syntax` (and `end-of-row`).

New option `dotted-lines-margin` for fine tuning of the dotted lines.

## Changes between versions 3.10 and 3.11

Correction of a bug linked to `first-row` and `last-row`.

## Changes between versions 3.11 and 3.12

Command `\rotate` in the cells of the array.

Options `vlines`, `hlines` and `hvlines`.

Option `baseline` pour `{NiceArray}` (not for the other environments).

The name of the Tikz nodes created by the command `\Block` has changed: when the command has been issued in the cell  $i-j$ , the name is  $i-j$ -`block` and, if the creation of the “medium nodes” is required, a node  $i-j$ -`block-medium` is created.

If the user tries to use more columns than allowed by its environment, an error is raised by `nicematrix` (instead of a low-level error).

The package must be loaded with the option `obsolete-environments` if we want to use the deprecated environments.

## Changes between versions 3.12 and 3.13

The behaviour of the command `\rotate` is improved when used in the “last row”.

The option `dotted-lines-margin` has been renamed in `xdots/shorten` and the options `xdots/color` and `xdots/line-style` have been added for a complete customisation of the dotted lines.

In the environments without preamble (`{NiceMatrix}`, `{pNiceMatrix}`, etc.), it’s possible to use the options `l` (`=L`) or `r` (`=R`) to specify the type of the columns.

The starred versions of the commands `\Cdots`, `\Ldots`, `\Vdots`, `\Ddots` and `\Iddots` are deprecated since the version 3.1 of `nicematrix`. Now, one should load `nicematrix` with the option `starred-commands` to avoid an error at the compilation.

The code of `nicematrix` no longer uses Tikz but only PGF. By default, Tikz is *not* loaded by `nicematrix`.

## Changes between versions 3.13 and 3.14

Correction of a bug (question 60761504 on [stackoverflow](#)).

Better error messages when the user uses `&` or `\\` when `light-syntax` is in force.

## Changes between versions 3.14 and 3.15

It’s possible to put labels on the dotted lines drawn by `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots`, `\Hdotsfor` and the command `\line` in the `code-after` with the tokens `_` and `^`.

The option `baseline` is now available in all the environments of `nicematrix`. Before, it was available only in `{NiceArray}`.

New keyword `\CodeAfter` (in the environments of `nicematrix`).

## Changes between versions 3.15 and 4.0

New environment `{NiceTabular}`

Commands to color cells, rows and columns with a perfect result in the PDF.

## Changes between versions 4.0 and 4.1

New keys `cell-space-top-limit` and `cell-space-bottom-limit`

New command `\diagbox`

The key `hvline` don’t draw rules in the blocks (commands `\Block`) and in the virtual blocks corresponding to the dotted lines.

## Changes between versions 4.1 and 4.2

It’s now possible to write `\begin{pNiceMatrix}a&b\\c&d\end{pNiceMatrix}`<sup>2</sup> with the expected result.

## Changes between versions 4.2 and 4.3

The horizontal centering of the content of a `\Block` is correct even when an instruction such as `!\quad` 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 forme  $(1.5)$ ,  $(2.5)$ ,  $(3.5)$ , etc.

Keys `t` and `b` for the command `\Block`.

Key `corners`.

# 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:		<code>\@@_Cell:</code> ..... 201, 829, 1595, 1642, 1664, 2265, 3383, 3384, 3385, 3386, 3387, 3388, 3389, 3390, 3391, 3392, 3393, 3394
<code>\@@_Block:</code> ..... 1151, 4743		<code>\@@_CodeAfter:</code> ..... 1155, 5456
<code>\@@_Block_i</code> ..... 4745, 4748		<code>\@@_CodeAfter_i:n</code> ..... 831, 2143, 2188, 5456, 5457, 5467
<code>\@@_Block_ii:nnnnn</code> ..... 4748, 4749		<code>\@@_CodeAfter_ii:n</code> ..... 5460, 5462
<code>\@@_Block_iv:nnnnn</code> ..... 4779, 4783		<code>\@@_CodeAfter_keys:</code> ..... 1234, 2424, 2446
<code>\@@_Block_iv:nnnnnn</code> ..... 4961, 4963		<code>\@@_Ddots</code> ..... 1078, 1143, 3315
<code>\@@_Block_v:nnnnn</code> ..... 4780, 4884		<code>\g_@@_Ddots_lines_tl</code> ..... 1171, 2483
<code>\@@_Block_v:nnnnnn</code> ..... 4990, 4993		<code>\g_@@_HVdotsfor_lines_tl</code> .....
<code>\@@_Cdots</code> ..... 1076, 1141, 3283		
<code>\g_@@_Cdots_lines_tl</code> ..... 1168, 2485		

..... 1173, 2481, 3442, 3518, 5974	\l_@@_auto_columns_width_bool .....
\@@_Hdotsfor: ..... 1081, 1148, 3418	..... 468, 604, 2053, 2057, 4504
\@@_Hdotsfor:nnnn ..... 3444, 3456	\l_@@_baseline_tl 459, 460, 597, 598, 599,
\@@_Hdotsfor_i ..... 3427, 3433, 3440	600, 1010, 1403, 1754, 1766, 1771, 1773,
\@@_Hline: ..... 1146, 4251	1778, 1783, 1865, 1866, 1870, 1875, 1877, 1882
\@@_Hline_i:n ..... 4251, 4252, 4258	\@@_begin_of_NiceMatrix:nn .... 2254, 2275
\@@_Hline_ii:nn ..... 4255, 4258	\@@_begin_of_row: ..... 834, 857, 2145
\@@_Hline_iii:n ..... 4256, 4259	\l_@@_block_auto_columns_width_bool ..
\@@_Hspace: ..... 1147, 3369	..... 1347, 2058, 4497, 4502, 4512, 4522
\@@_Iddots ..... 1079, 1144, 3339	\g_@@_block_box_int ..... 309, 1327,
\g_@@_Iddots_lines_tl ..... 1172, 2484	4785, 4799, 4801, 4837, 4849, 4861, 4870, 4880
\@@_Ldots ..... 1075, 1080, 1140, 3267	\g_@@_blocks_dp_dim .....
\g_@@_Ldots_lines_tl ..... 1169, 2486	..... 241, 915, 918, 919, 4864, 4867
\l_@@_Matrix_bool .... 247, 1458, 1474, 2256	\g_@@_blocks_ht_dim .....
\l_@@_NiceArray_bool .....	..... 240, 921, 924, 925, 4855, 4858
. 244, 365, 1290, 1359, 1401, 1512, 1524,	\g_@@_blocks_seq .....
2232, 4120, 4121, 4243, 4244, 4450, 4457, 5374	..... 287, 1349, 1804, 4874, 4886, 4961
\g_@@_NiceMatrixBlock_int .....	\g_@@_blocks_wd_dim .....
..... 235, 4509, 4514, 4517, 4528	..... 239, 909, 912, 913, 4843, 4846
\l_@@_NiceTabular_bool ..... 156,	\c_@@_booktabs_loaded_bool 25, 34, 1091, 1836
245, 836, 1002, 1232, 1236, 1334, 1434,	\l_@@_borders_clist ..... 300, 4942,
1438, 1513, 1525, 2285, 2294, 4811, 4893, 5382	5010, 5250, 5266, 5268, 5270, 5272, 5291, 5303
\@@_NotEmpty: ..... 1157, 2279	\@@_cartesian_path: .....
\@@_OnlyMainNiceMatrix:n ..... 1153, 3981	..... 3687, 3702, 3858, 3870, 3905
\@@_OnlyMainNiceMatrix_i:n 3984, 3991, 3994	\@@_cartesian_path:n ..... 3732, 3905, 3906
\@@_SubMatrix ..... 2406, 5599	\l_@@_cell_box 835, 881, 883, 889, 895, 898,
\@@_SubMatrix_in_code_before ... 1231, 5586	902, 911, 912, 917, 918, 923, 924, 934, 935,
\@@_Vdots ..... 1077, 1142, 3299	936, 937, 939, 942, 946, 948, 967, 1093,
\g_@@_Vdots_lines_tl ..... 1170, 2482	1245, 1247, 1663, 1674, 2146, 2170, 2173,
\@@_Vdotsfor: ..... 1149, 3516	2175, 2192, 2215, 2219, 5043, 5147, 5181, 5389
\@@_Vdotsfor:nnnn ..... 3520, 3531	\l_@@_cell_space_bottom_limit_dim ...
\@@_W: ..... 1477, 1553	..... 448, 516, 937
\@@_actually_color: ..... 1235, 3656	\l_@@_cell_space_top_limit_dim 447, 514, 935
\@@_actually_diagbox:nnnnnn ..... 4790, 5035, 5394, 5410	\l_@@_cell_type_tl .....
\@@_actually_draw_Cdots: ..... 2839, 2843	..... 237, 238, 1595, 1665, 4763, 4765
\@@_actually_draw_Ddots: ..... 2989, 2993	\@@_cellcolor .. 1223, 3734, 3746, 3747, 3787
\@@_actually_draw_Iddots: ..... 3041, 3045	\@@_cellcolor_tabular ..... 1085, 3952
\@@_actually_draw_Ldots: . 2800, 2804, 3507	\g_@@_cells_seq ..... 1980, 1981, 1982, 1984
\@@_actually_draw_Vdots: . 2921, 2925, 3582	\@@_chessboardcolors ..... 1230, 3739
\@@_adapt_S_column: ..... 171, 186, 1333	\@@_cline ..... 135, 1139
\@@_add_to_colors_seq:nn .....	\@@_cline_i:nn ..... 136, 137, 149, 152
..... 3643, 3655, 3678, 3693, 3708, 3717	\@@_cline_i:w ..... 137, 138
\@@_adjust_pos_of_blocks_seq: .. 2393, 2448	\l_@@_code_before_bool .....
\@@_adjust_pos_of_blocks_seq_i:nnnn ..	.. 277, 594, 621, 1017, 1179, 1315, 1355,
..... 2451, 2453	2000, 2017, 2035, 2066, 2092, 2119, 2344, 2440
\@@_adjust_size_box: .....	\l_@@_code_before_tl .....
..... 907, 933, 1673, 2167, 2212	..... 276, 593, 1234, 1314, 1356
\@@_adjust_to_submatrix:nn .....	\l_@@_code_for_first_col_tl ..... 533, 2157
..... 2684, 2787, 2826, 2907, 2982, 3034	\l_@@_code_for_first_row_tl . 537, 845, 5119
\@@_adjust_to_submatrix:nnnnnn . 2691, 2693	\l_@@_code_for_last_col_tl ..... 535, 2201
\@@_after_array: ..... 1467, 2298	\l_@@_code_for_last_row_tl . 539, 852, 5122
\g_@@_after_col_zero_bool .....	\l_@@_code_tl ..... 268, 5581, 5776
..... 273, 1043, 2144, 3424	\l_@@_col_max_int .....
\@@_analyze_end:Nn ..... 1941, 1986	..... 295, 2551, 2562, 2630, 2689, 2706
\l_@@_argspec_tl ..... 3265,	\l_@@_col_min_int .....
3266, 3267, 3283, 3299, 3315, 3339, 3438,	..... 294, 2556, 2619, 2624, 2687, 2704
3439, 3440, 3514, 3515, 3516, 3592, 3593, 3594	\g_@@_col_total_int .....
\@@_array: ..... 997, 1942, 1969	950,
\@@_arraycolor ..... 1228, 3757	1164, 1387, 2085, 2086, 2122, 2126, 2131,
\@@_arraycolor_code_after:nnn .. 3768, 3775	2132, 2191, 2302, 2305, 2310, 2317, 2361,
\c_@@_arydshln_loaded_bool ... 24, 31, 1581	2874, 2892, 2952, 3414, 3415, 3577, 3971,
	4555, 4565, 4599, 4686, 4973, 5154, 5614, 5650
	\l_@@_color_tl ..... 302, 4740, 4803, 4805
	\g_@@_colors_seq 1233, 3646, 3650, 3651, 3660

\@@_colortbl_like: . . . . .	1083, 1158	\@@_draw_Cdots:nnn . . . . .	2824
\l_@@_colortbl_like_bool 445, 620, 1158, 1500		\@@_draw_Ddots:nnn . . . . .	2980
\c_@@_colortbl_loaded_bool . . . 88, 92, 1108		\@@_draw_Iddots:nnn . . . . .	3032
\l_@@_cols_tl . . . . .		\@@_draw_Ldots:nnn . . . . .	2785
. . . 3686, 3701, 3731, 3797, 3805, 3806, 3908		\@@_draw_Vdots:nnn . . . . .	2905
\g_@@_cols_vlism_seq . . 256, 1495, 1572, 5678		\@@_draw_blocks: . . . . .	1804, 4958
\@@_columncolor . . . . .	1229, 3689	\@@_draw_dotted_lines: . . . . .	2391, 2470
\@@_columncolor:n . . . . .	3695, 3698	\@@_draw_dotted_lines_i: . . . . .	2473, 2477
\@@_columncolor_preamble . . . . .	1087, 3969	\l_@@_draw_first_bool . 308, 3330, 3354, 3365	
\c_@@_columncolor_regex . . . . .	210, 1503	\@@_draw_hlines: . . . . .	2404, 4240
\l_@@_columns_width_dim . . . . .		\@@_draw_line: . . . . .	2822,
. . . . . 236, 605, 740, 2054, 2060, 4510, 4516		. . . . . 2867, 2978, 3030, 3078, 3080, 3641, 4465, 4495	
\g_@@_com_or_env_str . . . . .	260, 263	\@@_draw_line_ii:nn . . . . .	3621, 3625
\@@_computations_for_large_nodes: . . .		\@@_draw_line_iii:nn . . . . .	3628, 3632
. . . . . 4626, 4639, 4644		\@@_draw_non_standard_dotted_line: . .	
\@@_computations_for_medium_nodes: . .		. . . . . 3086, 3088	
. . . . . 4546, 4615, 4625, 4636		\@@_draw_non_standard_dotted_line:n . .	
\@@_compute_a_corner:nnnnnn . . . . .		. . . . . 3091, 3094	
. . . . . 4322, 4324, 4326, 4328, 4333		\@@_draw_non_standard_dotted_line:nnn	
\@@_compute_corners: . . . . .	2392, 3779, 4314	. . . . . 3096, 3101, 3115	
\@@_construct_preamble: . . . . .	1303, 1471	\@@_draw_standard_dotted_line: . 3085, 3116	
\l_@@_corners_clist . . . . .		\@@_draw_standard_dotted_line_i: 3179, 3183	
. . . . . 464, 582, 587, 3778, 4021, 4147, 4317		\l_@@_draw_tl . . . . .	299, 4936,
\@@_create_col_nodes: . . . . .	1945, 1973, 1992	. . . . . 4940, 4998, 5189, 5195, 5197, 5199, 5236, 5237	
\@@_create_diag_nodes: . . . 1213, 2369, 2503		\@@_draw_vlines: . . . . .	2405, 4117
\@@_create_extra_nodes: . . . . .	1803, 4536	\g_@@_empty_cell_bool . . . 284, 941, 951,	
\@@_create_large_nodes: . . . . .	4544, 4620	. . . . . 2180, 2227, 3281, 3297, 3313, 3337, 3360, 3371	
\@@_create_medium_and_large_nodes: . .		\l_@@_empty_corner_cells_seq 2398, 3784,	
. . . . . 4541, 4631		. . . . . 4055, 4061, 4068, 4180, 4186, 4193, 4316, 4389	
\@@_create_medium_nodes: . . . . .	4542, 4610	\@@_end_Cell: . . . . .	203, 928, 1597,
\@@_create_nodes: 4617, 4628, 4638, 4641, 4682		. . . . . 1652, 1669, 2265, 3383, 3384, 3385, 3386,	
\@@_create_row_node: . . . . .	1013, 1046, 1092	. . . . . 3387, 3388, 3389, 3390, 3391, 3392, 3393, 3394	
\@@_cut_on_hyphen:w . . . . .	3669,	\l_@@_end_of_row_tl . . . . .	
. . . . . 3724, 3729, 3826, 3912, 3913, 3934, 3935,		. . . . . 476, 477, 527, 1965, 1966, 6062	
. . . . . 5207, 5216, 5255, 5258, 5588, 5593, 5606, 5609		\c_@@_endpgfortikzpicture_tl . . . . .	
\g_@@_ddots_int . . . . .	2372, 3013, 3014	. . . . . 43, 47, 2474, 3629, 4436	
\@@_def_env:nnn . . . . .		\c_@@_enumitem_loaded_bool . . . . .	
. . . . . 2238, 2249, 2250, 2251, 2252, 2253		. . . . . 26, 37, 338, 648, 653, 664, 669	
\@@_define_L_C_R: . . . . .	223, 1302	\@@_env: . . . . .	230, 234, 866, 872, 968, 974,
\c_@@_define_L_C_R_bool . . . 222, 1302, 5894		. . . . . 1022, 1028, 1034, 1193, 1194, 1200, 1201,	
\@@_define_com:nnn . . . . .		. . . . . 1208, 1209, 1220, 2001, 2004, 2006, 2022,	
. . . . . 5358, 5366, 5367, 5368, 5369, 5370		. . . . . 2028, 2031, 2040, 2046, 2049, 2071, 2077,	
\@@_delimiter:nnn . . . . .	1693, 1701, 5470	. . . . . 2080, 2097, 2103, 2109, 2122, 2126, 2132,	
\l_@@_delimiters_color_tl 479, 718, 720,		. . . . . 2415, 2520, 2527, 2529, 2530, 2591, 2659,	
. . . . . 774, 776, 793, 795, 1426, 1427, 1444, 1445,		. . . . . 2723, 2734, 2747, 2750, 2769, 2772, 2877,	
. . . . . 5450, 5508, 5509, 5536, 5830, 5831, 5854, 5855		. . . . . 2880, 2895, 2898, 3470, 3488, 3545, 3563,	
\l_@@_delimiters_max_width_bool . . . . .		. . . . . 3614, 3616, 3635, 3638, 4344, 4363, 4381,	
. . . . . 480, 579, 717, 1449		. . . . . 4568, 4570, 4578, 4689, 4698, 4716, 5058,	
\g_@@_delta_x_one_dim . . . . .	2374, 3016, 3026	. . . . . 5065, 5069, 5083, 5088, 5100, 5105, 5106,	
\g_@@_delta_x_two_dim . . . . .	2376, 3064, 3074	. . . . . 5109, 5126, 5160, 5485, 5488, 5630, 5632,	
\g_@@_delta_y_one_dim . . . . .	2375, 3018, 3026	. . . . . 5637, 5639, 5653, 5655, 5660, 5662, 5760, 5771	
\g_@@_delta_y_two_dim . . . . .	2377, 3066, 3074	\g_@@_env_int . . 229, 230, 232, 1181, 1185,	
\@@_diagbox:nn . . . . .	1156, 5390	. . . . . 1188, 1197, 1198, 1205, 1206, 1255, 1258,	
\@@_dotfill: . . . . .	5379	. . . . . 1273, 1276, 1346, 1353, 1357, 2309, 2330,	
\@@_dotfill_i: . . . . .	5384, 5386	. . . . . 2348, 2351, 2364, 2436, 3813, 3816, 3841, 4725	
\@@_dotfill_ii: . . . . .	5383, 5386, 5387	\@@_error:n . . . . .	12, 341,
\@@_dotfill_iii: . . . . .	5387, 5388	. . . . . 366, 489, 499, 549, 674, 723, 733, 739, 748,	
\@@_double_int_eval:n . . . . .	3588, 3602, 3603	. . . . . 756, 778, 785, 797, 798, 799, 805, 810, 811,	
\g_@@_dp_ante_last_row_dim . . . . .	860, 1124	. . . . . 812, 823, 825, 826, 827, 1382, 1392, 1461,	
\g_@@_dp_last_row_dim . . . . .		. . . . . 1788, 1841, 1887, 3800, 4956, 5248, 5454,	
. . . . . 860, 861, 1127, 1128, 1246, 1247, 1420		. . . . . 5566, 5577, 5584, 5615, 5904, 5909, 5936, 5946	
\g_@@_dp_row_zero_dim . . . . .		\@@_error:nn . . . . .	
. . . . . 880, 881, 1118, 1119, 1413, 1859, 1898		. . . . . 13, 612, 1561, 1562, 1563, 3270, 3273,	

3286, 3289, 3302, 3305, 3319, 3320, 3325,	4826, 4877, 4900, 4904, 4916, 4921, 4946,
3326, 3343, 3344, 3349, 3350, 4330, 5253, 5571	4948, 4950, 5128, 5140, 5151, 5155, 5162, 5174
\@@_error:nnn ..... 14, 3619, 5707, 5742	\g_@@_ht_last_row_dim ..... 862, 1125, 1126, 1244, 1245, 1419
\@@_error_too_much_cols: ..... 1534, 5959	\g_@@_ht_row_one_dim .. 888, 889, 1122, 1123
\@@_everycr: ..... 1039, 1113, 1116	\g_@@_ht_row_zero_dim ..... 882, 883, 1120, 1121, 1414, 1858, 1897
\@@_everycr_i: ..... 1039, 1040	\@@_i: ..... 4548, 4550,
\l_@@_exterior_arraycolsep_bool ..... 461, 736, 1515, 1527	4551, 4552, 4553, 4562, 4568, 4570, 4571,
\l_@@_extra_left_margin_dim ..... 474, 571, 1306, 2178	4572, 4573, 4578, 4579, 4580, 4581, 4589,
\l_@@_extra_right_margin_dim ..... 475, 572, 1374, 2223, 2955	4592, 4594, 4595, 4596, 4648, 4650, 4653,
\@@_extract_coords_values: ..... 4707, 4714	4654, 4658, 4659, 4684, 4689, 4691, 4693,
\@@_fatal:n .... 15, 252, 1337, 1689, 1699,	4697, 4698, 4709, 4716, 4718, 4720, 4724, 4725
1950, 1954, 1956, 1989, 3429, 5964, 5967, 5970	\g_@@_iddots_int ..... 2373, 3061, 3062
\@@_fatal:nn ..... 16, 1586	\l_@@_in_env_bool .... 243, 365, 1337, 1338
\l_@@_fill_tl ..... 298, 4934, 5020, 5025	\c_@@_in_preamble_bool . 21, 22, 23, 644, 660
\l_@@_final_i_int ..... 2381, 2538, 2543, 2546, 2571,	\l_@@_initial_i_int ..... 2379,
2579, 2583, 2592, 2600, 2680, 2735, 2816,	2536, 2611, 2614, 2639, 2647, 2651, 2660,
2889, 2895, 2898, 3461, 3489, 3557, 3567, 3569	2668, 2678, 2724, 2809, 2855, 2857, 2871,
\l_@@_final_j_int ..... 2382, 2539, 2544, 2551, 2556, 2562, 2572,	2877, 2880, 3460, 3461, 3471, 3539, 3549, 3551
2580, 2584, 2593, 2601, 2681, 2736, 2769,	\l_@@_initial_j_int ..... 2380, 2537, 2612, 2619,
2772, 2780, 3006, 3482, 3492, 3494, 3536, 3565	2624, 2630, 2640, 2648, 2652, 2661, 2669,
\l_@@_final_open_bool ..... 2384, 2545,	2679, 2725, 2747, 2750, 2758, 2945, 2947,
2549, 2552, 2559, 2565, 2569, 2585, 2813,	2952, 2998, 3464, 3474, 3476, 3535, 3536, 3547
2848, 2853, 2864, 2928, 2938, 2943, 2964,	\l_@@_initial_open_bool ..... 2383, 2613, 2617, 2620, 2627, 2633,
3003, 3053, 3187, 3202, 3233, 3234, 3459,	2637, 2653, 2806, 2845, 2852, 2862, 2928,
3483, 3495, 3534, 3558, 3570, 3611, 4433, 4470	2935, 2941, 2995, 3047, 3185, 3232, 3458,
\@@_find_extremities_of_line:nnnn ... 2533, 2790, 2829, 2910, 2985, 3037	3465, 3477, 3533, 3540, 3552, 3610, 4432, 4469
\l_@@_first_col_int ..... 123, 136, 319,	\@@_insert_tabularnotes: ..... 1808, 1811
320, 529, 803, 834, 1396, 1507, 1995, 2015,	\@@_instruction_of_type:nnn ..... 978, 3275, 3291, 3307, 3330, 3354
2355, 2874, 2892, 3422, 3925, 3983, 4555,	\c_@@_integers_alist_tl ..... 5788, 5799
4565, 4599, 4647, 4686, 5340, 5346, 5352, 5650	\l_@@_inter_dots_dim ..... 449, 450, 2388, 3190, 3197, 3208, 3216,
\l_@@_first_i_tl ..... 5602, 5607, 5626, 5644,	3223, 3228, 3240, 3248, 4461, 4463, 4491, 4493
5653, 5655, 5710, 5717, 5719, 5803, 5814, 5818	\g_@@_internal_code_after_tl ..... 269, 1631, 1692, 1700, 1720, 2420,
\l_@@_first_j_tl ..... 5603, 5608, 5630,	2421, 3766, 4266, 4425, 4788, 5033, 5392, 5596
5632, 5680, 5693, 5700, 5702, 5804, 5815, 5819	\@@_intersect_our_row:nnnn ..... 3894
\l_@@_first_row_int ..... 317, 318,	\@@_intersect_our_row_p:nnnn ..... 3845
530, 807, 1162, 1411, 1785, 1856, 1884,	\@@_j: ..... 4555, 4557,
1895, 2353, 2744, 2766, 4548, 4562, 4589,	4558, 4559, 4560, 4565, 4568, 4570, 4573,
4646, 4684, 5062, 5080, 5338, 5482, 5627, 6080	4575, 4576, 4578, 4581, 4583, 4584, 4599,
\c_@@_footnote_bool ..... 1322, 1469, 5871, 5907, 5930, 5933, 5943, 5949	4602, 4604, 4605, 4606, 4661, 4663, 4666,
\c_@@_footnotehyper_bool . 5870, 5908, 5940	4668, 4672, 4673, 4686, 4689, 4690, 4692,
\@@_full_name_env: ..... 261, 5980, 5987, 5995, 6003, 6007,	4697, 4698, 4710, 4716, 4717, 4719, 4724, 4725
6094, 6098, 6101, 6114, 6119, 6124, 6126,	\l_@@_l_dim ..... 3163, 3164, 3177, 3178, 3190, 3196,
6150, 6169, 6174, 6179, 6186, 6212, 6221, 6392	3207, 3215, 3223, 3228, 3240, 3241, 3248, 3249
\@@_hdottedline: ..... 1145, 4418	\l_@@_large_nodes_bool 471, 562, 4540, 4544
\@@_hdottedline:n ..... 4426, 4430	\g_@@_last_col_found_bool .. 327, 1167,
\@@_hdottedline_i: ..... 4421, 4423	1388, 1453, 2084, 2113, 2189, 2301, 2358, 5152
\@@_hdottedline_i:n ..... 4435, 4439	\l_@@_last_col_int ..... 325, 326, 724, 767, 769, 786, 806, 824,
\@@_hline:nn ..... 4128, 4248, 4267	1191, 1269, 1275, 1282, 1391, 1519, 2261,
\@@_hline_i:nn ..... 2402, 4131, 4134	2263, 2302, 2305, 2357, 2915, 2950, 3272,
\@@_hline_i_complete:nn ..... 2402, 4238	3288, 3326, 3350, 4966, 4971, 4972, 4973,
\@@_hline_ii:nnnn ... 4156, 4167, 4200, 4239	4976, 5005, 5017, 5027, 5039, 5054, 5083,
\l_@@_hlines_clist ..... 313, 541,	5088, 5096, 5111, 5342, 5348, 5354, 5963, 5981
555, 584, 1047, 1049, 1053, 2404, 4246, 4247	\l_@@_last_col_without_value_bool ... 324, 766, 2303, 5966
\l_@@_hpos_of_block_tl .. 304, 305, 4730,	
4732, 4734, 4764, 4765, 4767, 4810, 4816,	

<code>\l_@@_last_empty_column_int</code> .....	1280, 1283, 2005, 2006, 2030, 2031, 2048,
..... 4354, 4355, 4368, 4374, 4387	2049, 2079, 2080, 2105, 2108, 2128, 2131,
<code>\l_@@_last_empty_row_int</code> .....	2312, 2316, 2333, 2337, 4694, 4697, 4721, 4724
..... 4336, 4337, 4350, 4371	<code>\g_@@_names_seq</code> .....
<code>\l_@@_last_i_tl</code> .....	242, 611, 613, 6502
5610, 5613, 5626, 5647, 5660, 5662, 5710, 5717	<code>\l_@@_nb_cols_int</code> .....
<code>\l_@@_last_j_tl</code> .....	..... 5329, 5334, 5337, 5341, 5347, 5353
5605, 5611, 5614, 5637, 5639, 5683, 5693, 5700	<code>\l_@@_nb_rows_int</code> .....
<code>\l_@@_last_row_int</code> .....	5328, 5333, 5344
.... 321, 322, 531, 850, 896, 1059, 1189,	<code>\@@_newcolumnntype</code> .....
1240, 1250, 1257, 1264, 1376, 1380, 1383,	1066, 1476, 1477
1395, 1417, 1967, 1968, 2153, 2154, 2198,	<code>\@@_node_for_multicolumn:nn</code> ....
2199, 2324, 2795, 2834, 3304, 3320, 3344,	4705, 4712
3989, 3997, 4965, 4968, 4969, 4988, 5005,	<code>\@@_node_for_the_cell:</code> ..
5017, 5027, 5038, 5052, 5110, 5121, 5350, 6211	947, 954, 2174, 2224
<code>\l_@@_last_row_without_value_bool</code> ...	<code>\@@_node_left:nn</code> .....
..... 323, 1252, 1378, 2322	5759, 5760, 5822
<code>\l_@@_left_delim_dim</code> .....	<code>\@@_node_position:</code> ..
..... 1288, 1292, 1297, 1933, 2176	1200, 1202, 1208, 1210
<code>\l_@@_left_delim_tl</code> ..	<code>\@@_node_right:nn</code> .....
1296, 1324, 1428, 4460	5769, 5771, 5846
<code>\l_@@_left_margin_dim</code> .....	<code>\g_@@_not_empty_cell_bool</code> ..
..... 472, 565, 1305, 2177, 4451, 4677	275, 945, 952, 2280
<code>\l_@@_letter_for_dotted_lines_str</code> ...	<code>\@@_not_in_exterior:nnnn</code> .....
..... 747, 758, 759, 1567	3886
<code>\l_@@_letter_vlism_tl</code> .....	<code>\@@_not_in_exterior_p:nnnn</code> .....
255, 548, 1570	3818
<code>\l_@@_light_syntax_bool</code> .....	<code>\l_@@_notes_above_space_dim</code> ....
..... 458, 525, 1308, 1369, 2325	465, 466
<code>\@@_light_syntax_i</code> .....	<code>\l_@@_notes_bottomrule_bool</code> .....
1958, 1961	..... 632, 789, 818, 1834
<code>\@@_line</code> .....	<code>\l_@@_notes_code_after_tl</code> .....
2419, 3594	630, 1843
<code>\@@_line_i:nn</code> .....	<code>\l_@@_notes_code_before_tl</code> .....
3601, 3608	628, 1815
<code>\l_@@_line_width_dim</code> .....	<code>\@@_notes_label_in_list:n</code> ..
..... 303, 4944, 5190, 5228, 5239,	334, 353, 361, 640
5245, 5265, 5280, 5282, 5292, 5293, 5295, 5306	<code>\@@_notes_label_in_tabular:n</code> ..
<code>\@@_line_with_light_syntax:n</code> ...	333, 374, 637
1972, 1976	<code>\l_@@_notes_para_bool</code> ..
<code>\@@_line_with_light_syntax_i:n</code> .....	626, 787, 816, 1819
..... 1971, 1977, 1978	<code>\@@_notes_style:n</code> .....
<code>\@@_math_toggle_token:</code> .....	..... 332, 335, 353, 361, 377, 382, 634
155, 930, 2147, 2164, 2193, 2209, 5434, 5438	<code>\l_@@_nullify_dots_bool</code> .....
<code>\g_@@_max_cell_width_dim</code> .....	..... 467, 560, 3279, 3295, 3311, 3335, 3358
..... 938, 939, 1348, 2059, 4503, 4529	<code>\l_@@_number_of_notes_int</code> ..
<code>\c_@@_max_l_dim</code> .....	331, 368, 378, 388
3177, 3182	<code>\@@_old_CT@arc@</code> .....
<code>\l_@@_medium_nodes_bool</code> ..	1339, 2444
470, 561, 4538, 5102	<code>\@@_old_cdots</code> .....
<code>\@@_message_hdotsfor:</code> ..	1133, 3296
5972, 5980, 5987, 5995	<code>\@@_old_ddots</code> .....
<code>\@@_msg_new:nn</code> .....	1135, 3336
17, 5886,	<code>\@@_old_dotfill</code> .....
5912, 5921, 5977, 5984, 5992, 6000, 6005,	5378, 5381, 5389
6011, 6016, 6021, 6026, 6031, 6036, 6042,	<code>\@@_old_dotfill:</code> .....
6048, 6053, 6059, 6065, 6070, 6077, 6084,	1154
6091, 6097, 6099, 6105, 6122, 6129, 6134,	<code>\l_@@_old_iRow_int</code> .....
6141, 6148, 6155, 6162, 6167, 6177, 6183,	270, 1095, 2490
6190, 6197, 6203, 6209, 6217, 6219, 6225, 6504	<code>\@@_old_ialign:</code> .....
<code>\@@_msg_new:nnn</code> ...	1012, 1129, 4960
18, 5872, 6111, 6230,	<code>\@@_old_iddots</code> .....
6240, 6255, 6276, 6294, 6337, 6389, 6440, 6490	1136, 3359
<code>\@@_msg_redirect_name:nn</code> .....	<code>\l_@@_old_jCol_int</code> .....
..... 19, 742, 1465, 4979, 4982	271, 1098, 2491
<code>\@@_multicolumn:nnn</code> .....	<code>\@@_old_ldots</code> .....
1150, 3375	1132, 3280
<code>\g_@@_multicolumn_cells_seq</code> .....	<code>\@@_old_multicolumn</code> .....
.... 1160, 3402, 4573, 4581, 4703, 5067, 5085	3374, 3378
<code>\g_@@_multicolumn_sizes_seq</code> ..	<code>\@@_old_pgfpoinanchor</code> ....
1161, 3404, 4704	163, 5780, 5784
<code>\g_@@_name_env_str</code> .....	<code>\@@_old_pgful@check@rerun</code> .....
259,	81, 85
264, 265, 1331, 1332, 1988, 2233, 2234,	<code>\@@_old_vdots</code> .....
2242, 2243, 2272, 2283, 2291, 2442, 5362, 5961	1134, 3312
<code>\l_@@_name_str</code> ..	<code>\@@_open_x_final_dim:</code> .....
469, 614, 868, 871, 970,	..... 2763, 2815, 2849, 3007, 3056
973, 1030, 1033, 1253, 1262, 1265, 1271,	<code>\@@_open_x_initial_dim:</code> .....
	..... 2741, 2808, 2846, 3000, 3050
	<code>\@@_open_y_final_dim:</code> ..
	2887, 2939, 3005, 3055
	<code>\@@_open_y_initial_dim:</code> .....
	..... 2869, 2936, 2997, 3049
	<code>\l_@@_parallelize_diags_bool</code> .....
	..... 462, 463, 557, 2370, 3011, 3059
	<code>\@@_patch_preamble:n</code> .....
	1497, 1538,
	1576, 1584, 1605, 1634, 1694, 1714, 1722, 1742
	<code>\@@_patch_preamble_i:n</code> ..
	1542, 1543, 1544, 1591
	<code>\@@_patch_preamble_ii:nn</code> .....
	..... 1545, 1546, 1547, 1602
	<code>\@@_patch_preamble_iii:n</code> ..
	1548, 1607, 1615
	<code>\@@_patch_preamble_iii_i:n</code> ....
	1610, 1612
	<code>\@@_patch_preamble_iv:nnn</code> .....
	..... 1549, 1550, 1551, 1637
	<code>\@@_patch_preamble_ix:n</code> .....
	1727, 1745
	<code>\@@_patch_preamble_v:nnnn</code> ..
	1552, 1553, 1658



\@@_patch_preamble_vii:n	1554, 1680	\l_@@_respect_blocks_bool	3795, 3810, 3838
\@@_patch_preamble_viii:n	1555, 1556, 1557, 1686	\@@_restore_iRow_jCol:	2443, 2488
\@@_patch_preamble_viii:n	1558, 1559, 1560, 1696	\@@_revtex_array:	989, 1000
\@@_patch_preamble_viii_i:n	1702, 1704	\c_@@_revtex_bool	50, 52, 55, 57, 999
\@@_patch_preamble_x:n	1600, 1656, 1678, 1684, 1724, 1748	\l_@@_right_delim_dim	1289, 1293, 1299, 1936, 2221
\@@_patch_preamble_xi:n	1568, 1716	\l_@@_right_delim_tl	1298, 1325, 1446, 4462
\@@_pgf_rect_node:nnn	421, 5104	\l_@@_right_margin_dim	473, 567, 1373, 2222, 2954, 4458, 4680
\@@_pgf_rect_node:nnnnn	396, 4688, 4715, 5057, 5099, 5746	\@@_rotate:	1152, 3587
\c_@@_pgfortikzpicture_tl	42, 46, 2472, 3627, 4434	\g_@@_rotate_bool	248, 905, 932, 1672, 2166, 2211, 3587, 4810, 4834, 4839, 4899, 4915, 5044
\@@_pgfpointanchor:n	5774, 5781	\@@_rotate_cell_box:	893, 932, 1672, 2166, 2211, 5044
\@@_pgfpointanchor_i:nn	5784, 5786	\l_@@_rounded_corners_dim	301, 4938, 5028, 5204, 5205, 5240, 5247, 5304
\@@_pgfpointanchor_ii:w	5787, 5795	\@@_roundedrectanglecolor	1225, 3713
\@@_pgfpointanchor_iii:w	5808, 5810	\l_@@_row_max_int	293, 2546, 2688, 2705
\@@_picture_position:	1194, 1202, 1210	\l_@@_row_min_int	292, 2614, 2686, 2703
\g_@@_pos_of_blocks_seq	288, 1350, 2365, 2396, 2450, 3405, 4015, 4141, 4401, 4995, 5402	\g_@@_row_of_col_done_bool	274, 1044, 1330, 2014
\g_@@_pos_of_stroken_blocks_seq	290, 1351, 4019, 4145, 5007	\g_@@_row_total_int	1163, 1394, 1786, 1885, 2324, 2331, 2338, 2354, 2744, 2766, 3502, 4548, 4562, 4589, 4684, 4988, 5062, 5080, 5482, 5613, 5627, 6081
\g_@@_pos_of_xdots_seq	289, 1352, 2397, 2676, 4017, 4143	\@@_rowcolor	1226, 3674
\@@_pre_array:	1176, 1316, 1366	\@@_rowcolor:n	3680, 3683
\@@_pre_array_i:w	1312, 1366	\@@_rowcolor_tabular	1086, 3960
\@@_pre_array_ii:	1089, 1287	\@@_rowcolors	1227, 3802
\c_@@_preamble_first_col_tl	1508, 2139	\@@_rowcolors_i:nnnn	3846, 3881
\c_@@_preamble_last_col_tl	1520, 2184	\l_@@_rowcolors_restart_bool	3798, 3829
\g_@@_preamble_tl	1326, 1478, 1482, 1485, 1491, 1505, 1508, 1517, 1520, 1529, 1533, 1574, 1583, 1593, 1604, 1617, 1639, 1660, 1682, 1691, 1713, 1718, 1731, 1738, 1747, 1942, 1969	\g_@@_rows_seq	1964, 1966, 1968, 1970, 1972
\@@_pred:n	124, 154, 2263, 4056, 4069, 4181, 4194	\l_@@_rows_tl	3685, 3700, 3730, 3848, 3930
\@@_provide_pgfsyspdfmark:	211, 220, 1321	\l_@@_rules_color_tl	272, 503, 1363, 1364, 5675, 5676
\@@_put_box_in_flow:	1451, 1750, 1935	\@@_set_CT@arc@:	157, 1364, 5676
\@@_put_box_in_flow_bis:nn	1450, 1902	\@@_set_CT@arc@_i:	158, 159
\@@_put_box_in_flow_i:	1756, 1758	\@@_set_CT@arc@_ii:	158, 161
\@@_qpoint:n	233, 1761, 1763, 1775, 1791, 1850, 1852, 1868, 1879, 1890, 2509, 2511, 2513, 2515, 2523, 2525, 2758, 2780, 2809, 2816, 2855, 2857, 2871, 2889, 2945, 2947, 2998, 3006, 3635, 3638, 3924, 3928, 3941, 3943, 4079, 4081, 4083, 4204, 4206, 4208, 4442, 4446, 4453, 4487, 4490, 4492, 4594, 4604, 5048, 5050, 5052, 5054, 5076, 5096, 5124, 5212, 5214, 5221, 5223, 5279, 5281, 5283, 5290, 5294, 5296, 5415, 5417, 5420, 5422, 5475, 5477, 5644, 5647, 5685, 5702, 5719	\@@_set_final_coords:	2714, 2739
\l_@@_radius_dim	453, 454, 1719, 2387, 2820, 2821, 3257, 4420, 4444, 4488, 4489	\@@_set_final_coords_from_anchor:n	2730, 2819, 2850, 2931, 2940, 3010, 3058
\l_@@_real_left_delim_dim	1904, 1919, 1934	\@@_set_initial_coords:	2709, 2728
\l_@@_real_right_delim_dim	1905, 1931, 1937	\@@_set_initial_coords_from_anchor:n	2719, 2812, 2847, 2930, 2937, 3002, 3052
\@@_rectanglecolor	1224, 3704, 3737, 3762	\@@_set_size:n	5326, 5335
\@@_rectanglecolor:nnn	3710, 3719, 3722	\c_@@_siunitx_loaded_bool	164, 168, 173, 191
\@@_renew_NC@rewrite@S:	192, 194, 1166	\l_@@_small_bool	725, 772, 782, 808, 839, 1101, 1688, 1698, 2148, 2194, 2385
\@@_renew_dots:	1073, 1159	\@@_standard_cline	120, 1138
\l_@@_renew_dots_bool	558, 732, 1159, 5896, 5903	\@@_standard_cline:w	120, 121
\@@_renew_matrix:	727, 731, 5308, 5898, 5902	\l_@@_standard_cline_bool	446, 512, 1137
		\c_@@_standard_tl	456, 457, 3084, 4464, 4494
		\g_@@_static_num_of_col_int	297, 1460, 1498, 4976, 5996, 6008, 6170
		\l_@@_stop_loop_bool	2540, 2541, 2573, 2586, 2595, 2608, 2609, 2641, 2654, 2663
		\@@_stroke_block:nnn	5002, 5186
		\@@_stroke_borders_block:nnn	5014, 5243
		\@@_stroke_horizontal:n	5271, 5273, 5288
		\@@_stroke_vertical:n	5267, 5269, 5277
		\l_@@_submatrix_extra_height_dim	310, 5528, 5670

<code>\l_@@_submatrix_hlines_clist</code> .....	<code>\@@_vline_i_complete:nn</code> .....
..... 315, 5540, 5558, 5709, 5711	..... 2401, 4115
<code>\l_@@_submatrix_left_xshift_dim</code> .....	<code>\@@_vline_ii:nnnn</code> ... 4031, 4042, 4075, 4116
..... 311, 5530, 5722, 5755	<code>\l_@@_vlines_clist</code> .....
<code>\l_@@_submatrix_name_str</code> .....	..... 314, 542, 554, 583, 1483,
5573, 5617, 5744, 5746, 5758, 5760, 5768, 5771	1489, 1514, 1526, 1729, 1736, 2405, 4123, 4124
<code>\g_@@_submatrix_names_seq</code> .....	<code>\l_@@_vpos_of_block_tl</code> .....
..... 291, 2423, 5570, 5574, 6120	..... 306, 307,
<code>\l_@@_submatrix_right_xshift_dim</code> .....	4736, 4738, 4815, 4825, 4903, 4920, 4952, 4954
..... 312, 5532, 5731, 5765	<code>\@@_w:</code> .....
<code>\g_@@_submatrix_seq</code> ... 296, 1178, 2690, 5594	..... 1476, 1552
<code>\l_@@_submatrix_slim_bool</code> .....	<code>\g_@@_width_first_col_dim</code> .....
5538, 5625	..... 286, 1329, 1399, 2009, 2169, 2170
<code>\l_@@_submatrix_vlines_clist</code> .....	<code>\g_@@_width_last_col_dim</code> .....
..... 316, 5542, 5560, 5692, 5694	..... 285, 1328, 1455, 2118, 2214, 2215
<code>\@@_succ:n</code> .....	<code>\l_@@_x_final_dim</code> .....
. 149, 153, 1022, 1028, 1054, 1632, 1693,	..... 280,
1736, 1763, 2097, 2103, 2108, 2109, 2122,	2716, 2765, 2774, 2775, 2778, 2781, 2782,
2126, 2131, 2132, 2359, 2780, 2857, 2947,	2933, 2949, 2957, 2961, 2965, 2967, 2972,
3006, 3890, 3928, 3941, 4052, 4083, 4121,	2974, 3008, 3017, 3025, 3065, 3073, 3112,
4177, 4208, 4244, 4267, 4406, 4408, 4410,	3127, 3136, 3170, 3222, 3238, 3639, 4454,
4412, 4453, 4492, 4654, 4658, 4668, 4672,	4463, 4489, 5624, 5640, 5641, 5731, 5748, 5765
5052, 5054, 5096, 5221, 5223, 5420, 5422, 5477	<code>\l_@@_x_initial_dim</code> .....
<code>\l_@@_suffix_tl</code> .....	..... 278, 2711, 2743, 2752, 2753, 2756,
4616, 4627,	2759, 2760, 2933, 2948, 2949, 2956, 2961,
4637, 4640, 4689, 4697, 4698, 4716, 4724, 4725	2965, 2967, 2969, 2972, 2974, 2999, 3017,
<code>\c_@@_table_collect_begin_tl</code> . 181, 183, 201	3025, 3065, 3073, 3109, 3126, 3136, 3170,
<code>\c_@@_table_print_tl</code> .....	3222, 3236, 3238, 3256, 3258, 3636, 4447,
184, 185, 203	4461, 4488, 5623, 5633, 5634, 5722, 5747, 5755
<code>\l_@@_tabular_width_dim</code> .....	<code>\l_@@_xdots_color_tl</code> 478, 492, 2799, 2838,
..... 246, 1005, 1007, 1531, 2292	2919, 2920, 2988, 3040, 3092, 3506, 3581, 3598
<code>\l_@@_tabularnote_tl</code> 330, 791, 820, 1807, 1816	<code>\l_@@_xdots_down_tl</code> ... 496, 3099, 3120, 3155
<code>\g_@@_tabularnotes_seq</code> .....	<code>\l_@@_xdots_line_style_tl</code> .....
..... 329, 369, 1822, 1828, 1844	..... 455, 457, 488, 3084, 3092, 4464, 4494
<code>\@@_test_hline_in_block:nnnn</code> .....	<code>\l_@@_xdots_shorten_dim</code> .....
..... 4142, 4144, 4270	..... 451,
<code>\@@_test_hline_in_stroken_block:nnnn</code> .	452, 494, 2389, 3106, 3107, 3196, 3207, 3215
..... 4146, 4292	<code>\l_@@_xdots_up_tl</code> ... 497, 3098, 3119, 3145
<code>\@@_test_if_cell_in_a_block:nn</code> .....	<code>\l_@@_y_final_dim</code> .....
..... 4340, 4358, 4376, 4396	..... 281, 2717, 2817, 2821, 2859, 2863,
<code>\@@_test_if_cell_in_block:nnnnnnn</code> ...	2865, 2890, 2900, 2901, 3019, 3022, 3067,
..... 4402, 4404	3070, 3112, 3127, 3135, 3172, 3227, 3246,
<code>\@@_test_if_math_mode:</code> .... 249, 1336, 2244	3640, 4445, 4493, 5478, 5500, 5515, 5648,
<code>\@@_test_in_corner_h:</code> .....	5663, 5664, 5669, 5687, 5704, 5748, 5756, 5766
4147, 4175	<code>\l_@@_y_initial_dim</code> .....
<code>\@@_test_in_corner_v:</code> .....	..... 279, 2712, 2810, 2820, 2858,
4022, 4050	2859, 2863, 2865, 2872, 2882, 2883, 3019,
<code>\@@_test_vline_in_block:nnnn</code> .....	3024, 3067, 3072, 3109, 3126, 3135, 3172,
..... 4016, 4018, 4281	3227, 3244, 3246, 3256, 3259, 3637, 4443,
<code>\@@_test_vline_in_stroken_block:nnnn</code> .	4444, 4445, 4491, 5476, 5500, 5515, 5645,
..... 4020, 4303	5656, 5657, 5669, 5686, 5703, 5747, 5756, 5766
<code>\l_@@_the_array_box</code> .. 1301, 1304, 1801, 1802	<code>\</code> .....
<code>\c_@@_tikz_loaded_bool</code> .....	..... 1955,
..... 27, 41, 1215, 2407, 4471	1977, 5342, 5348, 5354, 5874, 5875, 5918,
<code>\@@_true_c:</code> .....	5927, 6008, 6013, 6018, 6023, 6028, 6056,
202, 1554	6061, 6067, 6074, 6081, 6087, 6088, 6102,
<code>\l_@@_type_of_col_tl</code> .. 770, 771, 2273, 2275	6108, 6114, 6115, 6126, 6138, 6145, 6151,
<code>\c_@@_types_of_matrix_seq</code> .....	6158, 6165, 6174, 6180, 6187, 6194, 6200,
..... 5951, 5952, 5957, 5961	6206, 6218, 6222, 6227, 6233, 6242, 6243,
<code>\@@_update_for_first_and_last_row:</code> ..	6257, 6258, 6274, 6278, 6279, 6297, 6298,
..... 876, 940, 1242, 2168, 2213	6340, 6341, 6392, 6393, 6443, 6444, 6497, 6498
<code>\@@_use_arraybox_with_notes:</code> ... 1408, 1863	<code>\{</code> .....
<code>\@@_use_arraybox_with_notes_b:</code> . 1405, 1847	..... 265, 1557,
<code>\@@_use_arraybox_with_notes_c:</code> .....	1710, 2251, 5370, 5727, 6107, 6187, 6340, 6443
..... 1406, 1437, 1799, 1861, 1900	<code>\}</code> .....
<code>\@@_vdottedline:n</code> .....	..... 265,
1721, 4467	1560, 2251, 5370, 5736, 6107, 6187, 6340, 6443
<code>\@@_vdottedline_i:n</code> .....	<code>\l</code> .....
4474, 4479, 4483	..... 2253, 5369
<code>\@@_vline:nn</code> .....	<code>\_</code> .....
1632, 3999, 4125	..... 5975, 5980, 5987, 5995,
<code>\@@_vline_i:nn</code> .....	5996, 6007, 6008, 6055, 6080, 6081, 6094,
2401, 4004, 4008	





<code>\color</code> . . . . .	99, 106, 160, 162, 1427, 1445, 2793, 2796, 2799, 2832, 2835, 2838, 2913, 2916, 2920, 2988, 3040, 3500, 3503, 3506, 3575, 3578, 3581, 3598, 3662, 3855, 3856, 3867, 3868, 4805, 4940, 5509, 5831, 5855	4108, 4209, 4220, 4225, 4231, 5053, 5059, 5101, 5215, 5226, 5282, 5285, 5421, 5424, 5432
<code>\colorlet</code> . . . . .	257, 258, 846, 853, 2158, 2202	
<code>\columncolor</code> . . . . .	1087, 1229, 2431, 3975	
<code>\cr</code> . . . . .	132, 150, 2137	
<code>\crr</code> . . . . .	1994	
cs commands:		
<code>\cs_generate_variant:Nn</code> . . . . .	58, 152, 3115, 3655, 4534, 4535	
<code>\cs_gset:Npn</code> . . . . .	99, 106, 2309, 2316, 2330, 2337, 4527	
<code>\cs_gset_eq:NN</code> . . . . .	186, 220, 1112, 1339, 2444	
<code>\cs_if_exist:NTF</code> . . . . .	57, 1094, 1097, 1255, 1262, 1273, 1280, 1340, 1343, 2490, 2491, 2576, 2589, 2644, 2657, 2746, 2768, 2876, 2894, 3468, 3486, 3543, 3561, 4514, 4567, 5064, 5082, 5484, 5629, 5636, 5652, 5659	
<code>\cs_if_exist_p:N</code> . . . . .	486, 3812, 3840, 4343, 4362, 4380	
<code>\cs_if_free:NTF</code> . . . . .	216, 2788, 2827, 2908, 2983, 3035	
<code>\cs_if_free_p:N</code> . . . . .	3614, 3616	
<code>\cs_new_protected:Npx</code> . . . . .	2470, 3625, 4430	
<code>\cs_set:Nn</code> . . . . .	634, 637, 640	
<code>\cs_set:Npn</code> . . . . .	95, 96, 102, 103, 108, 120, 121, 135, 137, 138, 160, 162, 335, 1068, 2535, 2597, 2665, 3510, 3585, 4251, 4252, 4258, 4259, 4807, 4896, 4912	
<code>\cs_set_nopar:Npn</code> . . . . .	994, 1006, 1103, 1106, 4709, 4710	
<code>\cs_set_nopar:Npx</code> . . . . .	1007	
<code>\cs_set_protected:Npn</code> . . . . .	5360	
<code>\cs_set_protected_nopar:Npn</code> . . . . .	4786, 5031	
<b>D</b>		
<code>\Ddots</code> . . . . .	1143, 3319, 3320, 3325, 3326	
<code>\ddots</code> . . . . .	1078, 1135	
<code>\diagbox</code> . . . . .	1156, 4786, 5031	
dim commands:		
<code>\dim_add:Nn</code> . . . . .	4678	
<code>\dim_compare:nNnTF</code> . . . . .	98, 105, 909, 915, 921, 1531, 2054, 2219, 2756, 2778, 2967, 4591, 4601, 5074, 5094, 5389	
<code>\dim_gzero:N</code> . . . . .	913, 919, 925	
<code>\dim_max:nn</code> . . . . .	4580, 4584	
<code>\dim_min:nn</code> . . . . .	4572, 4576	
<code>\dim_ratio:nn</code> . . . . .	3026, 3074, 3190, 3195, 3206, 3214, 3223, 3228, 3239, 3247	
<code>\dim_set:Nn</code> . . . . .	4553, 4560, 4571, 4575, 4579, 4583, 4595, 4596, 4605, 4606, 4650, 4663	
<code>\dim_set_eq:NN</code> . . . . .	4551, 4558, 4658, 4672	
<code>\dim_sub:Nn</code> . . . . .	4675	
<code>\dim_use:N</code> . . . . .	4592, 4602, 4653, 4654, 4666, 4667, 4690, 4691, 4692, 4693, 4717, 4718, 4719, 4720	
<code>\dim_zero_new:N</code> . . . . .	4550, 4552, 4557, 4559	
<code>\c_max_dim</code> . . . . .	2743, 2756, 2765, 2778, 4551, 4553, 4558, 4560, 4592, 4602, 5061, 5074, 5079, 5094, 5480, 5481, 5623, 5624	
<code>\l_tmpc_dim</code> . . . . .	282, 2514, 2518, 3926, 3927, 3947, 4084, 4092, 4097, 4102,	
		4108, 4209, 4220, 4225, 4231, 5053, 5059, 5101, 5215, 5226, 5282, 5285, 5421, 5424, 5432
		<code>\l_tmpd_dim</code> . . . . .
		283, 2516, 2519, 3944, 3947, 4093, 4097, 4216, 4220, 5055, 5059, 5079, 5090, 5094, 5097, 5101, 5224, 5227, 5423, 5424, 5436
<code>\dotfill</code> . . . . .	1154, 5378	
<code>\dots</code> . . . . .	1080	
<code>\doublerulesep</code> . . . . .	1626, 4094, 4106, 4217, 4229, 4264	
<code>\doublerulesepcolor</code> . . . . .	102	
<code>\draw</code> . . . . .	3103	
<b>E</b>		
<code>\egroup</code> . . . . .	1468	
else commands:		
<code>\else:</code> . . . . .	251	
<code>\endarray</code> . . . . .	1946, 1974	
<code>\endBNiceMatrix</code> . . . . .	5324	
<code>\endbNiceMatrix</code> . . . . .	5321	
<code>\endNiceArray</code> . . . . .	2288, 2297	
<code>\endNiceArrayWithDelims</code> . . . . .	2237, 2247	
<code>\endpgfscope</code> . . . . .	3160, 5435	
<code>\endpNiceMatrix</code> . . . . .	5312	
<code>\endsavenotes</code> . . . . .	1469	
<code>\endtabularnotes</code> . . . . .	1829	
<code>\endVNiceMatrix</code> . . . . .	5318	
<code>\endvNiceMatrix</code> . . . . .	5315	
<code>\enskip</code> . . . . .	1691, 1713	
<code>\ensuremath</code> . . . . .	3280, 3296, 3312, 3336, 3359	
<code>\everycr</code> . . . . .	131, 150, 1116	
exp commands:		
<code>\exp_after:wN</code> . . . . .	198, 1364, 1428, 1446, 1497, 5676	
<code>\exp_args:Ne</code> . . . . .	3381	
<code>\exp_args:NNc</code> . . . . .	4516	
<code>\exp_args:NNe</code> . . . . .	3377	
<code>\exp_args:Nne</code> . . . . .	2275	
<code>\exp_args:NNV</code> . . . . .	1966, 3267, 3283, 3299, 3315, 3339, 3440, 3516, 3594	
<code>\exp_args:NNv</code> . . . . .	1356	
<code>\exp_args:NNx</code> . . . . .	1051, 1735	
<code>\exp_args:No</code> . . . . .	3091	
<code>\exp_args:NV</code> . . . . .	1478, 1942, 1969, 1971, 5199	
<code>\exp_args:Nxx</code> . . . . .	4779, 4780	
<code>\exp_last_unbraced:NV</code> . . . . .	1234, 2424	
<code>\exp_not:N</code> . . . . .	42, 43, 46, 47, 1575, 1619, 3964, 3975, 4432, 4433, 4815, 4825, 4903, 4920, 5024, 5784	
<code>\exp_not:n</code> . . . . .	986, 2437, 3450, 3526, 3770, 3787, 3956, 3964, 3966, 3976, 4795, 4877, 4889, 4890, 5003, 5015, 5025, 5040, 5399, 5400	
<code>\ExplSyntaxOff</code> . . . . .	218, 2320, 2341, 2367, 2439, 4531	
<code>\ExplSyntaxOn</code> . . . . .	215, 2306, 2327, 2346, 2432, 4524	
<code>\extrarowheight</code> . . . . .	2873, 4808, 4897, 4913, 5646	
<b>F</b>		
<code>\fi</code> . . . . .	110, 1480, 4251, 4268	
fi commands:		
<code>\fi:</code> . . . . .	253	
<code>\five</code> . . . . .	2495, 2500	
flag commands:		
<code>\flag_clear_new:n</code> . . . . .	5775	
<code>\flag_height:n</code> . . . . .	5802	
<code>\flag_raise:n</code> . . . . .	5801	
<code>\fontdimen</code> . . . . .	1792	

fp commands:		
\fp_eval:n	3131	
\fp_to_dim:n	3166	
\futurelet	115	
<b>G</b>		
\globaldefs	1464, 4981	
group commands:		
\group_insert_after:N	5383, 5384, 5386, 5387	
<b>H</b>		
\halign	1130	
\hbox	1015,	
	1432, 1861, 1900, 2019, 2037, 2064, 2068, 2094	
hbox commands:		
\hbox:n	64, 66, 69	
\hbox_gset:Nn	4800	
\hbox_overlap_left:n	1998, 2171	
\hbox_overlap_right:n	389, 2115, 2217	
\hbox_set:Nn		
	372, 1296, 1298, 1423, 1906, 1921, 5043	
\hbox_set:Nw	835, 1304, 1663, 2146, 2192	
\hbox_set_end:	931, 1375, 1671, 2165, 2210	
\hbox_to_wd:nn	414, 439	
\Hdotsfor	1148, 5975, 6131	
\hdotsfor	1081	
\hdottedline	1145	
\heavyrulewidth	1839	
\hfil	1553	
\hfill	128, 145	
\Hline	1146, 4254	
\hline	108	
\hrule	112, 128, 145, 1060, 1839, 5514, 5836, 5860	
\hskip	111	
\Hspace	1147	
\hspace	3372	
\hss	1553	
<b>I</b>		
\ialign	1012, 1106, 1129, 4960	
\iddots	1144, 3343, 3344, 3349, 3350	
\iddots	59, 1079, 1136	
if commands:		
\if_mode_math:	251	
\IfBooleanTF	1366	
\ifnum	110, 4251, 4268	
\ifstandalone	1343	
int commands:		
\int_case:nnTF	3317, 3323, 3341, 3347	
\int_compare:nNnTF	123, 124, 140, 833, 834,	
	843, 850, 886, 896, 1057, 1059, 1189, 1191,	
	1240, 1250, 1269, 1376, 1380, 1391, 1395,	
	1396, 1460, 1690, 1817, 1856, 1895, 1967,	
	1995, 2195, 2551, 2558, 2562, 2564, 2619,	
	2626, 2630, 2632, 2795, 2834, 2915, 2950,	
	2952, 3400, 3414, 3502, 3577, 3883, 3922,	
	3939, 3971, 3988, 3989, 3996, 3997, 4001,	
	4406, 4408, 4410, 4412, 4804, 4841, 4853,	
	5121, 5150, 5154, 5217, 5219, 5338, 5340,	
	5342, 5346, 5348, 5350, 5352, 5354, 5680, 5682	
\int_compare_p:n		
	2697, 2698, 2699, 2700, 3898, 3900, 5209, 5210	
\int_do_until:nNnn	3834	
\int_gadd:Nn	3413	
\int_gincr:N	832, 859, 1346, 1599, 1655,	
	1677, 1683, 2090, 2190, 3013, 3061, 4509, 4785	
\int_if_even:nTF	3745, 5802	
\int_if_odd_p:n	3831	
\int_incr:N	368, 1609	
\int_min:nn		
	2509, 2511, 2513, 2515, 2523, 2525, 2705, 2706	
\int_step_inline:nn		
	2507, 3741, 3743, 3780, 3782, 5693, 5710	
\int_step_inline:nnnn	4338, 4356, 4371, 4374	
\l_tmpc_int	3832, 3833, 3834	
\c_zero_int	1690, 3648, 3889	
iow commands:		
\iow_now:Nn	76,	
	213, 2346, 2347, 2349, 2367, 2432, 2433, 2439	
\iow_shipout:Nn	2306, 2307, 2314,	
	2320, 2327, 2328, 2335, 2341, 4524, 4525, 4531	
\item	1822, 1828	
<b>K</b>		
\kern	69	
keys commands:		
\keys_define:nn	481,	
	501, 508, 577, 624, 676, 715, 762, 780, 801,	
	814, 3363, 3752, 3793, 4498, 4728, 4932,	
	5234, 5301, 5441, 5446, 5526, 5547, 5556, 5892	
\l_keys_key_str		
	5874, 6039, 6045, 6124, 6227,	
	6232, 6242, 6257, 6278, 6296, 6339, 6391, 6442	
\keys_set:nn	510, 524, 751, 754, 761,	
	1360, 1361, 2274, 2284, 2293, 2447, 2798,	
	2837, 2918, 2987, 3039, 3505, 3580, 3597,	
	3759, 3807, 4511, 4997, 5448, 5452, 5579, 5618	
\keys_set_known:nn		
	3329, 3353, 4768, 5191, 5246	
\l_keys_value_tl	6086, 6153, 6160, 6492	
<b>L</b>		
\Ldots	1140, 3270, 3273	
\ldots	1075, 1132	
\leaders	128, 145	
\left	1428, 1909, 1924, 5510, 5832, 5856	
legacy commands:		
\legacy_if:nTF	608	
\line	2419, 6107, 6253	
<b>M</b>		
\makebox	1674	
\mathinner	61	
mode commands:		
\mode_leave_vertical:	1335, 1644	
msg commands:		
\msg_error:nn	12	
\msg_error:nnn	13	
\msg_error:nnnn	14, 4978, 4985, 4989	
\msg_fatal:nn	15	
\msg_fatal:nnn	16	
\msg_new:nnn	17	
\msg_new:nnnn	18	
\msg_redirect_name:nnn	20	
\multicolumn	1150, 3374, 3426, 3432, 3453	
\multispan	124, 125, 141, 142	
\myfiledate	6	
\myfileversion	7	

<b>N</b>	
<code>\newcolumnntype</code> .....	225, 226, 227
<code>\newcounter</code> .....	328
<code>\NewDocumentCommand</code> 340, 363, 760, 2446, 3267, 3283, 3299, 3315, 3339, 3440, 3516, 3594, 3674, 3689, 3704, 3713, 3734, 3739, 3757, 3802, 3952, 3960, 3969, 5331, 5371, 5586, 5599	
<code>\NewDocumentEnvironment</code> .....	1318, 1939, 1948, 2230, 2240, 2270, 2281, 2289, 4507
<code>\NewExpandableDocumentCommand</code> .....	231, 4743
<code>\newlist</code> .....	344, 355
<code>\NiceArray</code> .....	2286, 2295
<code>\NiceArrayWithDelims</code> .....	2235, 2245
nicematrix commands:	
<code>\g_nicematrix_code_after_tl</code> .....	267, 617, 1963, 2424, 2426, 5000, 5012, 5459, 5466
<code>\g_nicematrix_code_before_tl</code> .....	1174, 2428, 2437, 3786, 3954, 3962, 3973, 5022
<code>\NiceMatrixLastEnv</code> .....	231
<code>\NiceMatrixOptions</code> .....	760, 6297, 6497
<code>\NiceMatrixoptions</code> .....	6157
<code>\noalign</code> 98, 105, 110, 133, 1039, 1112, 4251, 4420	
<code>\nobreak</code> .....	358
<code>\normalbaselines</code> .....	1100
<code>\NotEmpty</code> .....	1157
<code>\nulldelimiterspace</code> ....	1920, 1932, 5495, 5672
<code>\nullfont</code> ....	5506, 5513, 5828, 5835, 5852, 5859
<code>\numexpr</code> .....	153, 154
<b>O</b>	
<code>\omit</code> .....	123, 1997, 2013, 2089, 5456
<code>\OnlyMainNiceMatrix</code> .....	1153, 3980
<b>P</b>	
<code>\par</code> .....	1816, 1824
peek commands:	
<code>\peek_meaning:NTF</code> .....	158, 370, 1069
<code>\peek_meaning_ignore_spaces:NTF</code> 1941, 4254	
<code>\peek_meaning_remove_ignore_spaces:NTF</code> 148	
<code>\peek_remove_spaces:n</code> .....	3398
<code>\pgfdeclareshape</code> .....	2493
<code>\pgfextracty</code> .....	5124
<code>\pgfgetlastxy</code> .....	432
<code>\pgfpathcircle</code> .....	3255
<code>\pgfpathlineto</code> .....	4102, 4108, 4225, 4231, 5285, 5298, 5424, 5687, 5704, 5738
<code>\pgfpathmoveto</code> .....	4101, 4107, 4224, 4230, 5284, 5297, 5419, 5686, 5703, 5729
<code>\pgfpathrectanglecorners</code> 3946, 4095, 4218, 5225	
<code>\pgfpointadd</code> .....	430
<code>\pgfpointanchor</code> .....	163, 234, 2721, 2732, 2749, 2771, 2879, 2897, 4570, 4578, 5069, 5087, 5106, 5108, 5125, 5159, 5487, 5632, 5639, 5655, 5662, 5774, 5780
<code>\pgfpointdiff</code> .....	431, 1202, 1210
<code>\pgfpointlineattime</code> .....	3125
<code>\pgfpointorigin</code> .....	2004, 2127, 2501
<code>\pgfpointscale</code> .....	430
<code>\pgfpointshapeborder</code> .....	3635, 3638
<code>\pgfrememberpicturepositiononpagetrue</code> ...	864, 958, 1027, 2003, 2027, 2045, 2076, 2102, 2125, 2479, 2506, 3082, 3634, 4077, 4202, 4441, 4486, 4613, 4623, 4634, 5046, 5193, 5262, 5414, 5473, 5620
<code>\pgfscope</code> .....	3122, 5431
<code>\pgfset</code> 399, 424, 959, 5136, 5170, 5430, 5494, 5622	
<code>\pgfsetbaseline</code> .....	957
<code>\pgfsetcornersarced</code> .....	3945, 5201
<code>\pgfsetlinewidth</code> ..	4111, 4234, 5228, 5265, 5674
<code>\pgfsetrectcap</code> .....	4112, 4235
<code>\pgfsetroundcap</code> .....	5427
<code>\pgfsetstrokecolor</code> .....	5199
<code>\pgfsyspdfmark</code> .....	216, 217
<code>\pgftransformrotate</code> .....	3129
<code>\pgftransformshift</code> .....	405, 430, 2517, 3123, 5135, 5157, 5432, 5436, 5496, 5752, 5762
<code>\pgfusepath</code> .....	3149, 3159, 3665, 3859, 3871, 4098, 5229
<code>\pgfusepathqfill</code> .....	3261, 4221
<code>\pgfusepathqstroke</code> .....	4113, 4236, 5286, 5299, 5428, 5688, 5705, 5739
<code>\phantom</code> .....	3280, 3296, 3312, 3336, 3359
<code>\pNiceMatrix</code> .....	5311
prg commands:	
<code>\prg_do_nothing:</code> .....	177, 186, 192, 220, 498, 1112, 2422
<code>\prg_new_conditional:Nnn</code> .....	3886, 3894
<code>\prg_replicate:nn</code> .....	378, 2085, 2086, 3453, 4103, 4226, 5341, 5344, 5347, 5353
<code>\prg_return_false:</code> .....	3891, 3903
<code>\prg_return_true:</code> .....	3892, 3902
<code>\ProcessKeysOptions</code> .....	5911
<code>\ProvideDocumentCommand</code> .....	59
<code>\ProvidesExplPackage</code> .....	4
<b>Q</b>	
<code>\quad</code> .....	359
quark commands:	
<code>\q_stop</code> .....	120, 121, 137, 138, 159, 161, 1364, 1497, 1564, 1711, 1958, 1961, 3588, 3602, 3603, 3669, 3724, 3729, 3826, 3912, 3913, 3934, 3935, 4707, 4714, 4745, 4748, 5207, 5216, 5255, 5258, 5326, 5335, 5588, 5593, 5606, 5609, 5676, 5787, 5795
<b>R</b>	
<code>\rectanglecolor</code> .....	1224, 2430, 3964
<code>\refstepcounter</code> .....	387
regex commands:	
<code>\regex_const:Nn</code> .....	210
<code>\regex_match:nnTF</code> .....	5568
<code>\regex_replace_all:NnN</code> .....	1502
<code>\relax</code> .....	153, 154
<code>\renewcommand</code> .....	196
<code>\RenewDocumentEnvironment</code> .....	5310, 5313, 5316, 5319, 5322
<code>\RequirePackage</code> .....	1, 3, 9, 10, 11
<code>\right</code> .....	1446, 1916, 1928, 5519, 5840, 5864
<code>\rotate</code> .....	1152
<code>\roundedrectanglecolor</code> .....	1225, 5024
<code>\rowcolor</code> .....	1086, 1226
<code>\rowcolors</code> .....	1227
<b>S</b>	
<code>\savedanchor</code> .....	2495
<code>\savenotes</code> .....	1322
scan commands:	
<code>\scan_stop:</code> .....	2425

<code>\scriptstyle</code> . . . . .	839, 2148, 2194, 3110, 3111, 3145, 3155
seq commands:	
<code>\seq_clear:N</code> . . . . .	1495
<code>\seq_clear_new:N</code> . . . . .	2348, 4316
<code>\seq_count:N</code> . . . . .	1968, 3651
<code>\seq_gclear:N</code> . . . . .	1178, 1349, 1350, 1351, 1352, 1844, 2423
<code>\seq_gclear_new:N</code> . . . . .	1160, 1161, 1233, 1964, 1980
<code>\seq_gpop_left:NN</code> . . . . .	1970, 1982
<code>\seq_gput_left:Nn</code> . . . . .	613, 3402, 3404, 4995
<code>\seq_gput_right:Nn</code> . . . . .	369, 1572, 2676, 3405, 3650, 4874, 4886, 5007, 5402, 5574, 5594
<code>\seq_gset_from_clist:Nn</code> . . . . .	2351, 2363
<code>\seq_gset_map_x:NNn</code> . . . . .	2450
<code>\seq_gset_split:Nnn</code> . . . . .	1966, 1981
<code>\seq_if_empty:NTF</code> . . . . .	1804
<code>\seq_if_empty_p:N</code> . . . . .	2396, 2397, 2398
<code>\seq_if_exist:NTF</code> . . . . .	1181
<code>\seq_if_in:NnTF</code> . . . . .	611, 3784, 4054, 4060, 4067, 4179, 4185, 4192, 4573, 4581, 5067, 5085, 5570, 5961
<code>\seq_item:Nn</code> . . . . .	1185, 1188, 1197, 1198, 1205, 1206
<code>\seq_map_function:NN</code> . . . . .	1972
<code>\seq_map_indexed_inline:Nn</code> . . . . .	3646, 3660
<code>\seq_map_inline:Nn</code> . . . . .	1822, 1828, 1984, 2690, 3846, 4015, 4017, 4019, 4141, 4143, 4145, 4401, 4961, 5678
<code>\seq_mapthread_function:NNN</code> . . . . .	4702
<code>\seq_new:N</code> . . . . .	242, 256, 287, 288, 289, 290, 291, 296, 329, 5951
<code>\seq_put_right:Nn</code> . . . . .	4388
<code>\seq_set_eq:NN</code> . . . . .	3815
<code>\seq_set_filter:NNn</code> . . . . .	3817, 3844
<code>\seq_set_from_clist:Nn</code> . . . . .	5952
<code>\seq_set_map_x:NNn</code> . . . . .	5957
<code>\seq_use:Nnnn</code> . . . . .	2365, 6120, 6502
<code>\l_tmpa_seq</code> . . . . .	3817, 3844
<code>\l_tmpb_seq</code> . . . . .	3815, 3817, 3844, 3846
<code>\setlist</code> . . . . .	345, 356, 649, 654, 665, 670
skip commands:	
<code>\skip_gadd:Nn</code> . . . . .	2061
<code>\skip_gset:Nn</code> . . . . .	2052
<code>\skip_gset_eq:NN</code> . . . . .	2059, 2060
<code>\skip_horizontal:N</code> . . . . .	129, 146, 1305, 1306, 1373, 1374, 1398, 1399, 1435, 1436, 1439, 1440, 1455, 1456, 1486, 1492, 1575, 1719, 1732, 1739, 1933, 1934, 1936, 1937, 2008, 2009, 2021, 2023, 2039, 2041, 2063, 2070, 2072, 2091, 2096, 2098, 2117, 2118, 2176, 2177, 2178, 2181, 2216, 2221, 2222, 2223
<code>\skip_horizontal:n</code> . . . . .	390, 1621
<code>\skip_vertical:N</code> . . . . .	133, 1021, 1023, 1431, 1442, 1813, 1838, 4420
<code>\skip_vertical:n</code> . . . . .	901, 4261
<code>\g_tmpa_skip</code> . . . . .	2052, 2059, 2060, 2061, 2063, 2091
<code>\c_zero_skip</code> . . . . .	1117
<code>\space</code> . . . . .	264, 265
<code>\stepcounter</code> . . . . .	376, 381
str commands:	
<code>\c_backslash_str</code> . . . . .	264
<code>\c_colon_str</code> . . . . .	759
<code>\str_case:nn</code> . . . . .	3381, 5128, 5140, 5162, 5174, 5723, 5732
<code>\str_case:nnTF</code> . . . . .	1403, 1540, 1778, 4319, 5799, 5812
<code>\str_clear_new:N</code> . . . . .	5617
<code>\str_foldcase:n</code> . . . . .	3381
<code>\str_gclear:N</code> . . . . .	2442
<code>\str_gset:Nn</code> . . . . .	1332, 2234, 2243, 2272, 2283, 2291, 5362
<code>\str_if_empty:NTF</code> . . . . .	868, 970, 1030, 1253, 1271, 1331, 2005, 2030, 2048, 2079, 2105, 2128, 2233, 2242, 2312, 2333, 4694, 4721, 5744, 5758, 5768
<code>\str_if_eq:nnTF</code> . . . . .	84, 263, 1010, 1567, 1570, 1614, 1726, 1953, 1955, 1988, 5197, 5464
<code>\str_if_eq_p:nn</code> . . . . .	487, 1580, 1708, 1709, 1710, 1711, 3916, 3920, 4753, 4758
<code>\str_if_in:NnTF</code> . . . . .	1766, 1870
<code>\str_new:N</code> . . . . .	259, 469, 758
<code>\str_range:Nnn</code> . . . . .	1770, 1874
<code>\str_set:Nn</code> . . . . .	610, 747, 5573
<code>\str_set_eq:NN</code> . . . . .	614, 759
<code>\l_tmpa_str</code> . . . . .	610, 611, 613, 614
<code>\strut</code> . . . . .	1822, 1828
<code>\strutbox</code> . . . . .	2873, 2891, 5646, 5649
<code>\SubMatrix</code> . . . . .	1231, 2406, 5597, 6055, 6087, 6094, 6113, 6260
<b>T</b>	
<code>\tabcolsep</code> . . . . .	1003, 1435, 1439
<code>\tabskip</code> . . . . .	1117
<code>\tabularnote</code> . . . . .	340, 363, 370, 6185, 6205
<code>\tabularnotes</code> . . . . .	1827
TeX and L <sup>A</sup> T <sub>E</sub> X commands:	
<code>\@BTnormal</code> . . . . .	1092
<code>\@acol</code> . . . . .	993
<code>\@acoll</code> . . . . .	991
<code>\@acolr</code> . . . . .	992
<code>\@array@array</code> . . . . .	995
<code>\@arrayacol</code> . . . . .	991, 992, 993
<code>\@arstrutbox</code> . . . . .	861, 862, 901, 1119, 1121, 1123, 1126, 1128, 1646, 1650
<code>\@currenvir</code> . . . . .	5464
<code>\@depth</code> . . . . .	5516, 5837, 5861
<code>\@gobblethree</code> . . . . .	217
<code>\@halignto</code> . . . . .	994, 1006, 1007
<code>\@height</code> . . . . .	113, 128, 145, 5514, 5836, 5860
<code>\@ifclassloaded</code> . . . . .	51, 54, 5932, 5942
<code>\@ifnextchar</code> . . . . .	1165
<code>\@ifpackageloaded</code> . . . . .	30, 33, 36, 39, 75, 91, 167, 5935, 5945
<code>\@mainaux</code> . . . . .	76, 213, 2306, 2307, 2314, 2320, 2327, 2328, 2335, 2341, 2346, 2347, 2349, 2367, 2432, 2433, 2439, 4524, 4525, 4531
<code>\@tabarray</code> . . . . .	1008
<code>\@tempswafalse</code> . . . . .	1480
<code>\@tempswattrue</code> . . . . .	1479
<code>\@temptokena</code> . . . . .	176, 179, 198, 200, 1478, 1497
<code>\@whiles</code> . . . . .	1480
<code>\@width</code> . . . . .	113, 5517, 5838, 5862
<code>\@xhline</code> . . . . .	116
<code>\bBigg@</code> . . . . .	1296, 1298
<code>\cMaxMatrixCols</code> . . . . .	2262, 5989
<code>\c@tabularnote</code> . . . . .	1806, 1817, 1845

<code>\col@sep</code> . . . . .	1003, 1004, 1398, 1456, 2008, 2061, 2117, 2181, 2216, 2760, 2782
<code>\CT@arc</code> . . . . .	95, 96
<code>\CT@arc@</code> . . . . .	94, 99, 114, 127, 144, 160, 162, 1339, 1839, 2444, 4110, 4233, 4485, 5198, 5264, 5426, 5677
<code>\CT@drs</code> . . . . .	102, 103
<code>\CT@drsc@</code> . . . . .	101, 106, 4087, 4090, 4212, 4215
<code>\CT@everycr</code> . . . . .	1110
<code>\CT@row@color</code> . . . . .	1112
<code>\if@tempswa</code> . . . . .	1480
<code>\NC@</code> . . . . .	1068
<code>\NC@find</code> . . . . .	177, 205
<code>\NC@list</code> . . . . .	1480
<code>\NC@rewrite@S</code> . . . . .	178, 196
<code>\new@ifnextchar</code> . . . . .	1165
<code>\newcol@</code> . . . . .	1070, 1071
<code>\nicematrix@redefine@check@rerun</code> . . . . .	76, 79
<code>\pgf@relevantforpicturesizefalse</code> . . . . .	2480, 3083, 3252, 3659, 3821, 4078, 4203, 4614, 4624, 4635, 5047, 5194, 5263, 5413, 5474, 5621
<code>\pgfsys@getposition</code> . . . . .	1194, 1200, 1208
<code>\pgfsys@markposition</code> . . . . .	1022, 1193, 2001, 2022, 2040, 2071, 2097, 2121
<code>\pgfutil@check@rerun</code> . . . . .	81, 82
<code>\reserved@a</code> . . . . .	115
<code>\rvtx@ifformat@geq</code> . . . . .	57
<code>\set@color</code> . . . . .	4804, 5043
<code>\tikz@library@external@loaded</code> . . . . .	1340
tex commands:	
<code>\tex_mkern:D</code> . . . . .	63, 65, 67, 70
<code>\tex_the:D</code> . . . . .	200
<code>\textfont</code> . . . . .	1792
<code>\textit</code> . . . . .	332
<code>\textsuperscript</code> . . . . .	333, 334
<code>\the</code> . . . . .	153, 154, 1480, 1497
<code>\thetabularnote</code> . . . . .	335
<code>\tikzexternaldisable</code> . . . . .	1342
<code>\tikzset</code> . . . . .	1217, 1344, 2409
tl commands:	
<code>\tl_clear:N</code> . . . . .	4036, 4047, 4161, 4172, 5189
<code>\tl_clear_new:N</code> . . . . .	3725, 3726, 3805, 4011, 4137, 5589, 5590, 5602, 5603, 5604, 5605
<code>\tl_const:Nn</code> . . . . .	42, 43, 46, 47, 456, 2139, 2184, 5788
<code>\tl_count:n</code> . . . . .	1773, 1877
<code>\tl_gclear:N</code> . . . . .	1482, 2421, 2426, 3664
<code>\tl_gclear_new:N</code> . . . . .	1168, 1169, 1170, 1171, 1172, 1173, 1174
<code>\tl_gput_left:Nn</code> . . . . .	980, 1508, 3766, 3786, 3973
<code>\tl_gput_right:Nn</code> . . . . .	980, 1520, 1574, 1617, 1631, 1691, 1692, 1700, 1720, 3442, 3518, 3653, 3954, 3962, 4266, 4425, 4788, 5000, 5012, 5022, 5033, 5392
<code>\tl_gset:Nn</code> . . . . .	179, 183, 185, 1326, 1485, 1491, 2435, 3651
<code>\tl_if_blank:nTF</code> . . . . .	3676, 3679, 3691, 3694, 3706, 3709, 3715, 3718, 3852, 3864, 4745
<code>\tl_if_blank_p:n</code> . . . . .	3915, 3919, 4087, 4212, 4752, 4757
<code>\tl_if_empty:NTF</code> . . . . .	1047, 1363, 1426, 1444, 1816, 2404, 2405, 2428, 3936, 3937, 4025, 4029, 4040, 4150, 4154, 4165, 4763, 4803, 4998, 5020, 5195, 5508, 5675, 5830, 5854
<code>\tl_if_empty:nTF</code> . . . . .	591, 722, 764, 784, 804, 822, 1950, 1977, 2799, 2838, 2919, 2988, 3040, 3506, 3581, 3598, 3854, 3866, 5565, 5797, 5974
<code>\tl_if_empty_p:N</code> . . . . .	1514, 1526, 3119, 3120
<code>\tl_if_empty_p:n</code> . . . . .	1807
<code>\tl_if_eq:NNTF</code> . . . . .	3084
<code>\tl_if_eq:NnTF</code> . . . . .	1049, 1483, 1729, 1754, 1865, 4123, 4246, 4460, 4462, 5692, 5709
<code>\tl_if_eq:nnTF</code> . . . . .	603, 738, 3647
<code>\tl_if_exist:NTF</code> . . . . .	1353
<code>\tl_if_in:NnTF</code> . . . . .	3825, 3911, 3933
<code>\tl_if_single_token:nTF</code> . . . . .	547, 746
<code>\tl_if_single_token_p:n</code> . . . . .	58
<code>\tl_item:Nn</code> . . . . .	182, 183, 185
<code>\tl_map_inline:nn</code> . . . . .	1951
<code>\tl_new:N</code> . . . . .	181, 184, 237, 255, 267, 268, 269, 272, 276, 298, 299, 302, 304, 306, 330, 455, 459, 476, 478, 479
<code>\tl_put_left:Nn</code> . . . . .	1092
<code>\tl_put_right:Nn</code> . . . . .	593, 1242, 1314, 1356
<code>\tl_range:nnn</code> . . . . .	84
<code>\tl_set:Nn</code> . . . . .	182, 3730, 3731, 3827, 3848, 3921, 3923, 3938, 3940, 4010, 4769, 5218, 5220, 5259, 5260
<code>\tl_set_eq:NN</code> . . . . .	457, 3727, 3728, 4026, 4151, 4464, 4494, 4765, 5256, 5257, 5591, 5592, 5607, 5608, 5610, 5611
<code>\tl_set_rescan:Nnn</code> . . . . .	1965, 3266, 3439, 3515, 3593
<code>\tl_to_str:n</code> . . . . .	5958
<code>\g_tmpa_tl</code> . . . . .	179, 182, 185
<code>\l_tmpc_tl</code> . . . . .	3725, 3727, 3730, 4011, 4025, 4026, 4029, 4034, 4036, 4040, 4045, 4047, 4137, 4150, 4151, 4154, 4159, 4161, 4165, 4170, 4172, 5256, 5273, 5279, 5589, 5591, 5595
<code>\l_tmpd_tl</code> . . . . .	3726, 3728, 3731, 5257, 5269, 5290, 5590, 5592, 5595
token commands:	
<code>\token_to_str:N</code> . . . . .	5975, 6050, 6055, 6061, 6086, 6094, 6107, 6113, 6131, 6144, 6185, 6192, 6205, 6222, 6232, 6244, 6253, 6259, 6297, 6497
U	
<code>\unskip</code> . . . . .	1832
use commands:	
<code>\use:N</code> . . . . .	983, 1258, 1265, 1276, 1283, 1309, 1310, 1370, 1371, 2257, 2277, 3663
<code>\use:n</code> . . . . .	3599, 3980, 4813, 4823, 4901, 4918, 5783
<code>\usepackage</code> . . . . .	5937, 5947
<code>\usepgfmodule</code> . . . . .	2
V	
vbox commands:	
<code>\vbox:n</code> . . . . .	69
<code>\vbox_set_top:Nn</code> . . . . .	898
<code>\vbox_to_ht:nn</code> . . . . .	410, 437, 1912, 1925
<code>\vbox_to_zero:n</code> . . . . .	900
<code>\vcenter</code> . . . . .	1429, 1910, 5511, 5833, 5857, 6222
<code>\Vdots</code> . . . . .	1142, 3302, 3305
<code>\vdots</code> . . . . .	1077, 1134

<code>\Vdotsfor</code> .....	1149	<code>\vtop</code> .....	1019
<code>\vfill</code> .....	413, 439		
<code>\vline</code> .....	114		
<code>\VNiceMatrix</code> .....	5317		
<code>\vNiceMatrix</code> .....	5314		
<code>\vrule</code> .....	112, 1646, 1650		
<code>\vskip</code> .....	111		
		<b>X</b>	
		<code>\xglobal</code> .....	846, 853, 2158, 2202
		<b>Z</b>	
		<code>\Z</code> .....	5568

## 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	A small remark .....	6
<b>5</b>	<b>The rules</b>	<b>7</b>
5.1	Some differences with the classical environments .....	7
5.1.1	The vertical rules .....	7
5.1.2	The command <code>\cline</code> .....	8
5.2	The thickness and the color of the rules .....	8
5.3	The tools of nicematrix for the rules .....	8
5.3.1	The keys <code>hlines</code> and <code>vlines</code> .....	9
5.3.2	The key <code>hvlines</code> .....	9
5.3.3	The (empty) corners .....	9
5.4	The command <code>\diagbox</code> .....	10
5.5	Dotted rules .....	11
<b>6</b>	<b>The color of the rows and columns</b>	<b>11</b>
6.1	Use of <code>colortbl</code> .....	11
6.2	The tools of nicematrix in the <code>\CodeBefore</code> .....	12
6.3	Color tools with the syntax of <code>colortbl</code> .....	15
<b>7</b>	<b>The width of the columns</b>	<b>16</b>
<b>8</b>	<b>The exterior rows and columns</b>	<b>17</b>
<b>9</b>	<b>The continuous dotted lines</b>	<b>18</b>
9.1	The option <code>nullify-dots</code> .....	20
9.2	The commands <code>\Hdotsfor</code> and <code>\Vdotsfor</code> .....	20
9.3	How to generate the continuous dotted lines transparently .....	21
9.4	The labels of the dotted lines .....	22
9.5	Customisation of the dotted lines .....	22
9.6	The dotted lines and the rules .....	23
<b>10</b>	<b>The <code>\CodeAfter</code></b>	<b>24</b>
10.1	The command <code>\line</code> in the <code>\CodeAfter</code> .....	24
10.2	The command <code>\SubMatrix</code> in the <code>\CodeAfter</code> .....	24

<b>11</b>	<b>The notes in the tabulars</b>	<b>26</b>
11.1	The footnotes . . . . .	26
11.2	The notes of tabular . . . . .	26
11.3	Customisation of the tabular notes . . . . .	28
11.4	Use of <code>{NiceTabular}</code> with <code>threeparttable</code> . . . . .	30
<b>12</b>	<b>Other features</b>	<b>30</b>
12.1	Use of the column type <code>S</code> of <code>siunitx</code> . . . . .	30
12.2	Alignment option in <code>{NiceMatrix}</code> . . . . .	30
12.3	The command <code>\rotate</code> . . . . .	30
12.4	The option <code>small</code> . . . . .	31
12.5	The counters <code>iRow</code> and <code>jCol</code> . . . . .	32
12.6	The option <code>light-syntax</code> . . . . .	32
12.7	Color of the delimiters . . . . .	33
12.8	The environment <code>{NiceArrayWithDelims}</code> . . . . .	33
<b>13</b>	<b>Use of Tikz with <code>nicematrix</code></b>	<b>33</b>
13.1	The nodes corresponding to the contents of the cells . . . . .	33
13.2	The “medium nodes” and the “large nodes” . . . . .	34
13.3	The nodes which indicate the position of the rules . . . . .	35
13.4	The nodes corresponding to the command <code>\SubMatrix</code> . . . . .	36
<b>14</b>	<b>API for the developpers</b>	<b>36</b>
<b>15</b>	<b>Technical remarks</b>	<b>37</b>
15.1	Definition of new column types . . . . .	37
15.2	Diagonal lines . . . . .	38
15.3	The “empty” cells . . . . .	38
15.4	The option <code>exterior-arraycolsep</code> . . . . .	39
15.5	Incompatibilities . . . . .	39
<b>16</b>	<b>Examples</b>	<b>39</b>
16.1	Notes in the tabulars . . . . .	39
16.2	Dotted lines . . . . .	41
16.3	Dotted lines which are no longer dotted . . . . .	41
16.4	Stacks of matrices . . . . .	42
16.5	How to highlight cells of a matrix . . . . .	45
16.6	Utilisation of <code>\SubMatrix</code> in the <code>\CodeBefore</code> . . . . .	48
<b>17</b>	<b>Implementation</b>	<b>48</b>
<b>18</b>	<b>History</b>	<b>195</b>
	<b>Index</b>	<b>201</b>