

The package `nicematrix`*

F. Pantigny
fpantigny@wanadoo.fr

October 6, 2020

Abstract

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

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

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.

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

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

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

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

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

Important

Since the version 5.0 of `nicematrix`, one must use the letters `l`, `c` and `r` in the preambles of the environments and no longer the letters `L`, `C` and `R`.

For sake of compatibility with the previous versions, there exists an option `define-L-C-R` which must be used when loading `nicematrix`.

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

*This document corresponds to the version 5.4 of `nicematrix`, at the date of 2020/10/06.

1 The environments of this package

The package `nicematrix` defines the following new environments.

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

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

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

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

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

Important

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

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

2 The vertical space between the rows

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

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

Inspired by the package `cellspace` which deals with that problem, the package `nicematrix` provides two keys `cell-space-top-limit` and `cell-space-bottom-limit` similar to the parameters `\cellspacetoplimit` and `\cellspacebottomlimit` of `cellspace`. The initial value of these parameters is 0 pt in order to have for the environments of `nicematrix` the same behaviour as those of `array` and `amsmath`. However, a value of 1 pt would probably be a good choice and we suggest to set them with `\NiceMatrixOptions`.¹

```
\NiceMatrixOptions{cell-space-top-limit = 1pt,cell-space-bottom-limit = 1pt}

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

¹One 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

New 5.2 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-top-limit=1pt,cell-space-bottom-limit=1pt}
```

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

$$A = \begin{pmatrix} \frac{1}{A} & \frac{1}{B} & 0 & 0 \\ \frac{1}{C} & \frac{1}{D} & 0 & 0 \\ 0 & 0 & A & B \\ 0 & 0 & D & D \end{pmatrix}$$

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. The second argument is the content of the block.

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

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

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

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

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

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

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

New 5.3 It's possible to set the horizontal position of the block with one of the keys `l`, `c` and `r`.

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

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

One must remark that, by default, the commands `\Blocks` don't create space (excepted, to some extent, the mono-column blocks: see 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{2-2}{\LARGE\color{blue}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

New 5.4 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 $\text{>\{...\}}$ in the preamble of the array are taken into account for the mono-column blocks of that column (this behaviour is probably expected).

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

4.3 A small remark

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

```

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

```

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

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`).²

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

First	Second
Peter	
Mary	George

New 5.2 However, the vertical rules are not drawn in the blocks.

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`.

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

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

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

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

However, in this case, it is probably more clever to add a command `\OnlyMainNiceMatrix` (cf. p. 32):

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

5.1.2 The command `\cline`

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

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

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

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

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

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

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

²This is the behaviour since the version 5.1 of `nicematrix`. Prior to that version, the behaviour was the standard behaviour of `array`.

5.2 The thickness and the color of the rules

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

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

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

```
\begin{NiceTabular}{|ccc|}[rules/color=[gray]{0.9},rules/width=1pt]
\hline
rose & tulipe & lys \\
arum & iris & violette \\
muguet & dahlia & souci \\
\hline
\end{NiceTabular}
```

rose	tulipe	lys
arum	iris	violette
muguet	dahlia	souci

If one wishes to define new specifiers for columns in order to draw vertical rules (for example with a specific color or thicker than the standard rules), he should consider the command `\OnlyMainNiceMatrix` described on page 32.

5.3 The keys `hlines` and `vlines`

The key `hlines` draws all the horizontal rules and the key `vlines` draws all the vertical rules excepted in the blocks (and the virtual blocks determined by dotted lines). In fact, in the environments with delimiters (as `{pNiceMatrix}` or `{bNiceArray}`) the exteriors rules are not drawn (as expected).

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

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

5.4 The key `hvlines`

The key `hvlines` draws all the vertical and horizontal rules excepted in the blocks (and the virtual blocks determined by dotted lines).

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

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

5.5 The key `hvlines-except-corners`

The key `hvlines-except-corners` draws all the horizontal and vertical rules, excepted in the blocks (and the virtual blocks determined by dotted lines) and excepted in the empty corners.

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

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

As we can see, an “empty corner” is composed by the reunion of all the empty rectangles starting from the cell actually in the corner of the array.

New 5.2 It’s possible to give as value to the key `hvlines-except-corners` a list of the corners to take into consideration. The corners are designed by NW, SW, NE and SE (*north west*, *south west*, *north east* and *south east*).

```
\begin{NiceTabular}{*{6}{c}}%
[hvlines-except-corners=NE,cell-space-top-limit=3pt]
1\\
1&1\\
1&2&1\\
1&3&3&1\\
1&4&6&4&1\\
1&5&10&10&5&1
\end{NiceTabular}
```

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

5.6 The command `\diagbox`

The command `\diagbox` (inspired by the package `diagbox`), allows, when it is used in a cell, to slash that cell diagonally downwards.³

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

$\begin{smallmatrix} y \\ x \end{smallmatrix}$	<i>e</i>	<i>a</i>	<i>b</i>	<i>c</i>
<i>e</i>	<i>e</i>	<i>a</i>	<i>b</i>	<i>c</i>
<i>a</i>	<i>a</i>	<i>e</i>	<i>c</i>	<i>b</i>
<i>b</i>	<i>b</i>	<i>c</i>	<i>e</i>	<i>a</i>
<i>c</i>	<i>c</i>	<i>b</i>	<i>a</i>	<i>e</i>

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

³The author of this document considers that type of construction as graphically poor.

5.7 Dotted rules

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

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

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

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

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

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

It's possible to change in `nicematrix` the letter used to specify a vertical dotted line with the option `letter-for-dotted-lines` available in `\NiceMatrixOptions`.

Remark : In the package `array` (on which the package `nicematrix` relies), horizontal and vertical rules make the array larger or wider by a quantity equal to the width of the rule⁴. 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 example with the command `\hdotsfor`).
- The package `colortbl` constructs the array row by row, alternating colored rectangles, rules and contents of the cells. The resulting PDF is difficult to interpret by some PDF viewers and may lead to artefacts on the screen.
 - Some rules seem to disappear. 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.

⁴In fact, this is true only for `\hline` and “|” but not for `\cline`: cf p. 6

6.2 The tools of nicematrix in the code-before

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

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

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

- The command `\cellcolor` takes its name from the command `\cellcolor` of `colortbl`.
This command takes in as mandatory arguments a color and a list of cells, each of which with the format $i-j$ where i is the number of row and j the number of column of the cell.

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

a	b	c
e	f	g
h	i	j

A command `\cellcolor` generates only one instruction `fill` (coded `f`) in the resulting PDF.

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

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

a	b	c
e	f	g
h	i	j

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

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

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

A command `\rowcolor` generates only one instruction `fill` (coded `f`) in the resulting PDF.

- 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`⁵. The *s* emphasizes the fact that there is *two* colors. This command colors alternately the rows of the tabular, beginning with the row whose number is given in first (mandatory) argument. The two other (mandatory) arguments are the colors.

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

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

New 5.2 There is a key `respect-blocks` for the instruction `\rowcolors`. With that key, the “rows” alternately colored may extend over several rows if they have to incorporate blocks.

```
\begin{NiceTabular}{\lr}[hvlines,code-before = 
\rowcolors{1}{blue!10}{}[respect-blocks]]
\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.

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

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

We have used the key `r` which aligns all the columns rightwards (cf. p. 25).

One should remark that these commands are compatible with the commands of `booktabs` (`\toprule`, `\midrule`, `\bottomrule`, etc).

⁵The command `\rowcolors` of `xcolor` is available when `xcolor` is loaded with the option `table`.

```

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

```

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

We have used the type of column S of siunitx.

6.3 Color tools with the syntax of colortbl

It's possible to access the preceding tools with a syntax close to the syntax of colortbl. For that, one must use the key `colortbl-like` in the current environment.⁶

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).

```

\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

Each instruction `\cellcolor`, `\rowcolor` or `\columncolor` will generate an instruction `fill` (coded **f**) in the resulting PDF. In cases of juxtaposed colored rectangles, one may have a thin

⁶As of now, this key is not available in `\NiceMatrixOptions`.

white color line in some PDF viewers⁷ (between the two first columns in the above example). In you want to avoid this problem, you should use the tools in the `code-before`. That's what we do with the following code.

```
\begin{NiceTabular}[colortbl-like]{ccc}%
  [code-before = \columncolor{blue!15}{1,2}]
\toprule
\rowcolor{red!15}
Last name & First name & Birth day \\
\midrule
Achard & Jacques & 5 juin 1962 \\
Lefebvre & Mathilde & 23 mai 1988 \\
Vanesse & Stephany & 30 octobre 1994 \\
Dupont & Chantal & 15 janvier 1998 \\
\bottomrule
\end{NiceTabular}
```

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

7 The width of the columns

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

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

Paris	New York	Madrid
Berlin	London	Roma
Rio	Tokyo	Oslo

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

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

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

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

It's possible to give the special value `auto` to the option `columns-width`: all the columns of the array will have a width equal to the widest cell of the array.⁸

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

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

⁷For example SumatraPDF, which uses MuPDF of Artifex Software, or PDF.js used by Firefox.

⁸The result is achieved with only one compilation (but PGF/Tikz will have written informations in the `.aux` file and a message requiring a second compilation will appear).

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

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

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

But it's also possible to fix a zone where all the matrices will have their columns of the same width, equal to the widest cell of all the matrices. This construction uses the environment `{NiceMatrixBlock}` with the option `auto-columns-width`⁹. The environment `{NiceMatrixBlock}` has no direct link with the command `\Block` presented previously in this document (cf. p. 4).

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

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

Several compilations may be necessary to achieve the job.

8 The exterior rows and columns

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

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

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

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

The dotted lines have been drawn with the tools presented p. 16.

⁹At this time, this is the only usage of the environment `{NiceMatrixBlock}` but it may have other usages in the future.

We have several remarks to do.

- For the environments with an explicit preamble (i.e. `{NiceArray}` and its variants), no letter must be given in that preamble for the potential first column and the potential last column: they will automatically (and necessarily) be of type `r` for the first column and `l` for the last one.
- One may wonder how `nicematrix` determines the number of rows and columns which are needed for the composition of the “last row” and “last column”.
 - For the environments with explicit preamble, like `{NiceTabular}` and `{pNiceArray}`, the number of columns can obviously be computed from the preamble.
 - When the option `light-syntax` (cf. p. 27) is used, `nicematrix` has, in any case, to load the whole body of the environment (and that’s why it’s not possible to put verbatim material in the array with the option `light-syntax`). The analysis of this whole body gives the number of rows (but not the number of columns).
 - In the other cases, `nicematrix` compute the number of rows and columns during the first compilation and write the result in the `aux` file for the next run.

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

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

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

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

Remarks

- As shown in the previous example, the horizontal and vertical rules doesn’t extend in the exterior rows and columns.
However, if one wishes to define new specifiers for columns in order to draw vertical rules (for example thicker than the standard rules), he should consider the command `\OnlyMainNiceMatrix` described on page 32.
- 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.

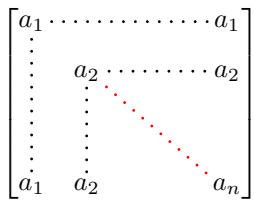
- Logically, the potential option `columns-width` (described p. 13) doesn't apply to the “first column” and “last column”.
- For technical reasons, it's not possible to use the option of the command `\` after the “first row” or before the “last row” (the placement of the delimiters would be wrong).

9 The continuous dotted lines

Inside the environments of the package `nicematrix`, new commands are defined: `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, and `\iddots`. These commands are intended to be used in place of `\dots`, `\cdots`, `\vdots`, `\ddots` and `\iddots`.¹⁰

Each of them must be used alone in the cell of the array and it draws a dotted line between the first non-empty cells¹¹ 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`.¹²

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



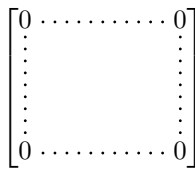
In order to represent the null matrix, one can use the following code:

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



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

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



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

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

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



¹⁰The 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`.

¹¹The precise definition of a “non-empty cell” is given below (cf. p. 33).

¹²It's also possible to change the color of all these dotted lines with the option `xdots/color` (`xdots` to remind that it works for `\Cdots`, `\Ldots`, `\Vdots`, etc.): cf. p. 19.

There are also other means to change the size of the matrix. Someone might want to use the optional argument of the command `\l` for the vertical dimension and a command `\hspace*` in a cell for the horizontal dimension.¹³

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

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

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

9.1 The option `nullify-dots`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

¹³In `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. 13

```

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

```

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

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

```

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

```

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

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

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

```

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

```

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

9.3 How to generate the continuous dotted lines transparently

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`.¹⁴

¹⁴The 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`¹⁰ 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 & & \ddots & & \vdots & \\
0 & & \cdots & & 0 & & & & 1 & \\
\end{pmatrix}
```

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

9.4 The labels of the dotted lines

The commands `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots` and `\Hdotsfor` (and the command `\line` in the `code-after` which is described p. 21) accept two optional arguments specified by the tokens `_` and `^` for labels positionned below and above the line. The arguments are composed in math mode with `\scriptstyle`.

```
$\begin{bNiceMatrix}
1 & & \hspace*{1cm} & & & & & 0 & \ll[8mm]
& & \Ddots^{\text{times}} & & & & & & \\
0 & & & & & & & 1 & \\
\end{bNiceMatrix}$
```

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

9.5 Customization of the dotted lines

The dotted lines drawn by `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots` and `\Hdotsfor` (and by the command `\line` in the `code-after` which is described p. 21) may be customized by three options (specified between square brackets after the command):

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

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

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

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

The option `xdots/color`

The option `xdots/color` fixes the color of the dotted line. However, one should remark that the dotted lines drawn in the exterior rows and columns have a special treatment: cf. p. 14.

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).¹⁵

```
\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 & & & \\
\Vdots & & & & & & & \\
0      & \Cdots & & & 0      & b      & a      & \\
\end{pNiceMatrix}
```

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

9.6 The dotted lines and the rules

The dotted lines determine virtual blocks which have the same behaviour regarding the rules (the rules specified by the specifier `|` in the preamble and by the keys `hlines`, `vlines`, `hvlines` and `hvlines-except-corners` are not drawn within the blocks).

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

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

10 The code-after

The option `code-after` may be used to give some code that will be executed after the construction of the matrix.¹⁶

¹⁵The 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.

¹⁶There is also a key `code-before` described p. 10.

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

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

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

For the legibility of the code, an alternative syntax is provided: it's possible to give the instructions of the `\code-after` at the end of the environment, after the keyword `\CodeAfter`¹⁷. For an example, cf. p. 38.

11 The notes in the tabulars

11.1 The footnotes

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

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

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

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

11.2 The notes of tabular

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

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

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

¹⁷In some circonsstancies, one must put `\omit \CodeAfter`. `\omit` is a keyword of TeX which cancels the pattern of the current cell.

```
\bottomrule
\end{NiceTabular}
```

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

^a Achard is an old family of the Poitou.

^b 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`).
- **New 5.4** There is a key `tabularnote` which provides a way to insert some texte 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. 23. This table has been composed with the following code.

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

Table 1: Use of `\tablarnote`^a

Last name	First name	Length of life
Churchill	Wiston	91
Nightingale ^{b,c}	Florence	90
Schoelcher	Victor	89 ^d
Touchet	Marie	89
Wallis	John	87

Some text before the notes.

^a It's possible to put a note in the caption.
^b Considered as the first nurse of history.
^c Nicknamed “the Lady with the Lamp”.
^d The label of the note is overlapping.

11.3 Customisation of the tabular notes

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

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

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

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

We detail these keys.

- The key `notes/para` requires the composition of the notes (at the end of the tabular) in a single paragraph.
Initial value: `false`
That key is also available within a given environment.

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

Initial value: `false`

That key is also available within a given environment.

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

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

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

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

Initial value: `#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: `#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. 23).

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

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

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

Initial value: `empty`

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

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


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

For an example of customization of the tabular notes, see p. 34.

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:

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

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

12 Other features

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

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

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

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

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

12.2 Alignment option in `{NiceMatrix}`

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

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

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

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

```

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

12.4 The option `small`

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

```

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

```

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

One should note that the environment `{NiceMatrix}` with the option `small` is not composed *exactly* as the environment `{smallmatrix}`. Indeed, all the environments of `nicematrix` are constructed upon `{array}` (of the package `array`) whereas the environment `{smallmatrix}` is constructed directly with an `\halign` of TeX.

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

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

12.5 The counters `iRow` and `jCol`

In the cells of the array, it’s possible to use the LaTeX counters `iRow` and `jCol` which represent the number of the current row and the number of the current column¹⁸. Of course, the user must not

¹⁸We recall that the exterior “first row” (if it exists) has the number 0 and that the exterior “first column” (if it exists) has also the number 0.

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

In the `code-before` (cf. p. 10) and in the `code-after` (cf. p. 20), `iRow` represents the total number of rows (excepted the potential exterior rows) and `jCol` represents the total number of columns (excepted the potential exterior columns).

```

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

```

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

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

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

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

```

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

```

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

12.6 The option `light-syntax`

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

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

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

```

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

```

$$\begin{matrix} & a & b \\ a & \left[\begin{matrix} 2 \cos a & \cos a + \cos b \end{matrix} \right] \\ b & \left[\begin{matrix} \cos a + \cos b & 2 \cos b \end{matrix} \right] \end{matrix}$$

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.¹⁹

¹⁹The reason is that, when the option `light-syntax` is used, the whole content of the environment is loaded as a TeX argument to be analyzed. The environment doesn't behave in that case as a standard environment of LaTeX which only put TeX commands before and after the content.

12.7 The environment `{NiceArrayWithDelims}`

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

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

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

13 Use of Tikz with nicematrix

13.1 The nodes corresponding to the contents of the cells

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

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

In the environment with the number n , the node of the row i and column j has for name `nm-n-i-j`. The command `\NiceMatrixLastEnv` provides the number of the last environment of `nicematrix` (for LaTeX, it's a "fully expandable" command and not a counter).

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

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

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

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

In the `code-after`, and if Tikz is loaded, the things are easier. One may design the nodes with the form $i-j$: there is no need to indicate the environment which is of course the current environment.

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

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

In the following example, we have underlined all the nodes of the matrix (we explain below the technic used : cf. p. 38).

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

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`.²⁰

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

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

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

The names of the “large nodes” are constructed by adding the suffix “-large” to the names of the “normal nodes”. In the following example, we have underlined the “large nodes”. We consider that this example is self-explanatory.²¹

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

The “large nodes” of the first column and last column may appear too small for some usage. That’s why it’s possible to use the options `left-margin` and `right-margin` to add space on both sides of the array and also space in the “large nodes” of the first column and last column. In the following example, we have used the options `left-margin` and `right-margin`.²²

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

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

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

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

Here is an array composed with the following code:

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

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

²⁰There is also an option `create-extra-nodes` which is an alias for the conjunction of `create-medium-nodes` and `create-large-nodes`.

²¹There is no “large nodes” created in the exterior rows and columns (for these rows and columns, cf. p. 14).

²²The options `left-margin` and `right-margin` take dimensions as values but, if no value is given, the default value is used, which is `\arraycolsep` (by default: 5 pt). There is also an option `margin` to fix both `left-margin` and `right-margin` to the same value.

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

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

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

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

13.3 The “row-nodes” and the “col-nodes”

The package `nicematrix` creates a PGF/Tikz node indicating the potential position of each horizontal rule (with the names `row-i`) and each vertical rule (with the names `col-j`), as described in the following figure. These nodes are available in the `code-before` and the `code-after`.

row-1	rose	tulipe	lys	
row-2	arum	iris	violette	
row-3	muguet	dahlia	souci	
row-4	col-1	col-2	col-3	col-4

If we use Tikz (we remind that `nicematrix` does not load Tikz by default), we can access (in the `code-before` and the `code-after`) to the intersection of the horizontal rule *i* and the vertical rule *j* with the syntax `(row-i-|col-j)`.

```
\[ \begin{NiceMatrix}[
  code-before =
  {
    \tikz \draw [fill = red!15]
      (row-7-|col-4) -- (row-8-|col-4) -- (row-8-|col-5) --
      (row-9-|col-5) -- (row-9-|col-6) |- cycle ;
  }
]
1 \\\
1 & 1 \\\
1 & 2 & 1 \\\
1 & 3 & 3 & 1 \\\
1 & 4 & 6 & 4 & 1 \\\
1 & 5 & 10 & 10 & 5 & 1 \\\
1 & 6 & 15 & 20 & 15 & 6 & 1 \\\
1 & 7 & 21 & 35 & 35 & 21 & 7 & 1 \\\
1 & 8 & 28 & 56 & 70 & 56 & 28 & 8 & 1
\end{NiceMatrix} \]
```

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

14 API for the developpers

The package `nicematrix` provides two variables which are internal but public²³:

- **New 5.2** `\g_nicematrix_code_before_tl` ;
- `\g_nicematrix_code_after_tl`.

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

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

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

```
ExplSyntaxOn
\cs_new_protected:Nn \__pantigny_hatchcell:nnn
{
  \begin { tikzpicture }
  \fill [ pattern = north~west~lines , pattern~color = #3 ]
    ( row - #1 -| col - #2 )
    rectangle
    ( row - \int_eval:n { #1 + 1 } -| col - \int_eval:n { #2 + 1 } ) ;
  \end { tikzpicture }
}

\NewDocumentCommand \hatchcell { ! O { black } }
{
  \tl_gput_right:Nx \g_nicematrix_code_before_tl
  {
    \__pantigny_hatchcell:nnn
    { \int_use:c { c@iRow } }
    { \int_use:c { c@jCol } }
    { #1 }
  }
}
\ExplSyntaxOff
```

Here is an example of use:

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

Tokyo	Paris	London
Lima	Oslo	Miami
Los Angeles	Madrid	Roma

²³According to the LaTeX3 conventions, each variable with name beginning with `\g_nicematrix` ou `\l_nicematrix` is public and each variable with name beginning with `\g__nicematrix` or `\l__nicematrix` is private.

15 Technical remarks

15.1 Definition of new column types

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

For example, one may wish to define a new column type `?` in order to draw a (black) heavy rule of width 1 pt. The following definition will do the job²⁴:

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

The heavy vertical rule won't extend in the exterior rows.²⁵

```
\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}$
```

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

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

15.2 Diagonal lines

By default, all the diagonal lines²⁶ of a same array are “parallelized”. That means that the first diagonal line is drawn and, then, the other lines are drawn parallel to the first one (by rotation around the left-most extremity of the line). That's why the position of the instructions `\Ddots` in the array can have a marked effect on the final result.

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

Example with parallelization (default):

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

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

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

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

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

The same example without parallelization:

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

²⁴The command `\vrule` is a TeX (and not LaTeX) command.

²⁵Of course, such rule is defined by the classical technics of `nicematrix` and, for this reason, won't cross the double rules of `\hline\hline`.

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

New 5.3 It's possible to specify the instruction `\Ddots` which will be drawn first (and which will be used to draw the other diagonal dotted line 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 cells on both sides. However, an empty cell is not necessarily a cell with no TeX content (that is to say a cell with no token between the two ampersands `&`). Indeed, a cell which only contains `\hspace*{1cm}` may be considered as empty.

For `nicematrix`, the precise rules are as follow.

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

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

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

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

15.4 The option `exterior-arraycolsep`

The environment `{array}` inserts an horizontal space equal to `\arraycolsep` before and after each column. In particular, there is a space equal to `\arraycolsep` before and after the array. This feature of the environment `{array}` was probably not a good idea²⁷. The environment `{matrix}` of `amsmath` and its variants (`{pmatrix}`, `{vmatrix}`, etc.) of `amsmath` prefer to delete these spaces with explicit instructions `\hspace -\arraycolsep`²⁸. 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`).

²⁷In 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).*

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

16 Examples

16.1 Notes in the tabulars

The tools provided by `nicematrix` for the composition of the tabular notes have been presented in the section 11 p. 21.

Let's consider that we wish to number the notes of a tabular with stars.²⁹

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 ³⁰

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

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

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

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

²⁹Of course, it's realistic only when there is very few notes in the tabular.

³⁰In fact: the value of its argument.

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

*Achard is an old family of the Poitou.

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

16.2 Dotted lines

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

```

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

```

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

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

```

 $\begin{pNiceMatrix}[xdots/shorten=0.9em]$ 
1 & \Cdots & & 1 & \\
\Vdots & & & 0 & \\
& \Iddots & \Iddots & \Vdots & \\
1 & 0 & & \Cdots & 0 \\
\end{pNiceMatrix}

```

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

An example with `\multicolumn`:

```

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

```

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

An example with `\Hdotsfor`:

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

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

An example for the resultant of two polynoms:

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

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

An example for a linear system:

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

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

16.3 Dotted lines which are no longer dotted

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

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

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

16.4 Width of the columns

In the following example, we use `{NiceMatrixBlock}` with the option `auto-columns-width` because we want the same automatic width for all the columns of the matrices.

```
\begin{NiceMatrixBlock}[auto-columns-width]
\NiceMatrixOptions
  { last-col,code-for-last-col = \color{blue}\scriptstyle,light-syntax}
\setlength{\extrarowheight}{1mm}
$\begin{pNiceArray}{cccc:c}
  1 1 1 1 1 {} ;
  2 4 8 16 9 ;
  3 9 27 81 36 ;
  4 16 64 256 100
\end{pNiceArray}$
\medskip
$\begin{pNiceArray}{cccc:c}
  1 1 1 1 1 ;
  0 2 6 14 7 { L_2 \gets -2 L_1 + L_2 } ;
  0 6 24 78 33 { L_3 \gets -3 L_1 + L_3 } ;
  0 12 60 252 96 { L_4 \gets -4 L_1 + L_4 }
\end{pNiceArray}$
...
\end{NiceMatrixBlock}
```

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

16.5 How to highlight cells of the matrix

The following examples require Tikz (by default, nicematrix only loads PGF) and the Tikz library `fit`. The following lines in the preamble of your document do the job:

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

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

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

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

$$\left(\begin{array}{cccc} \boxed{a_{11}} & a_{12} & a_{13} & a_{14} \\ a_{21} & \boxed{a_{22}} & a_{23} & a_{24} \\ a_{31} & a_{32} & \boxed{a_{33}} & a_{34} \\ a_{41} & a_{42} & a_{43} & \boxed{a_{44}} \end{array} \right)$$

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.³¹

³¹For the command `\cline`, see the remark p. 6.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

We obtain the following matrix.

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

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

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

We obtain the following matrix.

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

In the following example, we use the “large nodes” to highlight a zone of the matrix.

```
\begin{pNiceArray}{>{\strut}cccc}[create-large-nodes,margin,extra-margin=2pt]
A_{11} & A_{12} & A_{13} & A_{14} \\
A_{21} & A_{22} & A_{23} & A_{24} \\
A_{31} & A_{32} & A_{33} & A_{34} \\
A_{41} & A_{42} & A_{43} & A_{44}
\CodeAfter
\tikz \path [name suffix = -large,fill = red!15, blend mode = multiply]
(1-1.north west)
|- (2-2.north west)
|- (3-3.north west)
|- (4-4.north west)
|- (4-4.south east)
|- (1-1.north west) ;
\end{pNiceArray}
```


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

16.6 Direct use of the Tikz nodes

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

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

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

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

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

```
$\begin{array}{cc}
&
```

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

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

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

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

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

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

```
\end{NiceMatrixBlock}
```

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

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

17 Implementation

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

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

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

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

Declaration of the package and packages loaded

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

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

<@@=nicematrix>

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

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

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

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

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

We load some packages.

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

```

12 \cs_set_eq:NN \box_use_drop_i:N \box_use_drop:N
13 \cs_set_eq:NN \box_use_drop_ii:N \box_use_drop:N
14 \cs_set_eq:NN \box_use_drop_iii:N \box_use_drop:N
15 \cs_set_eq:NN \box_use_drop_iv:N \box_use_drop:N
16 \cs_set_eq:NN \box_use_drop_v:N \box_use_drop:N
17 \cs_set_eq:NN \box_use_drop_vi:N \box_use_drop:N
18 \cs_set_eq:NN \box_use_drop_vii:N \box_use_drop:N
19 \cs_set_eq:NN \box_use_drop_viii:N \box_use_drop:N
20 \cs_set_eq:NN \box_use_drop_ix:N \box_use_drop:N
21 \cs_set_eq:NN \box_use_drop_x:N \box_use_drop:N
22 \cs_set_eq:NN \box_use_drop_xi:N \box_use_drop:N
23 \cs_set_eq:NN \box_use_drop_xii:N \box_use_drop:N

24 \cs_new_protected:Npn \@@_error:n { \msg_error:nn { nicematrix } }
25 \cs_new_protected:Npn \@@_error:nn { \msg_error:nnn { nicematrix } }
26 \cs_new_protected:Npn \@@_error:nnn { \msg_error:nnnn { nicematrix } }
27 \cs_new_protected:Npn \@@_fatal:n { \msg_fatal:nn { nicematrix } }
28 \cs_new_protected:Npn \@@_fatal:nn { \msg_fatal:nnn { nicematrix } }
29 \cs_new_protected:Npn \@@_msg_new:nn { \msg_new:nnn { nicematrix } }
30 \cs_new_protected:Npn \@@_msg_new:nnn { \msg_new:nnnn { nicematrix } }

31 \cs_new_protected:Npn \@@_msg_redirect_name:nn
32 { \msg_redirect_name:nnn { nicematrix } }

```

Technical definitions

```

33 \bool_new:N \c_@@_in_preamble_bool
34 \bool_set_true:N \c_@@_in_preamble_bool
35 \AtBeginDocument { \bool_set_false:N \c_@@_in_preamble_bool }

36 \bool_new:N \c_@@_booktabs_loaded_bool
37 \bool_new:N \c_@@_enumitem_loaded_bool
38 \bool_new:N \c_@@_tikz_loaded_bool
39 \AtBeginDocument
40 {
41   \ifpackageloaded { booktabs }
42     { \bool_set_true:N \c_@@_booktabs_loaded_bool }
43   { }
44   \ifpackageloaded { enumitem }
45     { \bool_set_true:N \c_@@_enumitem_loaded_bool }
46   { }
47   \ifpackageloaded { tikz }
48   {

```

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.

```

49   \bool_set_true:N \c_@@_tikz_loaded_bool
50   \tl_const:Nn \c_@@_pgfortikzpicture_tl { \exp_not:N \tikzpicture }
51   \tl_const:Nn \c_@@_endpgfortikzpicture_tl { \exp_not:N \endtikzpicture }
52 }
53 {
54   \tl_const:Nn \c_@@_pgfortikzpicture_tl { \exp_not:N \pgfpicture }
55   \tl_const:Nn \c_@@_endpgfortikzpicture_tl { \exp_not:N \endpgfpicture }
56 }
57 }

```

We test whether the current class is revtex4-1 or revtex4-2 because these classes redefines `\array` (of `array`) in a way incompatible with our programming.

```

58 \bool_new:N \c_@@_revtex_bool
59 \@ifclassloaded { revtex4-1 }
60 { \bool_set_true:N \c_@@_revtex_bool }
61 { }
62 \@ifclassloaded { revtex4-2 }
63 { \bool_set_true:N \c_@@_revtex_bool }
64 { }

```

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

```

65 \ProvideDocumentCommand \iddots { }
66 {
67   \mathinner
68   {
69     \tex_mkern:D 1 mu
70     \box_move_up:nn { 1 pt } { \hbox:n { . } }
71     \tex_mkern:D 2 mu
72     \box_move_up:nn { 4 pt } { \hbox:n { . } }
73     \tex_mkern:D 2 mu
74     \box_move_up:nn { 7 pt }
75     { \vbox:n { \kern 7 pt \hbox:n { . } } }
76     \tex_mkern:D 1 mu
77   }
78 }

```

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.

```

79 \AtBeginDocument
80 {
81   \@ifpackageloaded { booktabs }
82   { \iow_now:Nn \mainaux \nicematrix@redefine@check@rerun }
83   { }
84 }
85 \cs_set_protected:Npn \nicematrix@redefine@check@rerun
86 {
87   \cs_set_eq:NN \@@_old_pgful@check@rerun \pgfutil@check@rerun

```

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

```

88   \cs_set_protected:Npn \pgfutil@check@rerun ##1 ##2
89   {
90     \str_if_eq:eeF { nm- } { \tl_range:nnn { ##1 } 1 3 }
91     { \@@_old_pgful@check@rerun { ##1 } { ##2 } }
92   }
93 }

```

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

```

94 \bool_new:N \c_@@_colortbl_loaded_bool
95 \AtBeginDocument
96 {
97   \@ifpackageloaded { colortbl }
98   { \bool_set_true:N \c_@@_colortbl_loaded_bool }
99   { }

```

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. Idem for

```

100 \cs_set_protected:Npn \CT@arc@ { }
101 \cs_set:Npn \arrayrulecolor #1 # { \CT@arc { #1 } }
102 \cs_set:Npn \CT@arc #1 #2
103 {
104   \dim_compare:nNnT \baselineskip = \c_zero_dim \noalign
105   { \cs_gset:Npn \CT@arc@ { \color #1 { #2 } } }
106 }

```

Idem for \CT@drs@.

```

107 \cs_set_protected:Npn \CT@drsc@ { }
108 \cs_set:Npn \doublerulesepcolor #1 # { \CT@drs { #1 } }
109 \cs_set:Npn \CT@drs #1 #2
110 {
111   \dim_compare:nNnT \baselineskip = \c_zero_dim \noalign
112   { \cs_gset:Npn \CT@drsc@ { \color #1 { #2 } } }
113 }
114 \cs_set:Npn \hline
115 {
116   \noalign { \ifnum 0 = ` } \fi
117   \cs_set_eq:NN \hskip \vskip
118   \cs_set_eq:NN \vrule \hrule
119   \cs_set_eq:NN \@width \@height
120   { \CT@arc@ \vline }
121   \futurelet \reserved@a
122   \@xhline
123 }
124 }
125 }

```

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.

```

126 \cs_set:Npn \@@_standard_cline #1 { \@@_standard_cline:w #1 \q_stop }
127 \cs_set:Npn \@@_standard_cline:w #1-#2 \q_stop
128 {
129   \int_compare:nNnT \l_@@_first_col_int = 0 { \omit & }
130   \int_compare:nNnT { #1 } > 1 { \multispan { \@@_pred:n { #1 } } & }
131   \multispan { \int_eval:n { #2 - #1 + 1 } }
132   {
133     \CT@arc@
134     \leaders \hrule \@height \arrayrulewidth \hfill

```

The following \skip_horizontal:N \c_zero_dim is to prevent a potential \unskip to delete the \leaders³²

```

135 \skip_horizontal:N \c_zero_dim
136 }

```

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.

```

137 \everycr { }
138 \cr
139 \noalign { \skip_vertical:N -\arrayrulewidth }
140 }

```

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.

```

141 \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.

```

142 { \@@_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*.

³²See question 99041 on TeX StackExchange.

```

143 \cs_set:Npn \@@_cline_i:nn #1 #2 { \@@_cline_i:w #1-#2 \q_stop }
144 \cs_set:Npn \@@_cline_i:w #1-#2-#3 \q_stop
145 {

```

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).

```

146 \int_compare:nNnT { #1 } < { #2 }
147 { \multispan { \int_eval:n { #2 - #1 } } & }
148 \multispan { \int_eval:n { #3 - #2 + 1 } }
149 {
150 \CT@arc@
151 \leaders \hrule \@height \arrayrulewidth \hfill
152 \skip_horizontal:N \c_zero_dim
153 }

```

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

```

154 \peek_meaning_remove_ignore_spaces:NTF \cline
155 { & \@@_cline_i:en { \@@_succ:n { #3 } } }
156 { \everycr { } \cr }
157 }
158 \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.

```

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

```

The following command is a small shortcut.

```

161 \cs_new:Npn \@@_math_toggle_token:
162 { \bool_if:NF \l_@@_NiceTabular_bool \c_math_toggle_token }

```

```

163 \cs_new_protected:Npn \@@_set_CT@arc@:
164 { \peek_meaning:NTF [ \@@_set_CT@arc@_i: \@@_set_CT@arc@_ii: }
165 \cs_new_protected:Npn \@@_set_CT@arc@_i: [ #1 ] #2 \q_stop
166 { \cs_set:Npn \CT@arc@ { \color [ #1 ] { #2 } } }
167 \cs_new_protected:Npn \@@_set_CT@arc@_ii: #1 \q_stop
168 { \cs_set:Npn \CT@arc@ { \color { #1 } } }

```

The column S of siunitx

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

```

169 \bool_new:N \c_@@_siunitx_loaded_bool
170 \AtBeginDocument
171 {
172 \ifpackageloaded { siunitx }
173 { \bool_set_true:N \c_@@_siunitx_loaded_bool }
174 { }
175 }

```

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:`.

```
176 \cs_set_protected:Npn \@@_adapt_S_column:
177 {
178   \bool_if:NT \c_@@_siunitx_loaded_bool
179   {
180     \group_begin:
181     \@temptokena = { }
```

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

```
182   \cs_set_eq:NN \NC@find \prg_do_nothing:
183   \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`).

```
184   \tl_gset:NV \g_tmpa_tl \@temptokena
185   \group_end:
186   \tl_new:N \c_@@_table_collect_begin_tl
187   \tl_set:Nx \l_tmpa_tl { \tl_item:Nn \g_tmpa_tl 2 }
188   \tl_gset:Nx \c_@@_table_collect_begin_tl { \tl_item:Nn \l_tmpa_tl 1 }
189   \tl_new:N \c_@@_table_print_tl
190   \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).

```
191   \cs_gset_eq:NN \@@_adapt_S_column: \prg_do_nothing:
192 }
193 }
```

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

```
194 \AtBeginDocument
195 {
196   \bool_if:nTF { ! \c_@@_siunitx_loaded_bool }
197   { \cs_set_eq:NN \@@_renew_NC@rewrite@S: \prg_do_nothing: }
198   {
199     \cs_new_protected:Npn \@@_renew_NC@rewrite@S:
200     {
201       \renewcommand*{\NC@rewrite@S}[1] []
202       {
203         \@temptokena \exp_after:wN
```

```

204         {
205             \tex_the:D \@temptokena
206             > { \c_@@_table_collect_begin_tl S {##1} }
\@@_true_c: will be replaced statically by c at the end of the construction of the preamble.
207             \@@_true_c:
208             < { \c_@@_table_print_tl \@@_end_Cell: }
209         }
210     \NC@find
211 }
212 }
213 }
214 }

```

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

```

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

```

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

```

216 \cs_new_protected:Npn \@@_provide_pgfsyspdfmark:
217 {
218     \iow_now:Nn \@mainaux
219     {
220         \ExplSyntaxOn
221         \cs_if_free:NT \pgfsyspdfmark
222         { \cs_set_eq:NN \pgfsyspdfmark \@gobblethree }
223         \ExplSyntaxOff
224     }
225     \cs_gset_eq:NN \@@_provide_pgfsyspdfmark: \prg_do_nothing:
226 }

```

Parameters

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

```

227 \bool_new:N \c_@@_define_L_C_R_bool
228 \cs_new_protected:Npn \@@_define_L_C_R:
229 {
230     \newcolumntype L l
231     \newcolumntype C c
232     \newcolumntype R r
233 }

```

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.

```

234 \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).

```

235 \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.

```
236 \NewExpandableDocumentCommand \NiceMatrixLastEnv { }
237 { \int_use:N \g_@@_env_int }
```

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

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

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

```
240 \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).

```
241 \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.

```
242 \tl_new:N \l_@@_cell_type_tl
243 \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_width_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_width_dim`.

```
244 \dim_new:N \g_@@_blocks_width_dim
```

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

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

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

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

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

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

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

```
248 \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.

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

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

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

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

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

```

252 \cs_new_protected:Npn \@@_test_if_math_mode:
253 {
254   \if_mode_math: \else:
255     \@@_fatal:n { Outside-math-mode }
256   \fi:
257 }

```

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

```

258 \colorlet { nicematrix-last-col } { . }
259 \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*).

```

260 \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*.

```

261 \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.

```

262 \cs_new:Npn \@@_full_name_env:
263 {
264   \str_if_eq:VnTF \g_@@_com_or_env_str { command }
265     { command \space \c_backslash_str \g_@@_name_env_str }
266     { environment \space \{ \g_@@_name_env_str \} }
267 }

```

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

```

268 \tl_new:N \g_nicematrix_code_after_tl

```

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

```

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

```

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).

```

273 \bool_new:N \g_@@_row_of_col_done_bool

```

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

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

```
274 \tl_new:N \l_@@_code_before_tl
275 \bool_new:N \l_@@_code_before_bool
```

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

```
276 \dim_new:N \l_@@_x_initial_dim
277 \dim_new:N \l_@@_y_initial_dim
278 \dim_new:N \l_@@_x_final_dim
279 \dim_new:N \l_@@_y_final_dim
```

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

```
280 \dim_zero_new:N \l_tmpc_dim
281 \dim_zero_new:N \l_tmpd_dim
```

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

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

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

```
283 \dim_new:N \@@_old_arraycolsep_dim
```

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

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

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

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

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

```
286 \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}`.

```
287 \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}`.

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

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

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

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

Used for the color of the blocks.

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

The parameter of position of the label of a block (c, r or l).

```
291 \tl_new:N \l_@@_pos_of_block_tl
292 \tl_set:Nn \l_@@_pos_of_block_tl { c }
```

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

```
293 \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.

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

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.

```
295 \int_new:N \l_@@_first_row_int
296 \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.

```
297 \int_new:N \l_@@_first_col_int
298 \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).

```
299 \int_new:N \l_@@_last_row_int
300 \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”.³³

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

Idem for `\l_@@_last_col_without_value_bool`

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

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

- **Last column**

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

```
303 \int_new:N \l_@@_last_col_int
304 \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.

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

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

The command `\tablarnote`

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

```
306 \newcounter { tablarnote }
```

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

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

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

```
308 \tl_new:N \l_@@_tablarnote_tl
```

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

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

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

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

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

```
311 \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`.

```
312 \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`.

```
313 \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`).

```
314 \AtBeginDocument
315 {
316   \bool_if:nTF { ! \c_@@_enumitem_loaded_bool }
317   {
318     \NewDocumentCommand \tabularnote { m }
319     { \@@_error:n { enumitem-not-loaded } }
320   }
321   {
```

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.

```
322   \newlist { tabularnotes } { enumerate } { 1 }
323   \setlist [ tabularnotes ]
324   {
325     topsep = 0pt ,
326     noitemsep ,
327     leftmargin = * ,
328     align = left ,
329     labelsep = 0pt ,
330     label =
331     \@@_notes_label_in_list:n { \@@_notes_style:n { tabularnotesi } } ,
332   }
333   \newlist { tabularnotes* } { enumerate* } { 1 }
334   \setlist [ tabularnotes* ]
335   {
336     afterlabel = \nobreak ,
337     itemjoin = \quad ,
338     label =
339     \@@_notes_label_in_list:n { \@@_notes_style:n { tabularnotes*i } }
340   }
```

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.³⁴

```
341   \NewDocumentCommand \tabularnote { m }
342   {
343     \bool_if:nTF { ! \l_@@_NiceArray_bool && \l_@@_in_env_bool }
344     { \@@_error:n { tabularnote-forbidden } }
345     {
```

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

```
346   \int_incr:N \l_@@_number_of_notes_int
```

³⁴We should try to find a solution to that problem.

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

```

347         \seq_gput_right:Nn \g_@@_tabularnotes_seq { #1 }
348         \peek_meaning:NF \tabularnote
349     {

```

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.

```

350         \hbox_set:Nn \l_tmpa_box
351     {

```

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

```

352         @@_notes_label_in_tabular:n
353     {
354         \stepcounter { tabularnote }
355         @@_notes_style:n { tabularnote }
356         \prg_replicate:nn { \l_@@_number_of_notes_int - 1 }
357     {
358         ,
359         \stepcounter { tabularnote }
360         @@_notes_style:n { tabularnote }
361     }
362 }
363 }

```

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.

```

364         \addtocounter { tabularnote } { -1 }
365         \refstepcounter { tabularnote }
366         \int_zero:N \l_@@_number_of_notes_int
367         \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).

```

368         \skip_horizontal:n { \box_wd:N \l_tmpa_box }
369     }
370 }
371 }
372 }
373 }

```

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.

```

374 \cs_new_protected:Npn \@@_pgf_rect_node:nnnnn #1 #2 #3 #4 #5
375 {
376     \begin { pgfscope }
377     \pgfset
378     {
379         outer~sep = \c_zero_dim ,
380         inner~sep = \c_zero_dim ,
381         minimum~size = \c_zero_dim
382     }
383     \pgftransformshift { \pgfpoint { 0.5 * ( #2 + #4 ) } { 0.5 * ( #3 + #5 ) } }
384     \pgfnode
385     { rectangle }

```

```

386     { center }
387     {
388         \vbox_to_ht:nn
389         { \dim_abs:n { #5 - #3 } }
390         {
391             \vfill
392             \hbox_to_wd:nn { \dim_abs:n { #4 - #2 } } { }
393         }
394     }
395     { #1 }
396     { }
397 \end { pgfscope }
398 }

```

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

```

399 \cs_new_protected:Npn \@@_pgf_rect_node:nnn #1 #2 #3
400 {
401     \begin { pgfscope }
402     \pgfset
403     {
404         outer~sep = \c_zero_dim ,
405         inner~sep = \c_zero_dim ,
406         minimum~size = \c_zero_dim
407     }
408     \pgftransformshift { \pgfpointscale { 0.5 } { \pgfpointadd { #2 } { #3 } } }
409     \pgfpointdiff { #3 } { #2 }
410     \pgfgetlastxy \l_tmpa_dim \l_tmpb_dim
411     \pgfnode
412     { rectangle }
413     { center }
414     {
415         \vbox_to_ht:nn
416         { \dim_abs:n \l_tmpb_dim }
417         { \vfill \hbox_to_wd:nn { \dim_abs:n \l_tmpa_dim } { } }
418     }
419     { #1 }
420     { }
421 \end { pgfscope }
422 }

```

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.

```

423 \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`.

```

424 \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`).

```

425 \dim_new:N \l_@@_cell_space_top_limit_dim
426 \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.

```
427 \dim_new:N \l_@@_inter_dots_dim
428 \dim_set:Nn \l_@@_inter_dots_dim { 0.45 em }
```

The following dimension is the minimal distance between a node (in fact an anchor of that node) and a dotted line (we say “minimal” because, by definition, a dotted line is not a continuous line and, therefore, this distance may vary a little).

```
429 \dim_new:N \l_@@_xdots_shorten_dim
430 \dim_set:Nn \l_@@_xdots_shorten_dim { 0.3 em }
```

The following dimension is the radius of the dots for the dotted lines (when `line-style` is equal to `standard`, which is the initial value). The initial value is 0.53 pt but it will be changed if the option `small` is used.

```
431 \dim_new:N \l_@@_radius_dim
432 \dim_set:Nn \l_@@_radius_dim { 0.53 pt }
```

The token list `\l_@@_xdots_line_style_tl` corresponds to the option `tikz` of the commands `\Cdots`, `\Ldots`, etc. and of the options `line-style` for the environments and `\NiceMatrixOptions`. The constant `\c_@@_standard_tl` will be used in some tests.

```
433 \tl_new:N \l_@@_xdots_line_style_tl
434 \tl_const:Nn \c_@@_standard_tl { standard }
435 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
```

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

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

The string `\l_@@_baseline_str` may contain one of the three values `t`, `c` or `b` as in the option of the environment `{array}`. However, it may also contain an integer (which represents the number of the row to which align the array).

```
437 \str_new:N \l_@@_baseline_str
438 \tl_set:Nn \l_@@_baseline_str c
```

The flag `\l_@@_exterior_arraycolsep_bool` corresponds to the option `exterior-arraycolsep`. If this option is set, a space equal to `\arraycolsep` will be put on both sides of an environment `{NiceArray}` (as it is done in `{array}` of `array`).

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

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

```
440 \bool_new:N \l_@@_parallelize_diags_bool
441 \bool_set_true:N \l_@@_parallelize_diags_bool
```

If the flag `\l_@@_vlines_bool` is raised, horizontal space will be reserved in the the preamble of the array (for the vertical rules) and, after the construction of the array, the vertical rules will be drawn.

```
442 \bool_new:N \l_@@_vlines_bool
```

If the flag `\l_@@_hlines_bool` is raised, vertical space will be reserved between the rows of the array (for the horizontal rules) and, after the construction of the array, the vertical rules will be drawn.

```
443 \bool_new:N \l_@@_hlines_bool
```

The flag `\l_@@_except_corners_bool` will be raised when the key `except-corners` will be used. In that case, the corners will be computed before we draw rules and the rules won’t be drawn in the corners. As expected, the key `hlines-except-corners` raises the key `except-corners`.

```
444 \clist_new:N \l_@@_except_corners_clist
```

```

445 \dim_new:N \l_@@_notes_above_space_dim
446 \dim_set:Nn \l_@@_notes_above_space_dim { 1 mm }

```

The flag `\l_@@_nullify_dots_bool` corresponds to the option `nullify-dots`. When the flag is down, the instructions like `\vdots` are inserted within a `\hphantom` (and so the constructed matrix has exactly the same size as a matrix constructed with the classical `{matrix}` and `\ldots`, `\vdots`, etc.).

```

447 \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).

```

448 \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.

```

449 \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”.

```

450 \bool_new:N \l_@@_medium_nodes_bool
451 \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`).

```

452 \dim_new:N \l_@@_left_margin_dim
453 \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`.

```

454 \dim_new:N \l_@@_extra_left_margin_dim
455 \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.

```

456 \tl_new:N \l_@@_end_of_row_tl
457 \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 “:”.

```

458 \tl_new:N \l_@@_xdots_color_tl

```

Sometimes, we want to have several arrays vertically juxtaposed in order to have an alignment of the columns of these arrays. To achieve this goal, one may wish to use the same width for all the columns (for example with the option `columns-width` or the option `auto-columns-width` of the environment `{NiceMatrixBlock}`). However, even if we use the same type of delimiters, the width of the delimiters may be different from an array to another because the width of the delimiter is fonction of its size. That’s why we create an option called `max-delimiter-width` which will give to the delimiters the width of a delimiter (of the same type) of big size. The following boolean corresponds to this option.

```

459 \bool_new:N \l_@@_max_delimiter_width_bool

```

```

460 \keys_define:nn { NiceMatrix / xdots }
461 {
462   line-style .code:n =
463     {
464       \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.

```

465     { \cs_if_exist_p:N \tikzpicture }
466     { \str_if_eq_p:nn { #1 } { standard } }
467     { \tl_set:Nn \l_@@_xdots_line_style_tl { #1 } }
468     { \@@_error:n { bad-option-for-line-style } }
469   } ,
470   line-style .value_required:n = true ,
471   color .tl_set:N = \l_@@_xdots_color_tl ,
472   color .value_required:n = true ,
473   shorten .dim_set:N = \l_@@_xdots_shorten_dim ,
474   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 `_`.

```

475   down .tl_set:N = \l_@@_xdots_down_tl ,
476   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).

```

477   draw-first .code:n = \prg_do_nothing: ,
478   unknown .code:n = \@@_error:n { Unknown-option-for-xdots }
479 }

```

```

480 \keys_define:nn { NiceMatrix / rules }
481 {
482   color .tl_set:N = \l_@@_rules_color_tl ,
483   color .value_required:n = true ,
484   width .dim_set:N = \arrayrulewidth ,
485   width .value_required:n = true
486 }

```

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

```

487 \keys_define:nn { NiceMatrix / Global }
488 {
489   standard-cline .bool_set:N = \l_@@_standard_cline_bool ,
490   standard-cline .default:n = true ,
491   cell-space-top-limit .dim_set:N = \l_@@_cell_space_top_limit_dim ,
492   cell-space-top-limit .value_required:n = true ,
493   cell-space-bottom-limit .dim_set:N = \l_@@_cell_space_bottom_limit_dim ,
494   cell-space-bottom-limit .value_required:n = true ,
495   xdots .code:n = \keys_set:nn { NiceMatrix / xdots } { #1 } ,
496   max-delimiter-width .bool_set:N = \l_@@_max_delimiter_width_bool ,
497   light-syntax .bool_set:N = \l_@@_light_syntax_bool ,
498   light-syntax .default:n = true ,
499   end-of-row .tl_set:N = \l_@@_end_of_row_tl ,
500   end-of-row .value_required:n = true ,
501   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
502   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
503   last-row .int_set:N = \l_@@_last_row_int ,
504   last-row .default:n = -1 ,
505   code-for-first-col .tl_set:N = \l_@@_code_for_first_col_tl ,
506   code-for-first-col .value_required:n = true ,
507   code-for-last-col .tl_set:N = \l_@@_code_for_last_col_tl ,
508   code-for-last-col .value_required:n = true ,
509   code-for-first-row .tl_set:N = \l_@@_code_for_first_row_tl ,
510   code-for-first-row .value_required:n = true ,
511   code-for-last-row .tl_set:N = \l_@@_code_for_last_row_tl ,
512   code-for-last-row .value_required:n = true ,
513   hlines .bool_set:N = \l_@@_hlines_bool ,

```

```

514 vlines .bool_set:N = \l_@@_vlines_bool ,
515 hvlines .code:n =
516 {
517     \bool_set_true:N \l_@@_vlines_bool
518     \bool_set_true:N \l_@@_hlines_bool
519 } ,
520 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.

```

521 renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
522 renew-dots .value_forbidden:n = true ,
523 nullify-dots .bool_set:N = \l_@@_nullify_dots_bool ,
524 create-medium-nodes .bool_set:N = \l_@@_medium_nodes_bool ,
525 create-large-nodes .bool_set:N = \l_@@_large_nodes_bool ,
526 create-extra-nodes .meta:n =
527 { create-medium-nodes , create-large-nodes } ,
528 left-margin .dim_set:N = \l_@@_left_margin_dim ,
529 left-margin .default:n = \arraycolsep ,
530 right-margin .dim_set:N = \l_@@_right_margin_dim ,
531 right-margin .default:n = \arraycolsep ,
532 margin .meta:n = { left-margin = #1 , right-margin = #1 } ,
533 margin .default:n = \arraycolsep ,
534 extra-left-margin .dim_set:N = \l_@@_extra_left_margin_dim ,
535 extra-right-margin .dim_set:N = \l_@@_extra_right_margin_dim ,
536 extra-margin .meta:n =
537 { extra-left-margin = #1 , extra-right-margin = #1 } ,
538 extra-margin .value_required:n = true ,
539 }

```

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

```

540 \keys_define:nn { NiceMatrix / Env }
541 {
542     except-corners .clist_set:N = \l_@@_except_corners_clist ,
543     except-corners .default:n = { NW , SW , NE , SE } ,
544     hvlines-except-corners .code:n =
545     {
546         \clist_set:Nn \l_@@_except_corners_clist { #1 }
547         \bool_set_true:N \l_@@_vlines_bool
548         \bool_set_true:N \l_@@_hlines_bool
549     } ,
550     hvlines-except-corners .default:n = { NW , SW , NE , SE } ,
551     code-before .code:n =
552     {
553         \tl_if_empty:nF { #1 }
554         {
555             \tl_put_right:Nn \l_@@_code_before_tl { #1 }
556             \bool_set_true:N \l_@@_code_before_bool
557         }
558     } ,

```

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

```

559 c .code:n = \tl_set:Nn \l_@@_baseline_str c ,
560 t .code:n = \tl_set:Nn \l_@@_baseline_str t ,
561 b .code:n = \tl_set:Nn \l_@@_baseline_str b ,
562 baseline .tl_set:N = \l_@@_baseline_str ,
563 baseline .value_required:n = true ,
564 columns-width .code:n =
565 \tl_if_eq:nnTF { #1 } { auto }

```

```

566     { \bool_set_true:N \l_@@_auto_columns_width_bool }
567     { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,
568     columns-width .value_required:n = true ,
569     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.

```

570     \legacy_if:nF { measuring@ }
571     {
572         \str_set:Nn \l_tmpa_str { #1 }
573         \seq_if_in:NVTF \g_@@_names_seq \l_tmpa_str
574         { \@@_error:nn { Duplicate~name } { #1 } }
575         { \seq_gput_left:NV \g_@@_names_seq \l_tmpa_str }
576         \str_set_eq:NN \l_@@_name_str \l_tmpa_str
577     } ,
578     name .value_required:n = true ,
579     code-after .tl_gset:N = \g_nicematrix_code_after_tl ,
580     code-after .value_required:n = true ,
581     colortbl-like .code:n =
582         \bool_set_true:N \l_@@_colortbl_like_bool
583         \bool_set_true:N \l_@@_code_before_bool ,
584     colortbl-like .value_forbidden:n = true
585 }

586 \keys_define:nn { NiceMatrix / notes }
587 {
588     para .bool_set:N = \l_@@_notes_para_bool ,
589     para .default:n = true ,
590     code-before .tl_set:N = \l_@@_notes_code_before_tl ,
591     code-before .value_required:n = true ,
592     code-after .tl_set:N = \l_@@_notes_code_after_tl ,
593     code-after .value_required:n = true ,
594     bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
595     bottomrule .default:n = true ,
596     style .code:n = \cs_set:Nn \@@_notes_style:n { #1 } ,
597     style .value_required:n = true ,
598     label-in-tabular .code:n =
599         \cs_set:Nn \@@_notes_label_in_tabular:n { #1 } ,
600     label-in-tabular .value_required:n = true ,
601     label-in-list .code:n =
602         \cs_set:Nn \@@_notes_label_in_list:n { #1 } ,
603     label-in-list .value_required:n = true ,
604     enumitem-keys .code:n =
605     {
606         \bool_if:NTF \c_@@_in_preamble_bool
607         {
608             \AtBeginDocument
609             {
610                 \bool_if:NT \c_@@_enumitem_loaded_bool
611                 { \setlist* [ tabularnotes ] { #1 } }
612             }
613         }
614         {
615             \bool_if:NT \c_@@_enumitem_loaded_bool
616             { \setlist* [ tabularnotes ] { #1 } }
617         }
618     } ,
619     enumitem-keys .value_required:n = true ,
620     enumitem-keys-para .code:n =
621     {
622         \bool_if:NTF \c_@@_in_preamble_bool
623         {
624             \AtBeginDocument
625             {
626                 \bool_if:NT \c_@@_enumitem_loaded_bool

```

```

627         { \setlist* [ tabularnotes* ] { #1 } }
628     }
629 }
630 {
631     \bool_if:NT \c_@@_enumitem_loaded_bool
632     { \setlist* [ tabularnotes* ] { #1 } }
633 }
634 } ,
635 enumitem-keys-para .value_required:n = true ,
636 unknown .code:n = \@@_error:n { Unknown~key~for~notes }
637 }

```

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

```

638 \keys_define:nn { NiceMatrix }
639 {
640     NiceMatrixOptions .inherit:n =
641     { NiceMatrix / Global } ,
642     NiceMatrixOptions / xdots .inherit:n = NiceMatrix / xdots ,
643     NiceMatrixOptions / rules .inherit:n = NiceMatrix / rules ,
644     NiceMatrixOptions / notes .inherit:n = NiceMatrix / notes ,
645     NiceMatrix .inherit:n =
646     {
647         NiceMatrix / Global ,
648         NiceMatrix / Env ,
649     } ,
650     NiceMatrix / xdots .inherit:n = NiceMatrix / xdots ,
651     NiceMatrix / rules .inherit:n = NiceMatrix / rules ,
652     NiceTabular .inherit:n =
653     {
654         NiceMatrix / Global ,
655         NiceMatrix / Env
656     } ,
657     NiceTabular / xdots .inherit:n = NiceMatrix / xdots ,
658     NiceTabular / rules .inherit:n = NiceMatrix / rules ,
659     NiceArray .inherit:n =
660     {
661         NiceMatrix / Global ,
662         NiceMatrix / Env ,
663     } ,
664     NiceArray / xdots .inherit:n = NiceMatrix / xdots ,
665     NiceArray / rules .inherit:n = NiceMatrix / rules ,
666     pNiceArray .inherit:n =
667     {
668         NiceMatrix / Global ,
669         NiceMatrix / Env ,
670     } ,
671     pNiceArray / xdots .inherit:n = NiceMatrix / xdots ,
672     pNiceArray / rules .inherit:n = NiceMatrix / rules ,
673 }

```

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

```

674 \keys_define:nn { NiceMatrix / NiceMatrixOptions }
675 {
676     last-col .code:n = \tl_if_empty:nF { #1 }
677     { \@@_error:n { last-col~non~empty~for~NiceMatrixOptions } }
678     \int_zero:N \l_@@_last_col_int ,
679     small .bool_set:N = \l_@@_small_bool ,
680     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.

```

681   renew-matrix .code:n = \@@_renew_matrix: ,
682   renew-matrix .value_forbidden:n = true ,

```

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

```

683   transparent .meta:n = { renew-dots , renew-matrix } ,
684   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}`.

```

685   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.

```

686   columns-width .code:n =
687     \tl_if_eq:nnTF { #1 } { auto }
688     { \@@_error:n { Option~auto~for~columns~width } }
689     { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,

```

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

```

690   allow-duplicate-names .code:n =
691     \@@_msg_redirect_name:nn { Duplicate-name } { none } ,
692   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`).

```

693   letter-for-dotted-lines .code:n =
694     {
695       \tl_if_single_token:nTF { #1 }
696       { \str_set:Nx \l_@@_letter_for_dotted_lines_str { #1 } }
697       { \@@_error:n { Bad-value~for~letter~for~dotted~lines } }
698     } ,
699   letter-for-dotted-lines .value_required:n = true ,
700   notes .code:n = \keys_set:nn { NiceMatrix / notes } { #1 } ,
701   notes .value_required:n = true ,
702   unknown .code:n = \@@_error:n { Unknown~key~for~NiceMatrixOptions }
703 }
704 \str_new:N \l_@@_letter_for_dotted_lines_str
705 \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.

```

706 \NewDocumentCommand \NiceMatrixOptions { m }
707 { \keys_set:nn { NiceMatrix / NiceMatrixOptions } { #1 } }

```

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

```

708 \keys_define:nn { NiceMatrix / NiceMatrix }
709 {
710   last-col .code:n = \tl_if_empty:nTF {#1}
711     {
712       \bool_set_true:N \l_@@_last_col_without_value_bool
713       \int_set:Nn \l_@@_last_col_int { -1 }
714     }
715     { \int_set:Nn \l_@@_last_col_int { #1 } } ,

```

```

716 l .code:n = \tl_set:Nn \l_@@_type_of_col_tl l ,
717 r .code:n = \tl_set:Nn \l_@@_type_of_col_tl r ,
718 small .bool_set:N = \l_@@_small_bool ,
719 small .value_forbidden:n = true ,
720 unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix }
721 }

```

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

```

722 \keys_define:nn { NiceMatrix / NiceArray }
723 {

```

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.

```

724 small .bool_set:N = \l_@@_small_bool ,
725 small .value_forbidden:n = true ,
726 last-col .code:n = \tl_if_empty:nF { #1 }
727 { \@@_error:n { last-col~non-empty~for~NiceArray } }
728 \int_zero:N \l_@@_last_col_int ,
729 notes / para .bool_set:N = \l_@@_notes_para_bool ,
730 notes / para .default:n = true ,
731 notes / bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
732 notes / bottomrule .default:n = true ,
733 tabularnote .tl_set:N = \l_@@_tabularnote_tl ,
734 tabularnote .value_required:n = true ,
735 unknown .code:n = \@@_error:n { Unknown~option~for~NiceArray }
736 }
737 \keys_define:nn { NiceMatrix / pNiceArray }
738 {
739 first-col .code:n = \int_zero:N \l_@@_first_col_int ,
740 last-col .code:n = \tl_if_empty:nF {#1}
741 { \@@_error:n { last-col~non-empty~for~NiceArray } }
742 \int_zero:N \l_@@_last_col_int ,
743 first-row .code:n = \int_zero:N \l_@@_first_row_int ,
744 small .bool_set:N = \l_@@_small_bool ,
745 small .value_forbidden:n = true ,
746 unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix }
747 }

```

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

```

748 \keys_define:nn { NiceMatrix / NiceTabular }
749 {
750 notes / para .bool_set:N = \l_@@_notes_para_bool ,
751 notes / para .default:n = true ,
752 notes / bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
753 notes / bottomrule .default:n = true ,
754 tabularnote .tl_set:N = \l_@@_tabularnote_tl ,
755 tabularnote .value_required:n = true ,
756 last-col .code:n = \tl_if_empty:nF {#1}
757 { \@@_error:n { last-col~non-empty~for~NiceArray } }
758 \int_zero:N \l_@@_last_col_int ,
759 unknown .code:n = \@@_error:n { Unknown~option~for~NiceTabular }
760 }

```


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}`).

```
761 \cs_new_protected:Npn \@@_Cell:
762 {
```

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

```
763 \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.

```
764 \int_compare:nNnT \c@jCol = 1
765 { \int_compare:nNnT \l_@@_first_col_int = 1 \@@_begin_of_row: }
766 \int_gset:Nn \g_@@_col_total_int { \int_max:nn \g_@@_col_total_int \c@jCol }
```

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

```
767 \hbox_set:Nw \l_@@_cell_box
768 \bool_if:NF \l_@@_NiceTabular_bool
769 {
770 \c_math_toggle_token
771 \bool_if:NT \l_@@_small_bool \scriptstyle
772 }
```

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.

```
773 \int_compare:nNnTF \c@iRow = 0
774 {
775 \int_compare:nNnT \c@jCol > 0
776 {
777 \l_@@_code_for_first_row_tl
778 \xglobal \colorlet { nicematrix-first-row } { . }
779 }
780 }
781 {
782 \int_compare:nNnT \c@iRow = \l_@@_last_row_int
783 {
784 \l_@@_code_for_last_row_tl
785 \xglobal \colorlet { nicematrix-last-row } { . }
786 }
787 }
788 }
```

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.

```
789 \cs_new_protected:Npn \@@_begin_of_row:
790 {
791 \int_gincr:N \c@iRow
792 \dim_gset_eq:NN \g_@@_dp_ante_last_row_dim \g_@@_dp_last_row_dim
793 \dim_gset:Nn \g_@@_dp_last_row_dim { \box_dp:N \@arstrutbox }
794 \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
795 \pgfpicture
796 \pgfrememberpicturepositiononpagetrue
797 \pgfcoordinate
798 { \@@_env: - row - \int_use:N \c@iRow - base }
799 { \pgfpoint \c_zero_dim { 0.5 \arrayrulewidth } }
800 \str_if_empty:NF \l_@@_name_str
```

```

801 {
802   \pgfnodealias
803   { \l_@@_name_str - row - \int_use:N \c@iRow - base }
804   { \@@_env: - row - \int_use:N \c@iRow - base }
805 }
806 \endpgfpicture
807 }

```

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.

```

808 \cs_new_protected:Npn \@@_update_for_first_and_last_row:
809 {
810   \int_compare:nNnTF \c@iRow = 0
811   {
812     \dim_gset:Nn \g_@@_dp_row_zero_dim
813     { \dim_max:nn \g_@@_dp_row_zero_dim { \box_dp:N \l_@@_cell_box } }
814     \dim_gset:Nn \g_@@_ht_row_zero_dim
815     { \dim_max:nn \g_@@_ht_row_zero_dim { \box_ht:N \l_@@_cell_box } }
816   }
817   {
818     \int_compare:nNnT \c@iRow = 1
819     {
820       \dim_gset:Nn \g_@@_ht_row_one_dim
821       { \dim_max:nn \g_@@_ht_row_one_dim { \box_ht:N \l_@@_cell_box } }
822     }
823   }
824 }
825 \cs_new_protected:Npn \@@_rotate_cell_box:
826 {
827   \box_rotate:Nn \l_@@_cell_box { 90 }
828   \int_compare:nNnT \c@iRow = \l_@@_last_row_int
829   {
830     \vbox_set_top:Nn \l_@@_cell_box
831     {
832       \vbox_to_zero:n { }
833       \skip_vertical:n { - \box_ht:N \@arstrutbox + 0.8 ex }
834       \box_use:N \l_@@_cell_box
835     }
836   }
837   \bool_gset_false:N \g_@@_rotate_bool
838 }
839 \cs_new_protected:Npn \@@_adjust_width_box:
840 {
841   \dim_compare:nNnT \g_@@_blocks_width_dim > \c_zero_dim
842   {
843     \box_set_wd:Nn \l_@@_cell_box
844     { \dim_max:nn { \box_wd:N \l_@@_cell_box } \g_@@_blocks_width_dim }
845     \dim_gzero:N \g_@@_blocks_width_dim
846   }
847 }
848 \cs_new_protected:Npn \@@_end_Cell:
849 {
850   \@@_math_toggle_token:
851   \hbox_set_end:
852   \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
853   \@@_adjust_width_box:
854   \box_set_ht:Nn \l_@@_cell_box
855   { \box_ht:N \l_@@_cell_box + \l_@@_cell_space_top_limit_dim }
856   \box_set_dp:Nn \l_@@_cell_box
857   { \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”).

```
858 \dim_gset:Nn \g_@@_max_cell_width_dim
859 { \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”.

```
860 \@@_update_for_first_and_last_row:
```

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

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

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

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

```
861 \bool_if:NTF \g_@@_empty_cell_bool
862 { \box_use_drop_i:N \l_@@_cell_box }
863 {
864   \dim_compare:nNnTF { \box_wd:N \l_@@_cell_box } > \c_zero_dim
865     \@@_node_for_the_cell:
866     { \box_use_drop_ii:N \l_@@_cell_box }
867 }
868 \bool_gset_false:N \g_@@_empty_cell_bool
869 }
```

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

```
870 \cs_new_protected:Npn \@@_node_for_the_cell:
871 {
872   \pgfpicture
873   \pgfsetbaseline \c_zero_dim
874   \pgfrememberpicturepositiononpagetrue
875   \pgfset
876   {
877     inner-sep = \c_zero_dim ,
878     minimum-width = \c_zero_dim
879   }
880   \pgfnode
881   { rectangle }
882   { base }
883   { \box_use_drop_iii:N \l_@@_cell_box }
884   { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
885   { }
886   \str_if_empty:NF \l_@@_name_str
887   {
888     \pgfnodealias
889     { \l_@@_name_str - \int_use:N \c@iRow - \int_use:N \c@jCol }
890     { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
891   }
892   \endpgfpicture
893 }
```

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.

```
894 \cs_new_protected:Npn \@@_instruction_of_type:nnn #1 #2 #3
895 {
```

It's important to use a `\tl_gput_right:cx` and not a `\tl_gput_left:cx` because we want the `\Ddots` lines to be drawn in the order of appearance in the array (for parallelisation).

```
896   \bool_if:NTF { #1 } \tl_gput_left:cx \tl_gput_right:cx
897   { \g_@@_ #2 _ lines _ tl }
898   {
899     \use:c { @@ _ draw _ #2 : nnn }
900     { \int_use:N \c@iRow }
901     { \int_use:N \c@jCol }
902     { \exp_not:n { #3 } }
903   }
904 }
```

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`.

```
905 \cs_new_protected:Npn \@@_revtex_array:
906 {
907   \cs_set_eq:NN \@acoll \@arrayacol
908   \cs_set_eq:NN \@acolr \@arrayacol
909   \cs_set_eq:NN \@acol \@arrayacol
910   \cs_set_nopar:Npn \@halignto { }
911   \@array@array
912 }

913 \cs_new_protected:Npn \@@_array:
914 {
915   \bool_if:NTF \c_@@_revtex_bool
916   \@_revtex_array:
917   {
918     \bool_if:NTF \l_@@_NiceTabular_bool
919     { \dim_set_eq:NN \col@sep \tabcolsep }
920     { \dim_set_eq:NN \col@sep \arraycolsep }
921     \dim_compare:nNnTF \l_@@_tabular_width_dim = \c_zero_dim
922     { \cs_set_nopar:Npn \@halignto { } }
923     { \cs_set_nopar:Npx \@halignto { to \dim_use:N \l_@@_tabular_width_dim } }
```

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

```
924   \@tabarray
925 }
```

`\l_@@_baseline_str` may have the value `t`, `c` or `b`. However, if the value is `b`, we compose the `\array` (of `array`) with the option `t` and the right translation will be done further.

```

926   [ \str_if_eq:VnTF \l_@@_baseline_str c c t ]
927   }

```

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.

```

928 \cs_set_eq:NN \@@_old_ialign: \ialign

```

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

```

929 \cs_new_protected:Npn \@@_create_row_node:
930 {

```

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

```

931   \hbox
932   {
933     \bool_if:NT \l_@@_code_before_bool
934     {
935       \vtop
936       {
937         \skip_vertical:N 0.5\arrayrulewidth
938         \pgfsys@markposition { \@@_env: - row - \@@_succ:n \c@iRow }
939         \skip_vertical:N -0.5\arrayrulewidth
940       }
941     }
942     \pgfpicture
943     \pgfrememberpicturepositiononpagetrue
944     \pgfcoordinate { \@@_env: - row - \@@_succ:n \c@iRow }
945     { \pgfpoint \c_zero_dim { - 0.5 \arrayrulewidth } }
946     \str_if_empty:NF \l_@@_name_str
947     {
948       \pgfnodealias
949       { \l_@@_name_str - row - \int_use:N \c@iRow }
950       { \@@_env: - row - \int_use:N \c@iRow }
951     }
952     \endpgfpicture
953   }
954 }

```

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

```

955 \cs_new:Npn \@@_everycr: { \noalign { \@@_everycr_i: } }
956 \cs_new_protected:Npn \@@_everycr_i:
957 {
958   \int_gzero:N \c@jCol
959   \bool_if:NF \g_@@_row_of_col_done_bool
960   {
961     \@@_create_row_node:

```

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

```

962   \bool_if:NT \l_@@_hlines_bool
963   {

```

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.

```

964     \int_compare:nNnT \c@iRow > { -1 }
965     {
966       \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@`.

```

967   { \hrule height \arrayrulewidth width \c_zero_dim }

```

```

968     }
969   }
970 }
971 }

```

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`).

```

972 \cs_set_protected:Npn \@@_newcolumntype #1
973 {
974   \cs_set:cpn { NC @ find @ #1 } ##1 #1 { \NC@ { ##1 } }
975   \peek_meaning:NTF [
976     { \newcol@ #1 }
977     { \newcol@ #1 [ 0 ] }
978   }

```

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

```

979 \cs_set_protected:Npn \@@_renew_dots:
980 {
981   \cs_set_eq:NN \ldots \@@_Ldots
982   \cs_set_eq:NN \cdots \@@_Cdots
983   \cs_set_eq:NN \vdots \@@_Vdots
984   \cs_set_eq:NN \ddots \@@_Ddots
985   \cs_set_eq:NN \iddots \@@_Iddots
986   \cs_set_eq:NN \dots \@@_Ldots
987   \cs_set_eq:NN \hdotsfor \@@_Hdotsfor:
988 }

```

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

```

989 \cs_new_protected:Npn \@@_colortbl_like:
990 {
991   \cs_set_eq:NN \cellcolor \@@_cellcolor_tabular
992   \cs_set_eq:NN \rowcolor \@@_rowcolor_tabular
993   \cs_set_eq:NN \columncolor \@@_columncolor_preamble
994 }

```

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

```

995 \cs_new_protected:Npn \@@_pre_array:
996 {

```

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 ³⁵.

```

997   \bool_if:NT \c_@@_booktabs_loaded_bool
998     { \tl_put_left:Nn \@BTnormal \@@_create_row_node: }
999   \box_clear_new:N \l_@@_cell_box
1000   \cs_if_exist:NT \theiRow
1001     { \int_set_eq:NN \l_@@_old_iRow_int \c@iRow }
1002   \int_gzero_new:N \c@iRow
1003   \cs_if_exist:NT \thejCol
1004     { \int_set_eq:NN \l_@@_old_jCol_int \c@jCol }
1005   \int_gzero_new:N \c@jCol
1006   \normalbaselines

```

³⁵cf. `\nicematrix@redefine@check@rerun`

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}`).

```

1007   \bool_if:NT \l_@@_small_bool
1008   {
1009       \cs_set_nopar:Npn \arraystretch { 0.47 }
1010       \dim_set:Nn \arraycolsep { 1.45 pt }
1011   }

```

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`.

```

1012   \cs_set_nopar:Npn \ialign
1013   {
1014       \bool_if:NTF \c_@@_colortbl_loaded_bool
1015       {
1016           \CT@everycr
1017           {
1018               \noalign { \cs_gset_eq:NN \CT@row@color \prg_do_nothing: }
1019               \@@_everycr:
1020           }
1021       }
1022       { \everycr { \@@_everycr: } }
1023       \tabskip = \c_zero_skip

```

The box `\@arstrutbox` is a box constructed in the beginning of the environment `{array}`. The construction of that box takes into account the current values of `\arraystretch`³⁶ 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`.

```

1024       \dim_gzero_new:N \g_@@_dp_row_zero_dim
1025       \dim_gset:Nn \g_@@_dp_row_zero_dim { \box_dp:N \@arstrutbox }
1026       \dim_gzero_new:N \g_@@_ht_row_zero_dim
1027       \dim_gset:Nn \g_@@_ht_row_zero_dim { \box_ht:N \@arstrutbox }
1028       \dim_gzero_new:N \g_@@_ht_row_one_dim
1029       \dim_gset:Nn \g_@@_ht_row_one_dim { \box_ht:N \@arstrutbox }
1030       \dim_gzero_new:N \g_@@_dp_ante_last_row_dim
1031       \dim_gzero_new:N \g_@@_ht_last_row_dim
1032       \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
1033       \dim_gzero_new:N \g_@@_dp_last_row_dim
1034       \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`.

```

1035       \cs_set_eq:NN \ialign \@@_old_ialign:
1036       \halign
1037   }

```

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).

```

1038   \cs_set_eq:NN \@@_old_ldots \ldots
1039   \cs_set_eq:NN \@@_old_cdots \cdots
1040   \cs_set_eq:NN \@@_old_vdots \vdots
1041   \cs_set_eq:NN \@@_old_ddots \ddots
1042   \cs_set_eq:NN \@@_old_iddots \iddots
1043   \bool_if:NTF \l_@@_standard_cline_bool
1044   { \cs_set_eq:NN \cline \@@_standard_cline }
1045   { \cs_set_eq:NN \cline \@@_cline }

```

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

```

1046 \cs_set_eq:NN \Ldots \@@_Ldots
1047 \cs_set_eq:NN \Cdots \@@_Cdots
1048 \cs_set_eq:NN \Vdots \@@_Vdots
1049 \cs_set_eq:NN \Ddots \@@_Ddots
1050 \cs_set_eq:NN \Iddots \@@_Iddots
1051 \cs_set_eq:NN \hdottedline \@@_hdottedline:
1052 \cs_set_eq:NN \Hline \@@_Hline:
1053 \cs_set_eq:NN \Hspace \@@_Hspace:
1054 \cs_set_eq:NN \Hdotsfor \@@_Hdotsfor:
1055 \cs_set_eq:NN \Vdotsfor \@@_Vdotsfor:
1056 \cs_set_eq:NN \multicolumn \@@_multicolumn:nnn
1057 \cs_set_eq:NN \Block \@@_Block:
1058 \cs_set_eq:NN \rotate \@@_rotate:
1059 \cs_set_eq:NN \OnlyMainNiceMatrix \@@_OnlyMainNiceMatrix:n
1060 \cs_set_eq:NN \dotfill \@@_old_dotfill:
1061 \cs_set_eq:NN \CodeAfter \@@_CodeAfter:n
1062 \cs_set_eq:NN \diagbox \@@_diagbox:nn
1063 \bool_if:NT \l_@@_colortbl_like_bool \@@_colortbl_like:
1064 \bool_if:NT \l_@@_renew_dots_bool \@@_renew_dots:

```

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

```

1065 \seq_gclear_new:N \g_@@_multicolumn_cells_seq
1066 \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).

```

1067 \int_gset:Nn \c@iRow { \l_@@_first_row_int - 1 }

```

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

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

```

1068 \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.

```

1069 \int_gzero_new:N \g_@@_col_total_int
1070 \cs_set_eq:NN \@ifnextchar \new@ifnextchar
1071 \@@_renew_NC@rewrite@S:
1072 \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.

```

1073 \tl_gclear_new:N \g_@@_Cdots_lines_tl
1074 \tl_gclear_new:N \g_@@_Ldots_lines_tl
1075 \tl_gclear_new:N \g_@@_Vdots_lines_tl
1076 \tl_gclear_new:N \g_@@_Ddots_lines_tl
1077 \tl_gclear_new:N \g_@@_Iddots_lines_tl
1078 \tl_gclear_new:N \g_@@_HVDotsfor_lines_tl

1079 \tl_gclear_new:N \g_nicematrix_code_before_tl
1080 }

```

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

The environment {NiceArrayWithDelims}

```

1081 \NewDocumentEnvironment { NiceArrayWithDelims } { m m O { } m ! O { } }
1082 {
1083   \@@_provide_pgfsyspdfmark:
1084   \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.

```

1085   \bgroup
1086   \tl_set:Nn \l_@@_left_delim_tl { #1 }
1087   \tl_set:Nn \l_@@_right_delim_tl { #2 }
1088   \int_gzero:N \g_@@_block_box_int
1089   \dim_zero:N \g_@@_width_last_col_dim
1090   \dim_zero:N \g_@@_width_first_col_dim
1091   \bool_gset_false:N \g_@@_row_of_col_done_bool
1092   \str_if_empty:NT \g_@@_name_env_str
1093   { \str_gset:Nn \g_@@_name_env_str { NiceArrayWithDelims } }
1094   \@@_adapt_S_column:
1095   \bool_if:NTF \l_@@_NiceTabular_bool
1096     \mode_leave_vertical:
1097     \@@_test_if_math_mode:
1098   \bool_if:NT \l_@@_in_env_bool { \@@_fatal:n { Yet~in~env } }
1099   \bool_set_true:N \l_@@_in_env_bool

```

The command `\CT@arc@` contains the instruction of color for the rules of the array³⁷. 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.

```

1100   \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.

```

1101   \cs_if_exist:NT \tikz@library@external@loaded
1102   {
1103     \tikzexternaldisable
1104     \cs_if_exist:NT \ifstandalone
1105     { \tikzset { external / optimize = false } }
1106   }

```

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

```

1107   \int_gincr:N \g_@@_env_int
1108   \bool_if:NF \l_@@_block_auto_columns_width_bool
1109   { \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.

```

1110   \seq_gclear:N \g_@@_blocks_seq
1111   \seq_gclear:N \g_@@_pos_of_blocks_seq
1112   \seq_gclear:N \g_@@_pos_of_xdots_seq

```

In fact, the sequence `\g_@@_pos_of_blocks_seq` will also contain the positions of the cells with a `\diagbox`.

```

1113   \tl_if_exist:cT { g_@@_code_before_ \int_use:N \g_@@_env_int _ tl }
1114   {
1115     \bool_set_true:N \l_@@_code_before_bool
1116     \exp_args:NNv \tl_put_right:Nn \l_@@_code_before_tl
1117     { g_@@_code_before_ \int_use:N \g_@@_env_int _ tl }
1118   }

```

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`.

³⁷e.g. `\color[rgb]{0.5,0.5,0}`

```

1119 \bool_if:NTF \l_@@_NiceArray_bool
1120 { \keys_set:nn { NiceMatrix / NiceArray } }
1121 { \keys_set:nn { NiceMatrix / pNiceArray } }
1122 { #3 , #5 }

1123 \tl_if_empty:NF \l_@@_rules_color_tl
1124 { \exp_after:wN \@@_set_CT@arc@: \l_@@_rules_color_tl \q_stop }

```

If the key `code-before` is used, we have to create the `col` nodes and the `row` nodes before the creation of the array. First, we have to test whether the size of the array has been written in the `aux` file in a previous run. In this case, a command `\@@_size_nb_of_env:` has been created.

```

1125 \bool_if:NT \l_@@_code_before_bool
1126 {
1127   \seq_if_exist:cT { @@_size_ \int_use:N \g_@@_env_int _ seq }
1128   {

```

First, we give values to the LaTeX counters `iRow` and `jCol`. We remind that, in the `code-before` (and in the `code-after`) they represent the numbers of rows and columns of the array (without the potential last row and last column).

```

1129 \int_zero_new:N \c@iRow
1130 \int_set:Nn \c@iRow
1131 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 2 }
1132 \int_zero_new:N \c@jCol
1133 \int_set:Nn \c@jCol
1134 { \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.

```

1135 \int_compare:nNnF \l_@@_last_row_int = { -2 }
1136 { \int_decr:N \c@iRow }
1137 \int_compare:nNnF \l_@@_last_col_int = { -2 }
1138 { \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.

```

1139 \pgfsys@markposition { \@@_env: - position }
1140 \pgfsys@getposition { \@@_env: - position } \@@_picture_position:
1141 \pgfpicture

```

First, the creation of the `row` nodes.

```

1142 \int_step_inline:nnn
1143 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 1 }
1144 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 2 + 1 }
1145 {
1146   \pgfsys@getposition { \@@_env: - row - ##1 } \@@_node_position:
1147   \pgfcoordinate { \@@_env: - row - ##1 }
1148   { \pgfpointdiff \@@_picture_position: \@@_node_position: }
1149 }

```

Now, the creation of the `col` nodes.

```

1150 \int_step_inline:nnn
1151 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 3 }
1152 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 4 + 1 }
1153 {
1154   \pgfsys@getposition { \@@_env: - col - ##1 } \@@_node_position:
1155   \pgfcoordinate { \@@_env: - col - ##1 }
1156   { \pgfpointdiff \@@_picture_position: \@@_node_position: }
1157 }
1158 \endpgfpicture
1159 \group_begin:
1160 \bool_if:NT \c_@@_tikz_loaded_bool
1161 {
1162   \tikzset
1163   {

```

```

1164         every-picture / .style =
1165         { overlay , name-prefix = \@@_env: - }
1166     }
1167 }
1168 \cs_set_eq:NN \cellcolor \@@_cellcolor
1169 \cs_set_eq:NN \rectanglecolor \@@_rectanglecolor
1170 \cs_set_eq:NN \rowcolor \@@_rowcolor
1171 \cs_set_eq:NN \rowcolors \@@_rowcolors
1172 \cs_set_eq:NN \columncolor \@@_columncolor
1173 \cs_set_eq:NN \chessboardcolors \@@_chessboardcolors

```

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

```

1174     \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
1175     \l_@@_code_before_tl
1176     \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
1177 \group_end:
1178 }
1179 }

```

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).

```

1180 \int_compare:nNnT \l_@@_last_row_int > { -2 }
1181 {
1182     \tl_put_right:Nn \@@_update_for_first_and_last_row:
1183     {
1184         \dim_gset:Nn \g_@@_ht_last_row_dim
1185         { \dim_max:nn \g_@@_ht_last_row_dim { \box_ht:N \l_@@_cell_box } }
1186         \dim_gset:Nn \g_@@_dp_last_row_dim
1187         { \dim_max:nn \g_@@_dp_last_row_dim { \box_dp:N \l_@@_cell_box } }
1188     }
1189 }
1190 \int_compare:nNnT \l_@@_last_row_int = { -1 }
1191 {
1192     \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.

```

1193 \str_if_empty:NTF \l_@@_name_str
1194 {
1195     \cs_if_exist:cT { @@_last_row_ \int_use:N \g_@@_env_int }
1196     {
1197         \int_set:Nn \l_@@_last_row_int
1198         { \use:c { @@_last_row_ \int_use:N \g_@@_env_int } }
1199     }
1200 }
1201 {
1202     \cs_if_exist:cT { @@_last_row_ \l_@@_name_str }
1203     {
1204         \int_set:Nn \l_@@_last_row_int
1205         { \use:c { @@_last_row_ \l_@@_name_str } }
1206     }
1207 }
1208 }

```

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).

```

1209 \int_compare:nNnT \l_@@_last_col_int = { -1 }
1210 {
1211     \str_if_empty:NTF \l_@@_name_str
1212     {
1213         \cs_if_exist:cT { @@_last_col_ \int_use:N \g_@@_env_int }
1214         {
1215             \int_set:Nn \l_@@_last_col_int
1216             { \use:c { @@_last_col_ \int_use:N \g_@@_env_int } }

```

```

1217     }
1218   }
1219   {
1220     \cs_if_exist:cT { @@_last_col_ \l_@@_name_str }
1221     {
1222       \int_set:Nn \l_@@_last_col_int
1223       { \use:c { @@_last_col_ \l_@@_name_str } }
1224     }
1225   }
1226 }

```

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

```

1227 \@@_pre_array:

```

We compute the width of the two delimiters.

```

1228 \dim_zero_new:N \l_@@_left_delim_dim
1229 \dim_zero_new:N \l_@@_right_delim_dim
1230 \bool_if:NTF \l_@@_NiceArray_bool
1231 {
1232   \dim_gset:Nn \l_@@_left_delim_dim { 2 \arraycolsep }
1233   \dim_gset:Nn \l_@@_right_delim_dim { 2 \arraycolsep }
1234 }
1235 {

```

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

```

1236 \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 #1 $ }
1237 \dim_set:Nn \l_@@_left_delim_dim { \box_wd:N \l_tmpa_box }
1238 \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 #2 $ }
1239 \dim_set:Nn \l_@@_right_delim_dim { \box_wd:N \l_tmpa_box }
1240 }

```

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

```

1241 \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).

```

1242 \bool_if:NT \c_@@_define_L_C_R_bool \@@_define_L_C_R:

```

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

```

1243 \@@_construct_preamble:n { #4 }

```

Now, the preamble is constructed in `\g_@@_preamble_tl`

Here is the beginning of the box which will contain the array. The `\hbox_set_end:` corresponding to this `\hbox_set:Nw` will be in the second part of the environment (and the closing `\c_math_toggle_token` also).

```

1244 \hbox_set:Nw \l_@@_the_array_box
1245 \skip_horizontal:N \l_@@_left_margin_dim
1246 \skip_horizontal:N \l_@@_extra_left_margin_dim
1247 \c_math_toggle_token
1248 \bool_if:NTF \l_@@_light_syntax_bool
1249 { \use:c { @@-light-syntax } }
1250 { \use:c { @@-normal-syntax } }
1251 }
1252 {
1253   \bool_if:NTF \l_@@_light_syntax_bool
1254   { \use:c { end @@-light-syntax } }
1255   { \use:c { end @@-normal-syntax } }
1256   \c_math_toggle_token
1257   \skip_horizontal:N \l_@@_right_margin_dim

```

```

1258 \skip_horizontal:N \l_@@_extra_right_margin_dim
1259 \hbox_set_end:

```

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

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

```

1260 \int_compare:nNnT \l_@@_last_row_int > { -2 }
1261 {
1262   \bool_if:NF \l_@@_last_row_without_value_bool
1263   {
1264     \int_compare:nNnF \l_@@_last_row_int = \c@iRow
1265     {
1266       \@@_error:n { Wrong~last~row }
1267       \int_gset_eq:NN \l_@@_last_row_int \c@iRow
1268     }
1269   }
1270 }

```

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”.³⁸

```

1271 \int_gset_eq:NN \c@jCol \g_@@_col_total_int
1272 \bool_if:nTF \g_@@_last_col_found_bool
1273 { \int_gdecr:N \c@jCol }
1274 {
1275   \int_compare:nNnT \l_@@_last_col_int > { -1 }
1276   { \@@_error:n { last~col~not~used } }
1277 }
1278 \bool_if:NF \l_@@_Matrix_bool
1279 {
1280   \int_compare:nNnT \c@jCol < \g_@@_static_num_of_col_int
1281   { \@@_error:n { columns~not~used } }
1282 }

```

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

```

1283 \int_gset_eq:NN \g_@@_row_total_int \c@iRow
1284 \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. 92).

```

1285 \int_compare:nNnT \l_@@_first_col_int = 0
1286 {
1287   \skip_horizontal:N \col@sep
1288   \dim_if_exist:cTF { g_@@_block _ width _ col _ 0 _ dim }
1289   {
1290     \skip_horizontal:n
1291     {
1292       \dim_max:nn
1293       {
1294         \dim_use:c { g_@@_block _ width _ col _ 0 _ dim }
1295         - 2 \col@sep
1296       }
1297       \g_@@_width_first_col_dim
1298     }
1299     \dim_gzero:c { g_@@_block _ width _ col _ 0 _ dim }
1300   }
1301   { \skip_horizontal:N \g_@@_width_first_col_dim }
1302 }

```

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

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

```

1303 \bool_if:NTF \l_@@_NiceArray_bool
1304 {
1305   \str_case:VnF \l_@@_baseline_str
1306   {
1307     b \@@_use_arraybox_with_notes_b:
1308     c \@@_use_arraybox_with_notes_c:
1309   }
1310   \@@_use_arraybox_with_notes:
1311 }

```

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).

```

1312 {
1313   \int_compare:nNnTF \l_@@_first_row_int = 0
1314   {
1315     \dim_set_eq:NN \l_tmpa_dim \g_@@_dp_row_zero_dim
1316     \dim_add:Nn \l_tmpa_dim \g_@@_ht_row_zero_dim
1317   }
1318   { \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”.³⁹

```

1319   \int_compare:nNnTF \l_@@_last_row_int > { -2 }
1320   {
1321     \dim_set_eq:NN \l_tmpb_dim \g_@@_ht_last_row_dim
1322     \dim_add:Nn \l_tmpb_dim \g_@@_dp_last_row_dim
1323   }
1324   { \dim_zero:N \l_tmpb_dim }
1325   \hbox_set:Nn \l_tmpa_box
1326   {
1327     \c_math_toggle_token
1328     \left #1
1329     \vcenter
1330     {

```

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.

```

1331     \skip_vertical:N -\l_tmpa_dim
1332     \skip_vertical:N -\arrayrulewidth
1333     \hbox
1334     {
1335       \bool_if:NTF \l_@@_NiceTabular_bool
1336       { \skip_horizontal:N -\tabcolsep }
1337       { \skip_horizontal:N -\arraycolsep }
1338       \@@_use_arraybox_with_notes_c:
1339       \bool_if:NTF \l_@@_NiceTabular_bool
1340       { \skip_horizontal:N -\tabcolsep }
1341       { \skip_horizontal:N -\arraycolsep }
1342     }

```

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

```

1343     \skip_vertical:N -\l_tmpb_dim
1344     \skip_vertical:N \arrayrulewidth
1345   }
1346   \right #2
1347   \c_math_toggle_token
1348 }

```

Now, the box `\l_tmpa_box` is created with the correct delimiters.

We will put the box in the TeX flow. However, we have a small work to do when the option `max-delimiter-width` is used.

```

1349   \bool_if:NTF \l_@@_max_delimiter_width_bool
1350   { \@@_put_box_in_flow_bis:nn { #1 } { #2 } }

```

³⁹A 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).

```

1351 \@@_put_box_in_flow:
1352 }
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. 93).
1353 \bool_if:NT \g_@@_last_col_found_bool
1354 {
1355   \dim_if_exist:cTF
1356   { \g_@@_block _ width _ col _ \int_use:N \g_@@_col_total_int _ dim }
1357   {
1358     \skip_horizontal:n
1359     {
1360       \dim_max:nn
1361       {
1362         \dim_use:c
1363         {
1364           \g_@@_block _ width _ col _
1365           \int_use:N \g_@@_col_total_int
1366           _ dim
1367         }
1368       }
1369       { \g_@@_width_first_col_dim + \col@sep }
1370     }
1371   }
1372   {
1373     \skip_horizontal:N \g_@@_width_last_col_dim
1374     \skip_horizontal:N \col@sep
1375   }
1376 }
1377 \@@_after_array:
1378 \egroup
1379 \bool_if:NT \c_@@_footnote_bool \endsavenotes
1380 }

```

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

We construct the preamble of the array

The transformation of the preamble is an operation in several steps.

The argument of `\@@_construct_preamble:n` is the preamble as given by the final user to the environment `{NiceTabular}` (or a variant). The preamble will be constructed in `\g_@@_preamble_tl`.

```

1381 \cs_new_protected:Npn \@@_construct_preamble:n #1
1382 {

```

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

Since we use the tools of `array` to do this expansion, we will have a programming which is not in the style of `expl3`.

We redefine the column types `w` and `W`. We use `\@@_newcolumntype` instead of `\newcolumntype` because we don’t want warnings for column types already defined. These redefinitions are in fact *protections* of the letters `w` and `W`. We don’t want these columns type expanded because we will do the patch ourselves after. We want to be able the standard column types `w` and `W` in potential `{tabular}` of `array` in some cells of our array. That’s why we do those redefinitions in a TeX group.

```

1383 \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).
1384 \bool_if:NTF \l_@@_Matrix_bool
1385 { \tl_gset:Nn \g_@@_preamble_tl { #1 } }
1386 {
1387   \@@_newcolumntype w [ 2 ] { \@@_w: { ##1 } { ##2 } }
1388   \@@_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`).

```
1389 \temptokena { #1 }
```

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

```
1390 \tempswatrue
```

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

```
1391 \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`.

```
1392 \int_gzero_new:N \c@jCol
1393 \bool_if:NTF \l_@@_vlines_bool
1394 {
1395   \tl_gset:Nn \g_@@_preamble_tl
1396   { ! { \skip_horizontal:N \arrayrulewidth } }
1397 }
1398 { \tl_gclear:N \g_@@_preamble_tl }
```

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

```
1399 \int_zero:N \l_tmpa_int
```

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

```
1400 \exp_after:wN \@@_patch_preamble:n \the \@temptokena \q_stop
1401 \int_gset_eq:NN \g_@@_static_num_of_col_int \c@jCol
1402 }
```

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

```
1403 \bool_if:NT \l_@@_colortbl_like_bool
1404 {
1405   \regex_replace_all:NnN
1406   \c_@@_columncolor_regex
1407   { \c { @@_columncolor_preamble } }
1408   \g_@@_preamble_tl
1409 }
```

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

```
1410 \int_compare:nNnTF \l_@@_first_col_int = 0
1411 { \tl_gput_left:NV \g_@@_preamble_tl \c_@@_preamble_first_col_tl }
1412 {
1413   \bool_lazy_all:nT
1414   {
1415     \l_@@_NiceArray_bool
1416     { \bool_not_p:n \l_@@_NiceTabular_bool }
1417     { \bool_not_p:n \l_@@_vlines_bool }
1418     { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
1419   }
1420   { \tl_gput_left:Nn \g_@@_preamble_tl { @ { } } }
1421 }
1422 \int_compare:nNnTF \l_@@_last_col_int > { -1 }
1423 { \tl_gput_right:NV \g_@@_preamble_tl \c_@@_preamble_last_col_tl }
1424 {
1425   \bool_lazy_all:nT
1426   {
1427     \l_@@_NiceArray_bool
1428     { \bool_not_p:n \l_@@_NiceTabular_bool }
1429     { \bool_not_p:n \l_@@_vlines_bool }
1430     { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
1431   }
}
```



```

1432     { \tl_gput_right:Nn \g_@@_preamble_tl { @ { } } }
1433   }

```

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`).

```

1434   \dim_compare:nNnT \l_@@_tabular_width_dim = \c_zero_dim
1435   {
1436     \tl_gput_right:Nn
1437       \g_@@_preamble_tl
1438       { > { \@@_error_too_much_cols: } 1 }
1439   }

```

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

```

1440   \group_end:
1441 }

```

```

1442 \cs_new_protected:Npn \@@_patch_preamble:n #1
1443 {
1444   \str_case:nnF { #1 }
1445   {
1446     c { \@@_patch_preamble_i:n #1 }
1447     l { \@@_patch_preamble_i:n #1 }
1448     r { \@@_patch_preamble_i:n #1 }
1449     > { \@@_patch_preamble_ii:nn #1 }
1450     ! { \@@_patch_preamble_ii:nn #1 }
1451     @ { \@@_patch_preamble_ii:nn #1 }
1452     | { \@@_patch_preamble_iii:n #1 }
1453     p { \@@_patch_preamble_iv:nnn t #1 }
1454     m { \@@_patch_preamble_iv:nnn c #1 }
1455     b { \@@_patch_preamble_iv:nnn b #1 }
1456     \@@_w: { \@@_patch_preamble_v:nnnn { } #1 }
1457     \@@_W: { \@@_patch_preamble_v:nnnn { \cs_set_eq:NN \hss \hfil } #1 }
1458     \@@_true_c: { \@@_patch_preamble_vi:n #1 }
1459     \q_stop { }
1460   }
1461   {
1462     \str_if_eq:VnTF \l_@@_letter_for_dotted_lines_str { #1 }
1463     { \@@_patch_preamble_vii:n #1 }
1464     { \@@_fatal:nn { unknown~column~type } { #1 } }
1465   }
1466 }

```

For `c`, `l` and `r`

```

1467 \cs_new_protected:Npn \@@_patch_preamble_i:n #1
1468 {
1469   \tl_gput_right:Nn \g_@@_preamble_tl
1470   {
1471     > { \@@_Cell: \tl_set:Nn \l_@@_cell_type_tl { #1 } }
1472     #1
1473     < \@@_end_Cell:
1474   }

```

We increment the counter of columns.

```

1475   \int_gincr:N \c@jCol
1476   \@@_patch_preamble_viii:n
1477 }

```

For `>`, `!` and `@`

```

1478 \cs_new_protected:Npn \@@_patch_preamble_ii:nn #1 #2
1479 {
1480   \tl_gput_right:Nn \g_@@_preamble_tl { #1 { #2 } }

```

```

1481 \@@_patch_preamble:n
1482 }

```

For l

```

1483 \cs_new_protected:Npn \@@_patch_preamble_iii:n #1
1484 {

```

\l_tmpa_int is the number of successive occurrences of l

```

1485 \int_incr:N \l_tmpa_int
1486 \@@_patch_preamble_iii_i:n
1487 }

```

```

1488 \cs_new_protected:Npn \@@_patch_preamble_iii_i:n #1
1489 {

```

```

1490 \str_if_eq:nnTF { #1 } |
1491 { \@@_patch_preamble_iii:n | }
1492 {
1493 \tl_gput_right:Nx \g_@@_preamble_tl
1494 {
1495 \exp_not:N !
1496 {
1497 \skip_horizontal:n
1498 {
1499 \dim_eval:n
1500 {
1501 \arrayrulewidth * \l_tmpa_int
1502 + \doublerulesep * ( \l_tmpa_int - 1)
1503 }
1504 }
1505 }
1506 }
1507 \tl_gput_right:Nx \g_@@_internal_code_after_tl
1508 { \@@_vline:nn { \@@_succ:n \c@jCol } { \int_use:N \l_tmpa_int } }
1509 \int_zero:N \l_tmpa_int
1510 \@@_patch_preamble:n #1
1511 }
1512 }

```

For p, m and b

```

1513 \cs_new_protected:Npn \@@_patch_preamble_iv:nnn #1 #2 #3
1514 {
1515 \tl_gput_right:Nn \g_@@_preamble_tl
1516 {
1517 > {
1518 \@@_Cell:
1519 \begin { minipage } [ #1 ] { #3 }
1520 \mode_leave_vertical:
1521 \box_use:N \@arstrutbox
1522 }
1523 c
1524 < { \box_use:N \@arstrutbox \end { minipage } \@@_end_Cell: }
1525 }

```

We increment the counter of columns.

```

1526 \int_gincr:N \c@jCol
1527 \@@_patch_preamble_viii:n
1528 }

```

For w and W

```

1529 \cs_new_protected:Npn \@@_patch_preamble_v:nnnn #1 #2 #3 #4
1530 {
1531 \tl_gput_right:Nn \g_@@_preamble_tl
1532 {
1533 > {

```

```

1534         \hbox_set:Nw \l_@@_cell_box
1535         \@@_Cell:
1536         \tl_set:Nn \l_@@_cell_type_tl { #1 }
1537     }
1538     c
1539     < {
1540         \@@_end_Cell:
1541         #1
1542         \hbox_set_end:
1543         \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
1544         \@@_adjust_width_box:
1545         \makebox [ #4 ] [ #3 ] { \box_use_drop_iv:N \l_@@_cell_box }
1546     }
1547 }

```

We increment the counter of columns.

```

1548     \int_gincr:N \c@jCol
1549     \@@_patch_preamble_viii:n
1550 }

```

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

```

1551 \cs_new_protected:Npn \@@_patch_preamble_vi:n #1
1552 {
1553     \tl_gput_right:Nn \g_@@_preamble_tl { c }

```

We increment the counter of columns.

```

1554     \int_gincr:N \c@jCol
1555     \@@_patch_preamble_viii:n
1556 }
1557 \cs_new_protected:Npn \@@_patch_preamble_vii:n #1
1558 {
1559     \tl_gput_right:Nn \g_@@_preamble_tl
1560     { ! { \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`.

```

1561     \tl_gput_right:Nx \g_@@_internal_code_after_tl
1562     { \@@_vdottedline:n { \int_use:N \c@jCol } }
1563     \@@_patch_preamble:n
1564 }

```

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.

```

1565 \cs_new_protected:Npn \@@_patch_preamble_viii:n #1
1566 {
1567     \str_if_eq:nnTF { #1 } { < }
1568     \@@_patch_preamble_ix:n
1569     {
1570         \bool_if:NT \l_@@_vlines_bool
1571         {
1572             \tl_gput_right:Nn \g_@@_preamble_tl
1573             { ! { \skip_horizontal:N \arrayrulewidth } }
1574         }
1575         \@@_patch_preamble:n { #1 }
1576     }
1577 }
1578 \cs_new_protected:Npn \@@_patch_preamble_ix:n #1
1579 {
1580     \tl_gput_right:Nn \g_@@_preamble_tl { < { #1 } }
1581     \@@_patch_preamble_viii:n
1582 }

```

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`).

```

1583 \cs_new_protected:Npn \@@_put_box_in_flow:
1584 {
1585   \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }
1586   \box_set_dp:Nn \l_tmpa_box { \box_dp:N \l_tmpa_box + \l_tmpb_dim }
1587   \str_if_eq:VnTF \l_@@_baseline_str { c }
1588     { \box_use_drop_v:N \l_tmpa_box }
1589     \@@_put_box_in_flow_i:
1590 }

```

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

```

1591 \cs_new_protected:Npn \@@_put_box_in_flow_i:
1592 {
1593   \pgfpicture
1594     \@@_qpoint:n { row - 1 }
1595     \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1596     \@@_qpoint:n { row - \@@_succ:n \c@iRow }
1597     \dim_gadd:Nn \g_tmpa_dim \pgf@y
1598     \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).

```

1599   \str_if_in:NnTF \l_@@_baseline_str { line- }
1600   {
1601     \int_set:Nn \l_tmpa_int
1602     {
1603       \str_range:Nnn
1604         \l_@@_baseline_str
1605         6
1606         { \str_count:N \l_@@_baseline_str }
1607     }
1608     \@@_qpoint:n { row - \int_use:N \l_tmpa_int }
1609   }
1610   {
1611     \str_case:VnF \l_@@_baseline_str
1612     {
1613       { t } { \int_set:Nn \l_tmpa_int 1 }
1614       { b } { \int_set_eq:NN \l_tmpa_int \c@iRow }
1615     }
1616     { \int_set:Nn \l_tmpa_int \l_@@_baseline_str }
1617     \bool_lazy_or:nnT
1618     { \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int }
1619     { \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int }
1620     {
1621       \@@_error:n { bad~value~for~baseline }
1622       \int_set:Nn \l_tmpa_int 1
1623     }
1624     \@@_qpoint:n { row - \int_use:N \l_tmpa_int - base }

```

We take into account the position of the mathematical axis.

```

1625     \dim_gsub:Nn \g_tmpa_dim { \fontdimen22 \textfont2 }
1626   }
1627   \dim_gsub:Nn \g_tmpa_dim \pgf@y

```

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

```

1628   \endpgfpicture
1629   \box_move_up:nn \g_tmpa_dim { \box_use_drop_v:N \l_tmpa_box }
1630   \box_use_drop_vii:N \l_tmpa_box
1631 }

```

```

1632 \cs_new_protected:Npn \@@_use_arraybox_with_notes_c:
1633 {
1634   \bool_lazy_and:nnTF
1635     { \int_compare_p:nNn \c@tabularnote = 0 }
1636     { \tl_if_empty_p:V \l_@@_tabularnote_tl }
1637     { \box_use_drop_viii:N \l_@@_the_array_box }
1638     {
1639       \begin { minipage } { \box_wd:N \l_@@_the_array_box }
1640       \box_use_drop_ix:N \l_@@_the_array_box
1641       \skip_vertical:N 0.65ex

```

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

```

1642   \group_begin:
1643   \l_@@_notes_code_before_tl
1644   \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.

```

1645   \int_compare:nNnF \c@tabularnote = 0
1646   {
1647     \bool_if:NTF \l_@@_notes_para_bool
1648     {
1649       \begin { tabularnotes* }
1650       \seq_map_inline:Nn \g_@@_tabularnotes_seq { \item ##1 } \strut
1651       \end { tabularnotes* }

```

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

```

1652       \par
1653     }
1654   {
1655     \tabularnotes
1656     \seq_map_inline:Nn \g_@@_tabularnotes_seq { \item ##1 } \strut
1657     \endtabularnotes
1658   }
1659 }
1660 \unskip
1661 \group_end:
1662 \bool_if:NT \l_@@_notes_bottomrule_bool
1663 {
1664   \bool_if:NTF \c_@@_booktabs_loaded_bool
1665   {

```

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

```

1666     \skip_vertical:N \aboverulesep
\CT@arc@ is the specification of color defined by colortbl but you use it even if colortbl is not loaded.
1667     { \CT@arc@ \hrule height \heavyrulewidth }
1668   }
1669   { \@@_error:n { bottomule~without~booktabs } }
1670 }
1671 \l_@@_notes_code_after_tl
1672 \end { minipage }
1673 \seq_gclear:N \g_@@_tabularnotes_seq
1674 \int_gzero:N \c@tabularnote
1675 }
1676 }

```

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`.

```

1677 \cs_new_protected:Npn \@@_use_arraybox_with_notes_b:
1678 {
1679   \pgfpicture
1680   \@@_qpoint:n { row - 1 }

```

```

1681 \dim_gset_eq:Nn \g_tmpa_dim \pgf@y
1682 \@@_qpoint:n { row - \int_use:N \c@iRow - base }
1683 \dim_gsub:Nn \g_tmpa_dim \pgf@y
1684 \endpgfpicture
1685 \dim_gadd:Nn \g_tmpa_dim \arrayrulewidth
1686 \int_compare:nNnT \l_@@_first_row_int = 0
1687 {
1688   \dim_gadd:Nn \g_tmpa_dim \g_@@_ht_row_zero_dim
1689   \dim_gadd:Nn \g_tmpa_dim \g_@@_dp_row_zero_dim
1690 }
1691 \box_move_up:nn \g_tmpa_dim { \@@_use_arraybox_with_notes_c: }
1692 }

```

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

```

1693 \cs_new_protected:Npn \@@_use_arraybox_with_notes:
1694 {

```

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

```

1695 \str_if_eq:VnT \l_@@_baseline_str { t }
1696 { \tl_set:Nn \l_@@_baseline_str { 1 } }

```

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

```

1697 \pgfpicture
1698 \@@_qpoint:n { row - 1 }
1699 \dim_gset_eq:Nn \g_tmpa_dim \pgf@y
1700 \str_if_in:NnTF \l_@@_baseline_str { line- }
1701 {
1702   \int_set:Nn \l_tmpa_int
1703   {
1704     \str_range:Nnn
1705       \l_@@_baseline_str
1706       6
1707     { \str_count:N \l_@@_baseline_str }
1708   }
1709   \@@_qpoint:n { row - \int_use:N \l_tmpa_int }
1710 }
1711 {
1712   \int_set:Nn \l_tmpa_int \l_@@_baseline_str
1713   \bool_lazy_or:nnT
1714     { \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int }
1715     { \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int }
1716   {
1717     \@@_error:n { bad-value-for-baseline }
1718     \int_set:Nn \l_tmpa_int 1
1719   }
1720   \@@_qpoint:n { row - \int_use:N \l_tmpa_int - base }
1721 }
1722 \dim_gsub:Nn \g_tmpa_dim \pgf@y
1723 \endpgfpicture
1724 \dim_gadd:Nn \g_tmpa_dim \arrayrulewidth
1725 \int_compare:nNnT \l_@@_first_row_int = 0
1726 {
1727   \dim_gadd:Nn \g_tmpa_dim \g_@@_ht_row_zero_dim
1728   \dim_gadd:Nn \g_tmpa_dim \g_@@_dp_row_zero_dim
1729 }
1730 \box_move_up:nn \g_tmpa_dim { \@@_use_arraybox_with_notes_c: }
1731 }

```

The command `\@@_put_box_in_flow_bis:` is used when the option `max-delimiter-width` is used because, in this case, we have to adjust the widths of the delimiters. The arguments `#1` and `#2` are the delimiters specified by the user.

```

1732 \cs_new_protected:Npn \@@_put_box_in_flow_bis:nn #1 #2
1733 {

```

We will compute the real width of both delimiters used.

```

1734 \dim_zero_new:N \l_@@_real_left_delim_dim
1735 \dim_zero_new:N \l_@@_real_right_delim_dim
1736 \hbox_set:Nn \l_tmpb_box
1737 {
1738   \c_math_toggle_token
1739   \left #1
1740   \vcenter
1741   {
1742     \vbox_to_ht:nn
1743     { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1744     { }
1745   }
1746   \right .
1747   \c_math_toggle_token
1748 }
1749 \dim_set:Nn \l_@@_real_left_delim_dim
1750 { \box_wd:N \l_tmpb_box - \nulldelimiterspace }
1751 \hbox_set:Nn \l_tmpb_box
1752 {
1753   \c_math_toggle_token
1754   \left .
1755   \vbox_to_ht:nn
1756   { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1757   { }
1758   \right #2
1759   \c_math_toggle_token
1760 }
1761 \dim_set:Nn \l_@@_real_right_delim_dim
1762 { \box_wd:N \l_tmpb_box - \nulldelimiterspace }

```

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

```

1763 \skip_horizontal:N \l_@@_left_delim_dim
1764 \skip_horizontal:N -\l_@@_real_left_delim_dim
1765 \@@_put_box_in_flow:
1766 \skip_horizontal:N \l_@@_right_delim_dim
1767 \skip_horizontal:N -\l_@@_real_right_delim_dim
1768 }

```

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).

```

1769 \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).

```

1770 {
1771   \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`).

```

1772 { \exp_args:NV \@@_array: \g_@@_preamble_tl }
1773 }
1774 {
1775   \@@_create_col_nodes:
1776   \endarray
1777 }

```

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`).

```
1778 \NewDocumentEnvironment { @@-light-syntax } { b }
1779 {
```

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`.

```
1780 \tl_if_empty:nT { #1 } { @@_fatal:n { empty-environment } }
1781 \tl_map_inline:nn { #1 }
1782 {
1783   \tl_if_eq:nnT { ##1 } { & }
1784   { @@_fatal:n { ampersand-in-light-syntax } }
1785   \tl_if_eq:nnT { ##1 } { \ }
1786   { @@_fatal:n { double-backslash-in-light-syntax } }
1787 }
```

Now, you extract the `code-after` of the body of the environment. Maybe, there is no command `\CodeAfter` in the body. That's why you put a marker `\CodeAfter` after `#1`. If there is yet a `\CodeAfter` in `#1`, this second (or third...) `\CodeAfter` will be caught in the value of `\g_nicematrix_code_after_tl`. That doesn't matter because `\CodeAfter` will be set to *no-op* before the execution of `\g_nicematrix_code_after_tl`.

```
1788 @@_light_syntax_i #1 \CodeAfter \q_stop
1789 }
```

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`.

```
1790 { }
1791 \cs_new_protected:Npn @@_light_syntax_i #1\CodeAfter #2\q_stop
1792 {
1793   \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).

```
1794 \seq_gclear_new:N \g_@@_rows_seq
1795 \tl_set_rescan:Nno \l_@@_end_of_row_tl { } \l_@@_end_of_row_tl
1796 \exp_args:NV \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.

```
1797 \int_compare:nNnT \l_@@_last_row_int = { -1 }
1798 { \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`).

```
1799 \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.

```
1800 \seq_gpop_left:NN \g_@@_rows_seq \l_tmpa_tl
1801 \exp_args:NV @@_line_with_light_syntax_i:n \l_tmpa_tl
1802 \seq_map_function:NN \g_@@_rows_seq @@_line_with_light_syntax:n
1803 @@_create_col_nodes:
1804 \endarray
1805 }
1806 \cs_new_protected:Npn @@_line_with_light_syntax:n #1
1807 { \tl_if_empty:nF { #1 } { \ \ @@_line_with_light_syntax_i:n { #1 } } }
1808 \cs_new_protected:Npn @@_line_with_light_syntax_i:n #1
1809 {
1810   \seq_gclear_new:N \g_@@_cells_seq
1811   \seq_gset_split:Nnn \g_@@_cells_seq { ~ } { #1 }
1812   \seq_gpop_left:NN \g_@@_cells_seq \l_tmpa_tl
1813   \l_tmpa_tl
1814   \seq_map_inline:Nn \g_@@_cells_seq { & ##1 }
1815 }
```


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).

```

1816 \cs_new_protected:Npn \@@_analyze_end:Nn #1 #2
1817 {
1818   \str_if_eq:VnT \g_@@_name_env_str { #2 }
1819   { \@@_fatal:n { empty-environment } }

```

We repeat in the stream the `\end{...}` we have extracted and the user will have an error for incorrect nested environments.

```

1820   \end { #2 }
1821 }

```

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

```

1822 \cs_new:Npn \@@_create_col_nodes:
1823 {
1824   \crrc
1825   \int_compare:nNnT \l_@@_first_col_int = 0
1826   {
1827     \omit
1828     \hbox_overlap_left:n
1829     {
1830       \bool_if:NT \l_@@_code_before_bool
1831       { \pgfsys@markposition { \@@_env: - col - 0 } }
1832       \pgfpicture
1833       \pgfrememberpicturepositiononpagetrue
1834       \pgfcoordinate { \@@_env: - col - 0 } \pgfpointorigin
1835       \str_if_empty:NF \l_@@_name_str
1836       { \pgfnodealias { \l_@@_name_str - col - 0 } { \@@_env: - col - 0 } }
1837       \endpgfpicture
1838       \skip_horizontal:N 2\col@sep
1839       \skip_horizontal:N \g_@@_width_first_col_dim
1840     }
1841     &
1842   }
1843   \omit

```

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

```

1844   \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`).

```

1845   \int_compare:nNnTF \l_@@_first_col_int = 0
1846   {
1847     \bool_if:NT \l_@@_code_before_bool
1848     {
1849       \hbox
1850       {
1851         \skip_horizontal:N -0.5\arrayrulewidth
1852         \pgfsys@markposition { \@@_env: - col - 1 }
1853         \skip_horizontal:N 0.5\arrayrulewidth
1854       }
1855     }
1856     \pgfpicture
1857     \pgfrememberpicturepositiononpagetrue
1858     \pgfcoordinate { \@@_env: - col - 1 }
1859     { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
1860     \str_if_empty:NF \l_@@_name_str
1861     { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
1862     \endpgfpicture
1863   }
1864   {

```

```

1865 \bool_if:NT \l_@@_code_before_bool
1866 {
1867     \hbox
1868     {
1869         \skip_horizontal:N 0.5 \arrayrulewidth
1870         \pgfsys@markposition { \@@_env: - col - 1 }
1871         \skip_horizontal:N -0.5\arrayrulewidth
1872     }
1873 }
1874 \pgfpicture
1875 \pgfrememberpicturepositiononpagetrue
1876 \pgfcoordinate { \@@_env: - col - 1 }
1877 { \pgfpoint { 0.5 \arrayrulewidth } \c_zero_dim }
1878 \str_if_empty:NF \l_@@_name_str
1879 { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
1880 \endpgfpicture
1881 }

```

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.

```

1882 \skip_gset:Nn \g_tmpa_skip { 0 pt+plus 1 fill }
1883 \bool_if:NF \l_@@_auto_columns_width_bool
1884 { \dim_compare:nNt \l_@@_columns_width_dim > \c_zero_dim }
1885 {
1886     \bool_lazy_and:nnTF
1887     \l_@@_auto_columns_width_bool
1888     { \bool_not_p:n \l_@@_block_auto_columns_width_bool }
1889     { \skip_gset_eq:NN \g_tmpa_skip \g_@@_max_cell_width_dim }
1890     { \skip_gset_eq:NN \g_tmpa_skip \l_@@_columns_width_dim }
1891     \skip_gadd:Nn \g_tmpa_skip { 2 \col@sep }
1892 }
1893 \skip_horizontal:N \g_tmpa_skip
1894 \hbox
1895 {
1896     \bool_if:NT \l_@@_code_before_bool
1897     {
1898         \hbox
1899         {
1900             \skip_horizontal:N -0.5\arrayrulewidth
1901             \pgfsys@markposition { \@@_env: - col - 2 }
1902             \skip_horizontal:N 0.5\arrayrulewidth
1903         }
1904     }
1905     \pgfpicture
1906     \pgfrememberpicturepositiononpagetrue
1907     \pgfcoordinate { \@@_env: - col - 2 }
1908     { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
1909     \str_if_empty:NF \l_@@_name_str
1910     { \pgfnodealias { \l_@@_name_str - col - 2 } { \@@_env: - col - 2 } }
1911     \endpgfpicture
1912 }

```

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.

```

1913 \int_gset:Nn \g_tmpa_int 1
1914 \bool_if:NTF \g_@@_last_col_found_bool
1915 { \prg_replicate:nn { \g_@@_col_total_int - 2 } }
1916 { \prg_replicate:nn { \g_@@_col_total_int - 1 } }
1917 {
1918     &

```

```

1919     \omit
1920     \int_gincr:N \g_tmpa_int

```

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

```

1921     \skip_horizontal:N \g_tmpa_skip
1922     \bool_if:NT \l_@@_code_before_bool
1923     {
1924         \hbox
1925         {
1926             \skip_horizontal:N -0.5\arrayrulewidth
1927             \pgfsys@markposition { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1928             \skip_horizontal:N 0.5\arrayrulewidth
1929         }
1930     }

```

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

```

1931     \pgfpicture
1932     \pgfrememberpicturepositiononpagetrue
1933     \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1934     { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
1935     \str_if_empty:NF \l_@@_name_str
1936     {
1937         \pgfnodealias
1938         { \l_@@_name_str - col - \@@_succ:n \g_tmpa_int }
1939         { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1940     }
1941     \endpgfpicture
1942 }
1943 \bool_if:NT \g_@@_last_col_found_bool
1944 {
1945     \hbox_overlap_right:n
1946     {
1947         % \skip_horizontal:N \col@sep
1948         \skip_horizontal:N \g_@@_width_last_col_dim
1949         \bool_if:NT \l_@@_code_before_bool
1950         {
1951             \pgfsys@markposition
1952             { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
1953         }
1954         \pgfpicture
1955         \pgfrememberpicturepositiononpagetrue
1956         \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
1957         \pgfpointorigin
1958         \str_if_empty:NF \l_@@_name_str
1959         {
1960             \pgfnodealias
1961             { \l_@@_name_str - col - \@@_succ:n \g_@@_col_total_int }
1962             { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
1963         }
1964         \endpgfpicture
1965     }
1966 }
1967 \cr
1968 }

```

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

```

1969 \tl_const:Nn \c_@@_preamble_first_col_tl
1970 {
1971     >
1972     {
1973         \@@_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.

```

1974     \hbox_set:Nw \l_@@_cell_box
1975     \@@_math_toggle_token:
1976     \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”.

```

1977     \bool_lazy_and:nnT
1978     { \int_compare_p:nNn \c@iRow > 0 }
1979     {
1980         \bool_lazy_or_p:nn
1981         { \int_compare_p:nNn \l_@@_last_row_int < 0 }
1982         { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
1983     }
1984     {
1985         \l_@@_code_for_first_col_tl
1986         \xglobal \colorlet { nicematrix-first-col } { . }
1987     }
1988 }

```

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`.

```

1989     l
1990     <
1991     {
1992         \@@_math_toggle_token:
1993         \hbox_set_end:
1994         \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
1995         \@@_adjust_width_box:
1996         \@@_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.

```

1997     \dim_gset:Nn \g_@@_width_first_col_dim
1998     { \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.

```

1999     \hbox_overlap_left:n
2000     {
2001         \dim_compare:nNnTF { \box_wd:N \l_@@_cell_box } > \c_zero_dim
2002         \@@_node_for_the_cell:
2003         { \box_use_drop_x:N \l_@@_cell_box }
2004         \skip_horizontal:N \l_@@_left_delim_dim
2005         \skip_horizontal:N \l_@@_left_margin_dim
2006         \skip_horizontal:N \l_@@_extra_left_margin_dim
2007     }
2008     \bool_gset_false:N \g_@@_empty_cell_bool
2009     \skip_horizontal:N -2\col@sep
2010 }
2011 }

```

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

```

2012 \tl_const:Nn \c_@@_preamble_last_col_tl
2013 {
2014     >
2015     {

```

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

```

2016     \bool_gset_true:N \g_@@_last_col_found_bool
2017     \int_gincr:N \c@jCol
2018     \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.

```
2019 \hbox_set:Nw \l_@@_cell_box
2020 \@@_math_toggle_token:
2021 \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”.

```
2022 \int_compare:nNnT \c@iRow > 0
2023 {
2024   \bool_lazy_or:nnT
2025     { \int_compare_p:nNn \l_@@_last_row_int < 0 }
2026     { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
2027   {
2028     \l_@@_code_for_last_col_tl
2029     \xglobal \colorlet { nicematrix-last-col } { . }
2030   }
2031 }
2032 }
2033 1
2034 <
2035 {
2036   \@@_math_toggle_token:
2037   \hbox_set_end:
2038   \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
2039   \@@_adjust_width_box:
2040   \@@_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.

```
2041 \dim_gset:Nn \g_@@_width_last_col_dim
2042 { \dim_max:nn \g_@@_width_last_col_dim { \box_wd:N \l_@@_cell_box } }
2043 \skip_horizontal:N -2\col@sep
```

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

```
2044 \hbox_overlap_right:n
2045 {
2046   \dim_compare:nNnT { \box_wd:N \l_@@_cell_box } > \c_zero_dim
2047   {
2048     \skip_horizontal:N \l_@@_right_delim_dim
2049     \skip_horizontal:N \l_@@_right_margin_dim
2050     \skip_horizontal:N \l_@@_extra_right_margin_dim
2051     \@@_node_for_the_cell:
2052   }
2053 }
2054 \bool_gset_false:N \g_@@_empty_cell_bool
2055 }
2056 }
```

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.

```
2057 \NewDocumentEnvironment { NiceArray } { }
2058 {
2059   \bool_set_true:N \l_@@_NiceArray_bool
2060   \str_if_empty:NT \g_@@_name_env_str
2061   { \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).

```
2062 \NiceArrayWithDelims . .
2063 }
2064 { \endNiceArrayWithDelims }
```

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

```

2065 \cs_new_protected:Npn \@@_def_env:nnn #1 #2 #3
2066 {
2067   \NewDocumentEnvironment { #1 NiceArray } { }
2068   {
2069     \str_if_empty:NT \g_@@_name_env_str
2070     { \str_gset:Nn \g_@@_name_env_str { #1 NiceArray } }
2071     \@@_test_if_math_mode:
2072     \NiceArrayWithDelims #2 #3
2073   }
2074   { \endNiceArrayWithDelims }
2075 }
2076 \@@_def_env:nnn p ( )
2077 \@@_def_env:nnn b [ ]
2078 \@@_def_env:nnn B \{ \}
2079 \@@_def_env:nnn v | |
2080 \@@_def_env:nnn V \| \|

```

The environment `{NiceMatrix}` and its variants

```

2081 \cs_new_protected:Npn \@@_begin_of_NiceMatrix:nn #1 #2
2082 {
2083   \bool_set_true:N \l_@@_Matrix_bool
2084   \use:c { #1 NiceArray }
2085   {
2086     *
2087     {
2088       \int_compare:nNnTF \l_@@_last_col_int < 0
2089       \c@MaxMatrixCols
2090       { \@@_pred:n \l_@@_last_col_int }
2091     }
2092     { > \@@_Cell: #2 < \@@_end_Cell: }
2093   }
2094 }
2095 \clist_map_inline:nn { { } , p , b , B , v , V }
2096 {
2097   \NewDocumentEnvironment { #1 NiceMatrix } { ! 0 { } }
2098   {
2099     \str_gset:Nn \g_@@_name_env_str { #1 NiceMatrix }
2100     \tl_set:Nn \l_@@_type_of_col_tl c
2101     \keys_set:nn { NiceMatrix / NiceMatrix } { ##1 }
2102     \exp_args:Nne \@@_begin_of_NiceMatrix:nn { #1 } \l_@@_type_of_col_tl
2103   }
2104   { \use:c { end #1 NiceArray } }
2105 }

```

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

```

2106 \NewDocumentEnvironment { NiceTabular } { 0 { } m ! 0 { } }
2107 {
2108   \str_gset:Nn \g_@@_name_env_str { NiceTabular }
2109   \keys_set:nn { NiceMatrix / NiceTabular } { #1 , #3 }
2110   \bool_set_true:N \l_@@_NiceTabular_bool
2111   \NiceArray { #2 }
2112 }
2113 { \endNiceArray }
2114 \NewDocumentEnvironment { NiceTabular* } { m 0 { } m ! 0 { } }
2115 {

```

```

2116 \str_gset:Nn \g_@@_name_env_str { NiceTabular* }
2117 \dim_set:Nn \l_@@_tabular_width_dim { #1 }
2118 \keys_set:nn { NiceMatrix / NiceTabular } { #2 , #4 }
2119 \bool_set_true:N \l_@@_NiceTabular_bool
2120 \NiceArray { #3 }
2121 }
2122 { \endNiceArray }

```

After the construction of the array

```

2123 \cs_new_protected:Npn \@@_after_array:
2124 {
2125   \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.

```

2126 \bool_if:NT \g_@@_last_col_found_bool
2127 { \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`.

```

2128 \bool_if:NT \l_@@_last_col_without_value_bool
2129 {
2130   \dim_set_eq:NN \l_@@_last_col_int \g_@@_col_total_int
2131   \iow_shipout:Nn \@mainaux \ExplSyntaxOn
2132   \iow_shipout:Nx \@mainaux
2133   {
2134     \cs_gset:cpn { @@_last_col_ \int_use:N \g_@@_env_int }
2135     { \int_use:N \g_@@_col_total_int }
2136   }
2137   \str_if_empty:NF \l_@@_name_str
2138   {
2139     \iow_shipout:Nx \@mainaux
2140     {
2141       \cs_gset:cpn { @@_last_col_ \l_@@_name_str }
2142       { \int_use:N \g_@@_col_total_int }
2143     }
2144   }
2145   \iow_shipout:Nn \@mainaux \ExplSyntaxOff
2146 }

```

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.

```

2147 \bool_if:NT \l_@@_last_row_without_value_bool
2148 {
2149   \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.

```

2150 \bool_if:NF \l_@@_light_syntax_bool
2151 {
2152   \iow_shipout:Nn \@mainaux \ExplSyntaxOn
2153   \iow_shipout:Nx \@mainaux
2154   {
2155     \cs_gset:cpn { @@_last_row_ \int_use:N \g_@@_env_int }
2156     { \int_use:N \g_@@_row_total_int }
2157   }

```

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.

```

2158 \str_if_empty:NF \l_@@_name_str

```

```

2159         {
2160             \iow_shipout:Nx \@mainaux
2161             {
2162                 \cs_gset:cpn { @@_last_row_ \l_@@_name_str }
2163                 { \int_use:N \g_@@_row_total_int }
2164             }
2165         }
2166         \iow_shipout:Nn \@mainaux \ExplSyntaxOff
2167     }
2168 }

```

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

```

2169     \bool_if:NT \l_@@_code_before_bool
2170     {
2171         \iow_now:Nn \@mainaux \ExplSyntaxOn
2172         \iow_now:Nx \@mainaux
2173         { \seq_clear_new:c { @@_size _ \int_use:N \g_@@_env_int _ seq } }
2174         \iow_now:Nx \@mainaux
2175         {
2176             \seq_gset_from_clist:cn { @@_size _ \int_use:N \g_@@_env_int _ seq }
2177             {
2178                 \int_use:N \l_@@_first_row_int ,
2179                 \int_use:N \g_@@_row_total_int ,
2180                 \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`.

```

2181         \bool_lazy_and:nnTF
2182         { \int_compare_p:nNn \l_@@_last_col_int > { -2 } }
2183         { \bool_not_p:n \g_@@_last_col_found_bool }
2184         \@@_succ:n
2185         \int_use:N
2186         \g_@@_col_total_int
2187     }

```

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`).

```

2188         \seq_gset_from_clist:cn
2189         { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq }
2190         { \seq_use:Nnnn \g_@@_pos_of_blocks_seq , , , }
2191     }
2192     \iow_now:Nn \@mainaux \ExplSyntaxOff
2193 }

```

By default, the diagonal lines will be parallelized⁴⁰. 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.

```

2194     \bool_if:NT \l_@@_parallelize_diags_bool
2195     {
2196         \int_gzero_new:N \g_@@_ddots_int
2197         \int_gzero_new:N \g_@@_iddots_int

```

The dimensions `\g_@@_delta_x_one_dim` and `\g_@@_delta_y_one_dim` will contain the Δ_x and Δ_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 Δ_x and Δ_y of the first `\Iddots` diagonal.

```

2198         \dim_gzero_new:N \g_@@_delta_x_one_dim
2199         \dim_gzero_new:N \g_@@_delta_y_one_dim
2200         \dim_gzero_new:N \g_@@_delta_x_two_dim
2201         \dim_gzero_new:N \g_@@_delta_y_two_dim
2202     }
2203     \bool_if:nTF \l_@@_medium_nodes_bool
2204     {

```

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


```

2205     \bool_if:NTF \l_@@_large_nodes_bool
2206     \@@_create_medium_and_large_nodes:
2207     \@@_create_medium_nodes:
2208   }
2209   { \bool_if:NT \l_@@_large_nodes_bool \@@_create_large_nodes: }
2210   \int_zero_new:N \l_@@_initial_i_int
2211   \int_zero_new:N \l_@@_initial_j_int
2212   \int_zero_new:N \l_@@_final_i_int
2213   \int_zero_new:N \l_@@_final_j_int
2214   \bool_set_false:N \l_@@_initial_open_bool
2215   \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.

```

2216   \bool_if:NT \l_@@_small_bool
2217   {
2218     \dim_set:Nn \l_@@_radius_dim { 0.37 pt }
2219     \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.

```

2220     \dim_set:Nn \l_@@_xdots_shorten_dim { 0.6 \l_@@_xdots_shorten_dim }
2221   }

```

Now, we actually draw the dotted lines (specified by `\Cdots`, `\Vdots`, etc.).

```

2222   \@@_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.

```

2223   \@@_compute_corners:

```

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`.

```

2224   \bool_lazy_all:nT
2225   {
2226     { \seq_if_empty_p:N \g_@@_pos_of_blocks_seq }
2227     { \seq_if_empty_p:N \g_@@_pos_of_xdots_seq }
2228     { \seq_if_empty_p:N \l_@@_empty_corner_cells_seq }
2229   }
2230   {
2231     \cs_set_eq:NN \@@_vline_i:nn \@@_vline_i_complete:nn
2232     \cs_set_eq:NN \@@_hline_i:nn \@@_hline_i_complete:nn
2233   }
2234   \bool_if:NT \l_@@_hlines_bool \@@_draw_hlines:
2235   \bool_if:NT \l_@@_vlines_bool \@@_draw_vlines:

```

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

```

2236   \cs_set_eq:NN \ialign \@@_old_ialign:
2237   \seq_if_empty:NF \g_@@_blocks_seq \@@_draw_blocks:
2238   \g_@@_internal_code_after_tl
2239   \tl_gclear:N \g_@@_internal_code_after_tl

```

Now, the code-after.

```

2240   \bool_if:NT \c_@@_tikz_loaded_bool
2241   {
2242     \tikzset

```

```

2243     {
2244         every~picture / .style =
2245         {
2246             overlay ,
2247             remember~picture ,
2248             name~prefix = \@@_env: -
2249         }
2250     }
2251 }
2252 \cs_set_eq:NN \line \@@_line

```

When `light-syntax` is used, we insert systematically a `\CodeAfter` in the flow. Thus, it's possible to have two instructions `\CodeAfter` and the second may be in `\g_nicematrix_code_after_tl`. That's why we set `\Code-after` to be *no-op* now.

```

2253 \cs_set_eq:NN \CodeAfter \prg_do_nothing:

```

And here's the `code-after`:

```

2254 \g_nicematrix_code_after_tl
2255 \tl_gclear:N \g_nicematrix_code_after_tl
2256 \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.

```

2257 \tl_if_empty:NF \g_nicematrix_code_before_tl
2258 {

```

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`.

```

2259 \cs_set_protected:Npn \rectanglecolor { }
2260 \cs_set_protected:Npn \columncolor { }
2261 \iow_now:Nn \@mainaux \ExplSyntaxOn
2262 \iow_now:Nx \@mainaux
2263 {
2264     \tl_gset:cn
2265     { g_@@_code_before_ \int_use:N \g_@@_env_int _ tl }
2266     { \g_nicematrix_code_before_tl }
2267 }
2268 \iow_now:Nn \@mainaux \ExplSyntaxOff
2269 \bool_set_true:N \l_@@_code_before_bool
2270 }

2271 \str_gclear:N \g_@@_name_env_str
2272 \@@_restore_iRow_jCol:

```

The command `\CT@arc@` contains the instruction of color for the rules of the array⁴¹. 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`.

```

2273 \cs_gset_eq:NN \CT@arc@ \@@_old_CT@arc@
2274 }

```

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).

```

2275 \AtBeginDocument
2276 {
2277     \cs_new_protected:Npx \@@_draw_dotted_lines:
2278     {
2279         \c_@@_pgfortikzpicture_tl
2280         \@@_draw_dotted_lines_i:

```

⁴¹e.g. `\color[rgb]{0.5,0.5,0}`)

```

2281         \c_@@_endpgfortikzpicture_tl
2282     }
2283 }

```

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

```

2284 \cs_new_protected:Npn \@@_draw_dotted_lines_i:
2285 {
2286     \pgfrememberpicturepositiononpagetrue
2287     \pgf@relevantforpicturesizefalse
2288     \g_@@_HVdotsfor_lines_tl
2289     \g_@@_Vdots_lines_tl
2290     \g_@@_Ddots_lines_tl
2291     \g_@@_Iddots_lines_tl
2292     \g_@@_Cdots_lines_tl
2293     \g_@@_Ldots_lines_tl
2294 }

2295 \cs_new_protected:Npn \@@_restore_iRow_jCol:
2296 {
2297     \cs_if_exist:NT \theiRow { \int_gset_eq:NN \c@iRow \l_@@_old_iRow_int }
2298     \cs_if_exist:NT \thejCol { \int_gset_eq:NN \c@jCol \l_@@_old_jCol_int }
2299 }

```

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.

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

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

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

This command computes:

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

```

2300 \cs_new_protected:Npn \@@_find_extremities_of_line:nnnn #1 #2 #3 #4
2301 {

```

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

```

2302     \cs_set:cpn { @@ _ dotted _ #1 - #2 } { }

```

Initialization of variables.

```

2303     \int_set:Nn \l_@@_initial_i_int { #1 }
2304     \int_set:Nn \l_@@_initial_j_int { #2 }
2305     \int_set:Nn \l_@@_final_i_int { #1 }
2306     \int_set:Nn \l_@@_final_j_int { #2 }

```

We will do two loops: one when determinating the initial cell and the other when determinating 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.

```

2307     \bool_set_false:N \l_@@_stop_loop_bool
2308     \bool_do_until:Nn \l_@@_stop_loop_bool
2309     {
2310         \int_add:Nn \l_@@_final_i_int { #3 }
2311         \int_add:Nn \l_@@_final_j_int { #4 }

```

We test if we are still in the matrix.

```

2312     \bool_set_false:N \l_@@_final_open_bool
2313     \int_compare:nNnTF \l_@@_final_i_int > \c@iRow
2314     {
2315         \int_compare:nNnTF { #3 } = 1
2316         { \bool_set_true:N \l_@@_final_open_bool }
2317         {
2318             \int_compare:nNnT \l_@@_final_j_int > \c@jCol
2319             { \bool_set_true:N \l_@@_final_open_bool }
2320         }
2321     }
2322     {
2323         \int_compare:nNnTF \l_@@_final_j_int < 1
2324         {
2325             \int_compare:nNnT { #4 } = { -1 }
2326             { \bool_set_true:N \l_@@_final_open_bool }
2327         }
2328         {
2329             \int_compare:nNnT \l_@@_final_j_int > \c@jCol
2330             {
2331                 \int_compare:nNnT { #4 } = 1
2332                 { \bool_set_true:N \l_@@_final_open_bool }
2333             }
2334         }
2335     }
2336     \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.

```

2337     {

```

We do a step backwards.

```

2338         \int_sub:Nn \l_@@_final_i_int { #3 }
2339         \int_sub:Nn \l_@@_final_j_int { #4 }
2340         \bool_set_true:N \l_@@_stop_loop_bool
2341     }

```

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`.

```

2342     {
2343         \cs_if_exist:cTF
2344         {
2345             @@ _ dotted _
2346             \int_use:N \l_@@_final_i_int -
2347             \int_use:N \l_@@_final_j_int
2348         }
2349         {
2350             \int_sub:Nn \l_@@_final_i_int { #3 }
2351             \int_sub:Nn \l_@@_final_j_int { #4 }
2352             \bool_set_true:N \l_@@_final_open_bool
2353             \bool_set_true:N \l_@@_stop_loop_bool
2354         }
2355     }
2356     \cs_if_exist:cTF
2357     {
2358         pgf @ sh @ ns @ \@@_env:
2359         - \int_use:N \l_@@_final_i_int

```

```

2360         - \int_use:N \l_@@_final_j_int
2361     }
2362     { \bool_set_true:N \l_@@_stop_loop_bool }

```

If the case is empty, we declare that the cell as non-empty. Indeed, we will draw a dotted line and the cell will be on that dotted line. All the cells of a dotted line have to be marked as “dotted” because we don’t want intersections between dotted lines. We recall that the research of the extremities of the lines are all done in the same TeX group (the group of the environnement), even though, when the extremities are found, each line is drawn in a TeX group that we will open for the options of the line.

```

2363     {
2364         \cs_set:cpn
2365         {
2366             @@ _ dotted _
2367             \int_use:N \l_@@_final_i_int -
2368             \int_use:N \l_@@_final_j_int
2369         }
2370         { }
2371     }
2372 }
2373 }
2374 }

```

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

```

2375 \bool_set_false:N \l_@@_stop_loop_bool
2376 \bool_do_until:Nn \l_@@_stop_loop_bool
2377 {
2378     \int_sub:Nn \l_@@_initial_i_int { #3 }
2379     \int_sub:Nn \l_@@_initial_j_int { #4 }
2380     \bool_set_false:N \l_@@_initial_open_bool
2381     \int_compare:nNnTF \l_@@_initial_i_int < 1
2382     {
2383         \int_compare:nNnTF { #3 } = 1
2384         { \bool_set_true:N \l_@@_initial_open_bool }
2385         {
2386             \int_compare:nNnT \l_@@_initial_j_int = 0
2387             { \bool_set_true:N \l_@@_initial_open_bool }
2388         }
2389     }
2390     {
2391         \int_compare:nNnTF \l_@@_initial_j_int < 1
2392         {
2393             \int_compare:nNnT { #4 } = 1
2394             { \bool_set_true:N \l_@@_initial_open_bool }
2395         }
2396         {
2397             \int_compare:nNnT \l_@@_initial_j_int > \c@jCol
2398             {
2399                 \int_compare:nNnT { #4 } = { -1 }
2400                 { \bool_set_true:N \l_@@_initial_open_bool }
2401             }
2402         }
2403     }
2404     \bool_if:NTF \l_@@_initial_open_bool
2405     {
2406         \int_add:Nn \l_@@_initial_i_int { #3 }
2407         \int_add:Nn \l_@@_initial_j_int { #4 }
2408         \bool_set_true:N \l_@@_stop_loop_bool
2409     }
2410     {
2411         \cs_if_exist:cTF

```

```

2412     {
2413         @@ _ dotted _
2414         \int_use:N \l_@@_initial_i_int -
2415         \int_use:N \l_@@_initial_j_int
2416     }
2417     {
2418         \int_add:Nn \l_@@_initial_i_int { #3 }
2419         \int_add:Nn \l_@@_initial_j_int { #4 }
2420         \bool_set_true:N \l_@@_initial_open_bool
2421         \bool_set_true:N \l_@@_stop_loop_bool
2422     }
2423     {
2424         \cs_if_exist:cTF
2425         {
2426             pgf @ sh @ ns @ \@@_env:
2427             - \int_use:N \l_@@_initial_i_int
2428             - \int_use:N \l_@@_initial_j_int
2429         }
2430         { \bool_set_true:N \l_@@_stop_loop_bool }
2431         {
2432             \cs_set:cpn
2433             {
2434                 @@ _ dotted _
2435                 \int_use:N \l_@@_initial_i_int -
2436                 \int_use:N \l_@@_initial_j_int
2437             }
2438             { }
2439         }
2440     }
2441 }
2442 }

```

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.

```

2443     \seq_gput_right:Nx \g_@@_pos_of_xdots_seq
2444     {
2445         { \int_use:N \l_@@_initial_i_int }
2446         { \int_use:N \l_@@_initial_j_int }
2447         { \int_use:N \l_@@_final_i_int }
2448         { \int_use:N \l_@@_final_j_int }
2449     }
2450 }

2451 \cs_new_protected:Npn \@@_set_initial_coords:
2452 {
2453     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2454     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2455 }
2456 \cs_new_protected:Npn \@@_set_final_coords:
2457 {
2458     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2459     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2460 }
2461 \cs_new_protected:Npn \@@_set_initial_coords_from_anchor:n #1
2462 {
2463     \pgfpointanchor
2464     {
2465         \@@_env:
2466         - \int_use:N \l_@@_initial_i_int
2467         - \int_use:N \l_@@_initial_j_int
2468     }
2469     { #1 }
2470     \@@_set_initial_coords:
2471 }

```

```

2472 \cs_new_protected:Npn \@@_set_final_coords_from_anchor:n #1
2473 {
2474   \pgfpointanchor
2475   {
2476     \@@_env:
2477     - \int_use:N \l_@@_final_i_int
2478     - \int_use:N \l_@@_final_j_int
2479   }
2480   { #1 }
2481   \@@_set_final_coords:
2482 }

```

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.

```

2483 \cs_new_protected:Npn \@@_draw_Ldots:nnn #1 #2 #3
2484 {
2485   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2486   {
2487     \@@_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.

```

2488     \group_begin:
2489     \int_compare:nNnTF { #1 } = 0
2490     { \color { nicematrix-first-row } }
2491     {

```

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.

```

2492         \int_compare:nNnT { #1 } = \l_@@_last_row_int
2493         { \color { nicematrix-last-row } }
2494     }
2495     \keys_set:nn { NiceMatrix / xdots } { #3 }
2496     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2497     \@@_actually_draw_Ldots:
2498   \group_end:
2499 }
2500 }

```

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`.

```

2501 \cs_new_protected:Npn \@@_actually_draw_Ldots:
2502 {
2503   \bool_if:NTF \l_@@_initial_open_bool
2504   {
2505     \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2506     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2507     \dim_add:Nn \l_@@_x_initial_dim \col@sep
2508     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int - base }
2509     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2510   }

```

```

2511     { \@@_set_initial_coords_from_anchor:n { base-east } }
2512 \bool_if:NTF \l_@@_final_open_bool
2513 {
2514     \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
2515     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2516     \dim_sub:Nn \l_@@_x_final_dim \col@sep
2517     \@@_qpoint:n { row - \int_use:N \l_@@_final_i_int - base }
2518     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2519 }
2520 { \@@_set_final_coords_from_anchor:n { base-west } }

```

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

```

2521     \dim_add:Nn \l_@@_y_initial_dim \l_@@_radius_dim
2522     \dim_add:Nn \l_@@_y_final_dim \l_@@_radius_dim
2523     \@@_draw_line:
2524 }

```

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.

```

2525 \cs_new_protected:Npn \@@_draw_Cdots:nnn #1 #2 #3
2526 {
2527     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2528     {
2529         \@@_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.

```

2530     \group_begin:
2531     \int_compare:nNnTF { #1 } = 0
2532     { \color { nicematrix-first-row } }
2533     {

```

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.

```

2534         \int_compare:nNnT { #1 } = \l_@@_last_row_int
2535         { \color { nicematrix-last-row } }
2536     }
2537     \keys_set:nn { NiceMatrix / xdots } { #3 }
2538     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2539     \@@_actually_draw_Cdots:
2540 \group_end:
2541 }
2542 }

```

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`.

```

2543 \cs_new_protected:Npn \@@_actually_draw_Cdots:
2544 {
2545     \bool_if:NTF \l_@@_initial_open_bool
2546     {
2547         \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2548         \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x

```



```

2549     \dim_add:Nn \l_@@_x_initial_dim \col@sep
2550   }
2551   { \@@_set_initial_coords_from_anchor:n { mid~east } }
2552 \bool_if:NTF \l_@@_final_open_bool
2553 {
2554   \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
2555   \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2556   \dim_sub:Nn \l_@@_x_final_dim \col@sep
2557 }
2558 { \@@_set_final_coords_from_anchor:n { mid~west } }
2559 \bool_lazy_and:nnTF
2560 \l_@@_initial_open_bool
2561 \l_@@_final_open_bool
2562 {
2563   \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2564   \dim_set_eq:NN \l_tmpa_dim \pgf@y
2565   \@@_qpoint:n { row - \@@_succ:n \l_@@_initial_i_int }
2566   \dim_set:Nn \l_@@_y_initial_dim { ( \l_tmpa_dim + \pgf@y ) / 2 }
2567   \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim
2568 }
2569 {
2570   \bool_if:NT \l_@@_initial_open_bool
2571     { \dim_set_eq:NN \l_@@_y_initial_dim \l_@@_y_final_dim }
2572   \bool_if:NT \l_@@_final_open_bool
2573     { \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim }
2574 }
2575 \@@_draw_line:
2576 }

```

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.

```

2577 \cs_new_protected:Npn \@@_draw_Vdots:nnn #1 #2 #3
2578 {
2579   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2580   {
2581     \@@_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.

```

2582   \group_begin:
2583     \int_compare:nNnTF { #2 } = 0
2584       { \color { nicematrix-first-col } }
2585       {
2586         \int_compare:nNnT { #2 } = \l_@@_last_col_int
2587         { \color { nicematrix-last-col } }
2588       }
2589     \keys_set:nn { NiceMatrix / xdots } { #3 }
2590     \tl_if_empty:VF \l_@@_xdots_color_tl
2591       { \color { \l_@@_xdots_color_tl } }
2592     \@@_actually_draw_Vdots:
2593   \group_end:
2594 }
2595 }

```

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`.

```
2596 \cs_new_protected:Npn \@@_actually_draw_Vdots:
2597 {
```

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

```
2598   \bool_set_false:N \l_tmpa_bool
2599   \bool_lazy_or:nnF \l_@@_initial_open_bool \l_@@_final_open_bool
2600   {
2601     \@@_set_initial_coords_from_anchor:n { south~west }
2602     \@@_set_final_coords_from_anchor:n { north~west }
2603     \bool_set:Nn \l_tmpa_bool
2604     { \dim_compare_p:nNn \l_@@_x_initial_dim = \l_@@_x_final_dim }
2605   }
```

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

```
2606   \bool_if:NTF \l_@@_initial_open_bool
2607   {
2608     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2609     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2610   }
2611   { \@@_set_initial_coords_from_anchor:n { south } }
2612   \bool_if:NTF \l_@@_final_open_bool
2613   {
2614     \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }
2615     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2616   }
2617   { \@@_set_final_coords_from_anchor:n { north } }
2618   \bool_if:NTF \l_@@_initial_open_bool
2619   {
2620     \bool_if:NTF \l_@@_final_open_bool
2621     {
2622       \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2623       \dim_set_eq:NN \l_tmpa_dim \pgf@x
2624       \@@_qpoint:n { col - \@@_succ:n \l_@@_initial_j_int }
2625       \dim_set:Nn \l_@@_x_initial_dim { ( \pgf@x + \l_tmpa_dim ) / 2 }
2626       \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.

```
2627       \int_compare:nNnT \l_@@_last_col_int > { -2 }
2628       {
2629         \int_compare:nNnT \l_@@_initial_j_int = \g_@@_col_total_int
2630         {
2631           \dim_set_eq:NN \l_tmpa_dim \l_@@_right_margin_dim
2632           \dim_add:Nn \l_tmpa_dim \l_@@_extra_right_margin_dim
2633           \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
2634           \dim_add:Nn \l_@@_x_final_dim \l_tmpa_dim
2635         }
2636       }
2637     }
2638     { \dim_set_eq:NN \l_@@_x_initial_dim \l_@@_x_final_dim }
2639   }
2640   {
2641     \bool_if:NTF \l_@@_final_open_bool
2642     { \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim }
2643     {
```

Now the case where both extremities are closed. The first conditional tests whether the column is of type `c` (`C` of `{NiceArray}`) or may be considered as if.

```
2644       \dim_compare:nNnF \l_@@_x_initial_dim = \l_@@_x_final_dim
2645       {
2646         \dim_set:Nn \l_@@_x_initial_dim
```

```

2647         {
2648             \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
2649             \l_@@_x_initial_dim \l_@@_x_final_dim
2650         }
2651         \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim
2652     }
2653 }
2654 }
2655 \@@_draw_line:
2656 }

```

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.

```

2657 \cs_new_protected:Npn \@@_draw_Ddots:nnn #1 #2 #3
2658 {
2659     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2660     {
2661         \@@_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.

```

2662         \group_begin:
2663         \keys_set:nn { NiceMatrix / xdots } { #3 }
2664         \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2665         \@@_actually_draw_Ddots:
2666         \group_end:
2667     }
2668 }

```

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`.

```

2669 \cs_new_protected:Npn \@@_actually_draw_Ddots:
2670 {
2671     \bool_if:NTF \l_@@_initial_open_bool
2672     {
2673         \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2674         \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2675         \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2676         \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2677     }
2678     { \@@_set_initial_coords_from_anchor:n { south-east } }
2679     \bool_if:NTF \l_@@_final_open_bool
2680     {
2681         \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }
2682         \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2683         \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
2684         \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2685     }
2686     { \@@_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.

```
2687 \bool_if:NT \l_@@_parallelize_diags_bool
2688 {
2689   \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).

```
2690   \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 Δ_x and the Δ_y of the line because these values will be used to draw the others diagonal lines parallels to the first one.

```
2691   {
2692     \dim_gset:Nn \g_@@_delta_x_one_dim
2693     { \l_@@_x_final_dim - \l_@@_x_initial_dim }
2694     \dim_gset:Nn \g_@@_delta_y_one_dim
2695     { \l_@@_y_final_dim - \l_@@_y_initial_dim }
2696   }
```

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`.

```
2697   {
2698     \dim_set:Nn \l_@@_y_final_dim
2699     {
2700       \l_@@_y_initial_dim +
2701       ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2702       \dim_ratio:nn \g_@@_delta_y_one_dim \g_@@_delta_x_one_dim
2703     }
2704   }
2705 }
2706 \@@_draw_line:
2707 }
```

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.

```
2708 \cs_new_protected:Npn \@@_draw_Iddots:nnn #1 #2 #3
2709 {
2710   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2711   {
2712     \@@_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.

```
2713   \group_begin:
2714     \keys_set:nn { NiceMatrix / xdots } { #3 }
2715     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2716     \@@_actually_draw_Iddots:
2717   \group_end:
2718 }
2719 }
```

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`.

```

2720 \cs_new_protected:Npn \@@_actually_draw_iddots:
2721 {
2722   \bool_if:NTF \l_@@_initial_open_bool
2723   {
2724     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2725     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2726     \@@_qpoint:n { col - \@@_succ:n \l_@@_initial_j_int }
2727     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2728   }
2729   { \@@_set_initial_coords_from_anchor:n { south-west } }
2730   \bool_if:NTF \l_@@_final_open_bool
2731   {
2732     \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }
2733     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2734     \@@_qpoint:n { col - \int_use:N \l_@@_final_j_int }
2735     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2736   }
2737   { \@@_set_final_coords_from_anchor:n { north-east } }
2738   \bool_if:NT \l_@@_parallelize_diags_bool
2739   {
2740     \int_gincr:N \g_@@_iddots_int
2741     \int_compare:nNnTF \g_@@_iddots_int = 1
2742     {
2743       \dim_gset:Nn \g_@@_delta_x_two_dim
2744       { \l_@@_x_final_dim - \l_@@_x_initial_dim }
2745       \dim_gset:Nn \g_@@_delta_y_two_dim
2746       { \l_@@_y_final_dim - \l_@@_y_initial_dim }
2747     }
2748     {
2749       \dim_set:Nn \l_@@_y_final_dim
2750       {
2751         \l_@@_y_initial_dim +
2752         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2753         \dim_ratio:nn \g_@@_delta_y_two_dim \g_@@_delta_x_two_dim
2754       }
2755     }
2756   }
2757   \@@_draw_line:
2758 }

```

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`

```

2759 \cs_new_protected:Npn \@@_draw_line:
2760 {
2761   \pgfrememberpicturepositiononpagetrue
2762   \pgf@relevantforpicturesizefalse
2763   \tl_if_eq:NNTF \l_@@_xdots_line_style_tl \c_@@_standard_tl

```

```

2764 \@@_draw_standard_dotted_line:
2765 \@@_draw_non_standard_dotted_line:
2766 }

```

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.

```

2767 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:
2768 {
2769   \begin { scope }
2770   \exp_args:No \@@_draw_non_standard_dotted_line:n
2771   { \l_@@_xdots_line_style_tl , \l_@@_xdots_color_tl }
2772 }

```

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.

```

2773 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:n #1
2774 {
2775   \draw
2776   [
2777     #1 ,
2778     shorten-> = \l_@@_xdots_shorten_dim ,
2779     shorten-< = \l_@@_xdots_shorten_dim ,
2780   ]
2781   ( \l_@@_x_initial_dim , \l_@@_y_initial_dim )
2782   -- node [ sloped , above ]
2783     { \c_math_toggle_token \scriptstyle \l_@@_xdots_up_tl \c_math_toggle_token }
2784     node [ sloped , below ]
2785     {
2786       \c_math_toggle_token
2787       \scriptstyle \l_@@_xdots_down_tl
2788       \c_math_toggle_token
2789     }
2790   ( \l_@@_x_final_dim , \l_@@_y_final_dim ) ;
2791   \end { scope }
2792 }

```

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

```

2793 \cs_new_protected:Npn \@@_draw_standard_dotted_line:
2794 {

```

First, we put the labels.

```

2795   \bool_lazy_and:nnF
2796   { \tl_if_empty_p:N \l_@@_xdots_up_tl }
2797   { \tl_if_empty_p:N \l_@@_xdots_down_tl }
2798   {
2799     \pgfscope
2800     \pgftransformshift
2801     {
2802       \pgfpointlineattime { 0.5 }
2803       { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
2804       { \pgfpoint \l_@@_x_final_dim \l_@@_y_final_dim }
2805     }
2806     \pgftransformrotate
2807     {
2808       \fp_eval:n
2809       {
2810         atand
2811         (
2812           \l_@@_y_final_dim - \l_@@_y_initial_dim ,
2813           \l_@@_x_final_dim - \l_@@_x_initial_dim
2814         )

```

```

2815         }
2816     }
2817     \pgfnode
2818     { rectangle }
2819     { south }
2820     {
2821         \c_math_toggle_token
2822         \scriptstyle \l_@@_xdots_up_tl
2823         \c_math_toggle_token
2824     }
2825     { }
2826     { \pgfusepath { } }
2827     \pgfnode
2828     { rectangle }
2829     { north }
2830     {
2831         \c_math_toggle_token
2832         \scriptstyle \l_@@_xdots_down_tl
2833         \c_math_toggle_token
2834     }
2835     { }
2836     { \pgfusepath { } }
2837     \endpgfscope
2838 }
2839 \pgfrememberpicturepositiononpagetrue
2840 \pgf@relevantforpicturesizefalse
2841 \group_begin:

```

The dimension `\l_@@_l_dim` is the length ℓ of the line to draw. We use the floating point reals of `expl3` to compute this length.

```

2842     \dim_zero_new:N \l_@@_l_dim
2843     \dim_set:Nn \l_@@_l_dim
2844     {
2845         \fp_to_dim:n
2846         {
2847             sqrt
2848             (
2849                 ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) ^ 2
2850                 +
2851                 ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) ^ 2
2852             )
2853         }
2854     }

```

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.

```

2855     \bool_lazy_or:nnF
2856     { \dim_compare_p:nNn { \dim_abs:n \l_@@_l_dim } > \c_@@_max_l_dim }
2857     { \dim_compare_p:nNn \l_@@_l_dim = \c_zero_dim }
2858     \@@_draw_standard_dotted_line_i:
2859     \group_end:
2860 }
2861 \dim_const:Nn \c_@@_max_l_dim { 50 cm }
2862 \cs_new_protected:Npn \@@_draw_standard_dotted_line_i:
2863 {

```

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

```

2864     \bool_if:NTF \l_@@_initial_open_bool
2865     {
2866         \bool_if:NTF \l_@@_final_open_bool
2867         {

```

```

2868         \int_set:Nn \l_tmpa_int
2869         { \dim_ratio:nn \l_@@_l_dim \l_@@_inter_dots_dim }
2870     }
2871     {
2872         \int_set:Nn \l_tmpa_int
2873         {
2874             \dim_ratio:nn
2875             { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
2876             \l_@@_inter_dots_dim
2877         }
2878     }
2879 }
2880 {
2881     \bool_if:NTF \l_@@_final_open_bool
2882     {
2883         \int_set:Nn \l_tmpa_int
2884         {
2885             \dim_ratio:nn
2886             { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
2887             \l_@@_inter_dots_dim
2888         }
2889     }
2890     {
2891         \int_set:Nn \l_tmpa_int
2892         {
2893             \dim_ratio:nn
2894             { \l_@@_l_dim - 2 \l_@@_xdots_shorten_dim }
2895             \l_@@_inter_dots_dim
2896         }
2897     }
2898 }

```

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

```

2899     \dim_set:Nn \l_tmpa_dim
2900     {
2901         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2902         \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
2903     }
2904     \dim_set:Nn \l_tmpb_dim
2905     {
2906         ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
2907         \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
2908     }

```

The length ℓ is the length of the dotted line. We note Δ 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`.

```

2909     \int_set:Nn \l_tmpb_int
2910     {
2911         \bool_if:NTF \l_@@_initial_open_bool
2912         { \bool_if:NTF \l_@@_final_open_bool 1 0 }
2913         { \bool_if:NTF \l_@@_final_open_bool 2 1 }
2914     }

```

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.

```

2915     \dim_gadd:Nn \l_@@_x_initial_dim
2916     {
2917         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2918         \dim_ratio:nn
2919         { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
2920         { 2 \l_@@_l_dim }

```



```

2921     * \l_tmpb_int
2922   }
2923   \dim_gadd:Nn \l_@@_y_initial_dim
2924   {
2925     ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
2926     \dim_ratio:nn
2927     { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
2928     { 2 \l_@@_l_dim }
2929     * \l_tmpb_int
2930   }
2931   \pgf@relevantforpicturesizefalse
2932   \int_step_inline:nnn 0 \l_tmpa_int
2933   {
2934     \pgfpathcircle
2935     { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
2936     { \l_@@_radius_dim }
2937     \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
2938     \dim_add:Nn \l_@@_y_initial_dim \l_tmpb_dim
2939   }
2940   \pgfusepathqfill
2941 }

```

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.

```

2942 \AtBeginDocument
2943 {
2944   \tl_set:Nn \l_@@_argspec_tl { 0 { } E { _ ^ } { { } { } } }
2945   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
2946   \exp_args:NNV \NewDocumentCommand \@@_Ldots \l_@@_argspec_tl
2947   {
2948     \int_compare:nNnTF \c@jCol = 0
2949     { \@@_error:nn { in~first~col } \Ldots }
2950     {
2951       \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
2952       { \@@_error:nn { in~last~col } \Ldots }
2953       {
2954         \@@_instruction_of_type:nnn \c_false_bool { \Ldots }
2955         { #1 , down = #2 , up = #3 }
2956       }
2957     }
2958     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_ldots }
2959     \bool_gset_true:N \g_@@_empty_cell_bool
2960   }

2961   \exp_args:NNV \NewDocumentCommand \@@_Cdots \l_@@_argspec_tl
2962   {
2963     \int_compare:nNnTF \c@jCol = 0

```

```

2964     { \@@_error:nn { in~first~col } \Cdots }
2965     {
2966       \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
2967       { \@@_error:nn { in~last~col } \Cdots }
2968       {
2969         \@@_instruction_of_type:nnn \c_false_bool { Cdots }
2970         { #1 , down = #2 , up = #3 }
2971       }
2972     }
2973     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_cdots }
2974     \bool_gset_true:N \g_@@_empty_cell_bool
2975   }

\exp_args:NNV \NewDocumentCommand \@@_Vdots \l_@@_argspec_tl
{
  \int_compare:nNnTF \c@iRow = 0
  { \@@_error:nn { in~first~row } \Vdots }
  {
    \int_compare:nNnTF \c@iRow = \l_@@_last_row_int
    { \@@_error:nn { in~last~row } \Vdots }
    {
      \@@_instruction_of_type:nnn \c_false_bool { Vdots }
      { #1 , down = #2 , up = #3 }
    }
  }
  \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_vdots }
  \bool_gset_true:N \g_@@_empty_cell_bool
}

\exp_args:NNV \NewDocumentCommand \@@_Ddots \l_@@_argspec_tl
{
  \int_case:nnF \c@iRow
  {
    0 { \@@_error:nn { in~first~row } \Ddots }
    \l_@@_last_row_int { \@@_error:nn { in~last~row } \Ddots }
  }
  {
    \int_case:nnF \c@jCol
    {
      0 { \@@_error:nn { in~first~col } \Ddots }
      \l_@@_last_col_int { \@@_error:nn { in~last~col } \Ddots }
    }
    {
      \keys_set_known:nn { NiceMatrix / Ddots } { #1 }
      \@@_instruction_of_type:nnn \l_@@_draw_first_bool { Ddots }
      { #1 , down = #2 , up = #3 }
    }
  }
  \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_ddots }
  \bool_gset_true:N \g_@@_empty_cell_bool
}

\exp_args:NNV \NewDocumentCommand \@@_Iddots \l_@@_argspec_tl
{
  \int_case:nnF \c@iRow
  {
    0 { \@@_error:nn { in~first~row } \Iddots }
    \l_@@_last_row_int { \@@_error:nn { in~last~row } \Iddots }
  }
  {

```

```

3022 \int_case:nnF \c@jCol
3023 {
3024     0 { \@@_error:nn { in~first~col } \Iddots }
3025     \l_@@_last_col_int { \@@_error:nn { in~last~col } \Iddots }
3026 }
3027 {
3028     \keys_set_known:nn { NiceMatrix / Ddots } { #1 }
3029     \@@_instruction_of_type:nnn \l_@@_draw_first_bool { Iddots }
3030     { #1 , down = #2 , up = #3 }
3031 }
3032
3033 }
3034 \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_iddots }
3035 \bool_gset_true:N \g_@@_empty_cell_bool
3036 }
3037 }

```

End of the \AtBeginDocument.

Despite its name, the following set of keys will be used for \Ddots but also for \Iddots.

```

3038 \keys_define:nn { NiceMatrix / Ddots }
3039 {
3040     draw-first .bool_set:N = \l_@@_draw_first_bool ,
3041     draw-first .default:n = true ,
3042     draw-first .value_forbidden:n = true
3043 }

```

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

```

3044 \cs_new_protected:Npn \@@_Hspace:
3045 {
3046     \bool_gset_true:N \g_@@_empty_cell_bool
3047     \hspace
3048 }

```

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

```

3049 \cs_set_eq:NN \@@_old_multicolumn \multicolumn
3050 \cs_new:Npn \@@_multicolumn:nnn #1 #2 #3
3051 {
3052     % \begin{macrocode}
3053     % We have to act in an expandable way since it will begin by a |\multicolumn|.
3054     % \end{macrocode}
3055     \exp_args:NNe
3056     \@@_old_multicolumn
3057     { #1 }

```

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

```

3058 {
3059     \exp_args:Ne \str_case:nn { \tl_lower_case:n { #2 } }
3060     {
3061         l { > \@@_Cell: l < \@@_end_Cell: }
3062         r { > \@@_Cell: r < \@@_end_Cell: }
3063         c { > \@@_Cell: c < \@@_end_Cell: }
3064         { l | } { > \@@_Cell: l < \@@_end_Cell: | }
3065         { r | } { > \@@_Cell: r < \@@_end_Cell: | }
3066         { c | } { > \@@_Cell: c < \@@_end_Cell: | }
3067         { | l } { | > \@@_Cell: l < \@@_end_Cell: }
3068         { | r } { | > \@@_Cell: r < \@@_end_Cell: }
3069         { | c } { | > \@@_Cell: c < \@@_end_Cell: }
3070         { | l | } { | > \@@_Cell: l < \@@_end_Cell: | }
3071         { | r | } { | > \@@_Cell: r < \@@_end_Cell: | }
3072         { | c | } { | > \@@_Cell: c < \@@_end_Cell: | }
3073     }

```

```

3074     }
3075     { #3 }
The \peek_remove_spaces:n is mandatory.
3076     \peek_remove_spaces:n
3077     {
3078         \int_compare:nNnT #1 > 1
3079         {
3080             \seq_gput_left:Nx \g_@@_multicolumn_cells_seq
3081             { \int_use:N \c@iRow - \int_use:N \c@jCol }
3082             \seq_gput_left:Nn \g_@@_multicolumn_sizes_seq { #1 }
3083             \seq_gput_right:Nx \g_@@_pos_of_blocks_seq
3084             {
3085                 { \int_use:N \c@iRow }
3086                 { \int_use:N \c@jCol }
3087                 { \int_use:N \c@iRow }
3088                 { \int_eval:n { \c@jCol + #1 - 1 } }
3089             }
3090         }
3091         \int_gadd:Nn \c@jCol { #1 - 1 }
3092         \int_compare:nNnT \c@jCol > \g_@@_col_total_int
3093         { \int_gset_eq:NN \g_@@_col_total_int \c@jCol }
3094     }
3095 }

```

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`.

```

3096 \cs_new:Npn \@@_Hdotsfor:
3097 {
3098     \int_compare:nNnTF \c@jCol = 0
3099     { \@@_error:n { Hdotsfor~in~col~0 } }
3100     {
3101         \multicolumn { 1 } { c } { }
3102         \@@_Hdotsfor_i
3103     }
3104 }

```

The command `\@@_Hdotsfor_i` is defined with the tools of `xparse` because it has an optional argument. Note that such a command defined by `\NewDocumentCommand` is protected and that's why we have put the `\multicolumn` before (in the definition of `\@@_Hdotsfor:`).

```

3105 \AtBeginDocument
3106 {
3107     \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
3108     \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.

```

3109     \exp_args:NNV \NewDocumentCommand \@@_Hdotsfor_i \l_@@_argspec_tl
3110     {
3111         \tl_gput_right:Nx \g_@@_HVDotsfor_lines_tl
3112         {
3113             \@@_Hdotsfor:nnnn
3114             { \int_use:N \c@iRow }
3115             { \int_use:N \c@jCol }
3116             { #2 }
3117             {
3118                 #1 , #3 ,
3119                 down = \exp_not:n { #4 } , up = \exp_not:n { #5 }
3120             }
3121         }
3122         \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { c } { } }
3123     }
3124 }

```

Enf of \AtBeginDocument.

```

3125 \cs_new_protected:Npn \@@_Hdotsfor:nnnn #1 #2 #3 #4
3126 {
3127   \bool_set_false:N \l_@@_initial_open_bool
3128   \bool_set_false:N \l_@@_final_open_bool

```

For the row, it's easy.

```

3129   \int_set:Nn \l_@@_initial_i_int { #1 }
3130   \int_set_eq:NN \l_@@_final_i_int \l_@@_initial_i_int

```

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

```

3131   \int_compare:nNnTF #2 = 1
3132   {
3133     \int_set:Nn \l_@@_initial_j_int 1
3134     \bool_set_true:N \l_@@_initial_open_bool
3135   }
3136   {
3137     \cs_if_exist:cTF
3138     {
3139       pgf @ sh @ ns @ \@@_env:
3140       - \int_use:N \l_@@_initial_i_int
3141       - \int_eval:n { #2 - 1 }
3142     }
3143     { \int_set:Nn \l_@@_initial_j_int { #2 - 1 } }
3144     {
3145       \int_set:Nn \l_@@_initial_j_int { #2 }
3146       \bool_set_true:N \l_@@_initial_open_bool
3147     }
3148   }
3149   \int_compare:nNnTF { #2 + #3 - 1 } = \c@jCol
3150   {
3151     \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
3152     \bool_set_true:N \l_@@_final_open_bool
3153   }
3154   {
3155     \cs_if_exist:cTF
3156     {
3157       pgf @ sh @ ns @ \@@_env:
3158       - \int_use:N \l_@@_final_i_int
3159       - \int_eval:n { #2 + #3 }
3160     }
3161     { \int_set:Nn \l_@@_final_j_int { #2 + #3 } }
3162     {
3163       \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
3164       \bool_set_true:N \l_@@_final_open_bool
3165     }
3166   }
3167   \group_begin:
3168   \int_compare:nNnTF { #1 } = 0
3169   { \color { nicematrix-first-row } }
3170   {
3171     \int_compare:nNnT { #1 } = \g_@@_row_total_int
3172     { \color { nicematrix-last-row } }
3173   }
3174   \keys_set:nn { NiceMatrix / xdots } { #4 }
3175   \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3176   \@@_actually_draw_Ldots:
3177   \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).

```

3178 \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
3179 { \cs_set:cpn { @@ _ dotted _ #1 - ##1 } { } }
3180 }

3181 \AtBeginDocument
3182 {
3183 \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
3184 \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
3185 \exp_args:NNV \NewDocumentCommand \@@_Vdotsfor: \l_@@_argspec_tl
3186 {
3187 \tl_gput_right:Nx \g_@@_HVdotsfor_lines_tl
3188 {
3189 \@@_Vdotsfor:nnnn
3190 { \int_use:N \c@iRow }
3191 { \int_use:N \c@jCol }
3192 { #2 }
3193 {
3194 #1 , #3 ,
3195 down = \exp_not:n { #4 } , up = \exp_not:n { #5 }
3196 }
3197 }
3198 }
3199 }

```

Enf of \AtBeginDocument.

```

3200 \cs_new_protected:Npn \@@_Vdotsfor:nnnn #1 #2 #3 #4
3201 {
3202 \bool_set_false:N \l_@@_initial_open_bool
3203 \bool_set_false:N \l_@@_final_open_bool

```

For the column, it's easy.

```

3204 \int_set:Nn \l_@@_initial_j_int { #2 }
3205 \int_set_eq:NN \l_@@_final_j_int \l_@@_initial_j_int

```

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

```

3206 \int_compare:nNnTF #1 = 1
3207 {
3208 \int_set:Nn \l_@@_initial_i_int 1
3209 \bool_set_true:N \l_@@_initial_open_bool
3210 }
3211 {
3212 \cs_if_exist:cTF
3213 {
3214 pgf @ sh @ ns @ \@@_env:
3215 - \int_eval:n { #1 - 1 }
3216 - \int_use:N \l_@@_initial_j_int
3217 }
3218 { \int_set:Nn \l_@@_initial_i_int { #1 - 1 } }
3219 {
3220 \int_set:Nn \l_@@_initial_i_int { #1 }
3221 \bool_set_true:N \l_@@_initial_open_bool
3222 }
3223 }
3224 \int_compare:nNnTF { #1 + #3 - 1 } = \c@iRow
3225 {
3226 \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }
3227 \bool_set_true:N \l_@@_final_open_bool
3228 }
3229 {
3230 \cs_if_exist:cTF
3231 {
3232 pgf @ sh @ ns @ \@@_env:
3233 - \int_eval:n { #1 + #3 }
3234 - \int_use:N \l_@@_final_j_int

```

```

3235     }
3236     { \int_set:Nn \l_@@_final_i_int { #1 + #3 } }
3237     {
3238         \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }
3239         \bool_set_true:N \l_@@_final_open_bool
3240     }
3241 }
3242 \group_begin:
3243 \int_compare:nNnTF { #2 } = 0
3244 { \color { nicematrix-first-col } }
3245 {
3246     \int_compare:nNnT { #2 } = \g_@@_col_total_int
3247     { \color { nicematrix-last-col } }
3248 }
3249 \keys_set:nn { NiceMatrix / xdots } { #4 }
3250 \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3251 \@@_actually_draw_Vdots:
3252 \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).

```

3253     \int_step_inline:nnn { #1 } { #1 + #3 - 1 }
3254     { \cs_set:cpn { @@ _ dotted _ ##1 - #2 } { } }
3255 }

```

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

```

3256 \cs_new_protected:Npn \@@_rotate: { \bool_gset_true:N \g_@@_rotate_bool }

```

The command `\line` accessible in code-after

In the `code-after`, the command `\@@_line:nn` will be linked to `\line`. This command takes two arguments which are the specifications of two cells in the array (in the format *i-j*) and draws a dotted line between these cells.

First, we write a command with an argument of the format *i-j* and applies the command `\int_eval:n` to *i* and *j* ; this must *not* be protected (and is, of course fully expandable).⁴²

```

3257 \cs_new:Npn \@@_double_int_eval:n #1-#2 \q_stop
3258 { \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).

```

3259 \AtBeginDocument
3260 {
3261     \tl_set:Nn \l_@@_argspec_tl { 0 { } m m ! 0 { } E { _ ^ } { { } { } } }
3262     \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
3263     \exp_args:NNV \NewDocumentCommand \@@_line \l_@@_argspec_tl
3264     {
3265         \group_begin:
3266         \keys_set:nn { NiceMatrix / xdots } { #1 , #4 , down = #5 , up = #6 }
3267         \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3268         \use:e
3269         {
3270             \@@_line_i:nn

```

⁴²Indeed, we want that the user may use the command `\line` in `code-after` with LaTeX counters in the arguments — with the command `\value`.

```

3271         { \@@_double_int_eval:n #2 \q_stop }
3272         { \@@_double_int_eval:n #3 \q_stop }
3273     }
3274     \group_end:
3275 }
3276 }
3277 \cs_new_protected:Npn \@@_line_i:nn #1 #2
3278 {
3279     \bool_set_false:N \l_@@_initial_open_bool
3280     \bool_set_false:N \l_@@_final_open_bool
3281     \bool_if:nTF
3282     {
3283         \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #1 }
3284         ||
3285         \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #2 }
3286     }
3287     {
3288         \@@_error:nnn { unknown~cell~for~line~in~code~after } { #1 } { #2 }
3289     }
3290     { \@@_draw_line_ii:nn { #1 } { #2 } }
3291 }
3292 \AtBeginDocument
3293 {
3294     \cs_new_protected:Npx \@@_draw_line_ii:nn #1 #2
3295     {

```

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:`.

```

3296     \c_@@_pgfortikzpicture_tl
3297     \@@_draw_line_iii:nn { #1 } { #2 }
3298     \c_@@_endpgfortikzpicture_tl
3299 }
3300 }

```

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

```

3301 \cs_new_protected:Npn \@@_draw_line_iii:nn #1 #2
3302 {
3303     \pgfrememberpicturepositiononpagetrue
3304     \pgfpointshapeborder { \@@_env: - #1 } { \@@_qpoint:n { #2 } }
3305     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
3306     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
3307     \pgfpointshapeborder { \@@_env: - #2 } { \@@_qpoint:n { #1 } }
3308     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
3309     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
3310     \@@_draw_line:
3311 }

```

The commands `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, and `\Iddots` don’t use this command because they have to do other settings (for example, the diagonal lines must be parallelized).

Colors of cells, rows and columns

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

```

3312 \cs_set_protected:Npn \@@_cut_on_hyphen:w #1-#2\q_stop
3313 {
3314     \tl_set:Nn \l_tmpa_tl { #1 }
3315     \tl_set:Nn \l_tmpb_tl { #2 }
3316 }

```


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

```

3317 \NewDocumentCommand \@@_rowcolor { 0 { } m m }
3318 {
3319   \tl_if_blank:nF { #2 }
3320   {
3321     \pgfpicture
3322     \pgf@relevantforpicturesizefalse
3323     \tl_if_empty:nTF { #1 } \color { \color [ #1 ] } { #2 }

```

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

```

3324   \@@_qpoint:n { col - 1 }
3325   \int_compare:nNnTF \l_@@_first_col_int = 0
3326   { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
3327   { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
3328   \@@_qpoint:n { col - \@@_succ:n \c@jCol }
3329   \dim_set:Nn \l_tmpa_dim { \pgf@x + 0.5 \arrayrulewidth }
3330   \clist_map_inline:nn { #3 }
3331   {
3332     \tl_set:Nn \l_tmpa_tl { ##1 }
3333     \tl_if_in:NnTF \l_tmpa_tl { - }
3334     { \@@_cut_on_hyphen:w ##1 \q_stop }
3335     { \@@_cut_on_hyphen:w ##1 - ##1 \q_stop }
3336     \tl_if_empty:NT \l_tmpa_tl { \tl_set:Nn \l_tmpa_tl { 1 } }
3337     \tl_if_empty:NT \l_tmpb_tl
3338     { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@iRow } }
3339     \int_compare:nNnT \l_tmpb_tl > \c@iRow
3340     { \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.

```

3341     \@@_qpoint:n { row - \@@_succ:n \l_tmpb_tl }
3342     \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
3343     \@@_qpoint:n { row - \l_tmpa_tl }
3344     \dim_set:Nn \l_tmpd_dim { \pgf@y + 0.5 \arrayrulewidth }
3345     \pgfpathrectanglecorners
3346     { \pgfpoint \l_tmpc_dim \l_tmpd_dim }
3347     { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3348   }
3349   \pgfusepathqfill
3350   \endpgfpicture
3351 }
3352 }

```

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

```

3353 \NewDocumentCommand \@@_columncolor { 0 { } m m }
3354 {
3355   \tl_if_blank:nF { #2 }
3356   {
3357     \pgfpicture
3358     \pgf@relevantforpicturesizefalse
3359     \tl_if_empty:nTF { #1 } \color { \color [ #1 ] } { #2 }
3360     \@@_qpoint:n { row - 1 }

```

\l_tmpa_dim is the y -value of the top of the columns et \l_tmpb_dim is the y -value of the bottom.

```

3361     \dim_set:Nn \l_tmpa_dim { \pgf@y + 0.5 \arrayrulewidth }
3362     \@@_qpoint:n { row - \@@_succ:n \c@iRow }
3363     \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
3364     \clist_map_inline:nn { #3 }
3365     {
3366       \tl_set:Nn \l_tmpa_tl { ##1 }
3367       \tl_if_in:NnTF \l_tmpa_tl { - }
3368       { \@@_cut_on_hyphen:w ##1 \q_stop }
3369       { \@@_cut_on_hyphen:w ##1 - ##1 \q_stop }
3370       \tl_if_empty:NT \l_tmpa_tl { \tl_set:Nn \l_tmpa_tl { 1 } }
3371       \tl_if_empty:NT \l_tmpb_tl

```

```

3372         { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
3373     \int_compare:nNnT \l_tmpb_tl > \c@jCol
3374         { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }

```

Now, the numbers of both columns are in \l_tmpa_tl and \l_tmpb_tl.

```

3375     \@@_qpoint:n { col - \l_tmpa_tl }
3376     \int_compare:nNnTF \l_@@_first_col_int = \l_tmpa_tl
3377         { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
3378         { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
3379     \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
3380     \dim_set:Nn \l_tmpd_dim { \pgf@x + 0.5 \arrayrulewidth }
3381     \pgfpathrectanglecorners
3382         { \pgfpoint \l_tmpc_dim \l_tmpa_dim }
3383         { \pgfpoint \l_tmpd_dim \l_tmpb_dim }
3384     }
3385     \pgfusepathqfill
3386     \endpgfpicture
3387 }
3388 }

```

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

```

3389 \NewDocumentCommand \@@_cellcolor { 0 { } m m }
3390 {
3391     \tl_if_blank:nF { #2 }
3392     {
3393         \pgfpicture
3394         \pgf@relevantforpicturesizefalse
3395         \tl_if_empty:nTF { #1 } \color { \color [ #1 ] } { #2 }
3396         \clist_map_inline:nn { #3 }
3397         {
3398             \@@_cut_on_hyphen:w ##1 \q_stop
3399             \@@_qpoint:n { row - \l_tmpa_tl }
3400             \bool_lazy_and:nnT
3401                 { \int_compare_p:n { \l_tmpa_tl <= \c@iRow } }
3402                 { \int_compare_p:n { \l_tmpb_tl <= \c@jCol } }
3403             {
3404                 \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
3405                 \@@_qpoint:n { row - \@@_succ:n \l_tmpa_tl }
3406                 \dim_set:Nn \l_tmpa_dim { \pgf@y + 0.5 \arrayrulewidth }
3407                 \@@_qpoint:n { col - \l_tmpb_tl }
3408                 \int_compare:nNnTF \l_@@_first_col_int = \l_tmpb_tl
3409                     { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
3410                     { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
3411                 \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
3412                 \dim_set:Nn \l_tmpd_dim { \pgf@x + 0.5 \arrayrulewidth }
3413                 \pgfpathrectanglecorners
3414                     { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
3415                     { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
3416             }
3417         }
3418         \pgfusepathqfill
3419         \endpgfpicture
3420     }
3421 }

```

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

```

3422 \NewDocumentCommand \@@_rectanglecolor { 0 { } m m m }
3423 {
3424     \tl_if_blank:nF { #2 }
3425     {
3426         \pgfpicture
3427         \pgf@relevantforpicturesizefalse
3428         \tl_if_empty:nTF { #1 } \color { \color [ #1 ] } { #2 }

```

```

3429 \@@_cut_on_hyphen:w #3 \q_stop
3430 \bool_lazy_and:nnT
3431 { \int_compare_p:n { \l_tmpa_tl <= \c@iRow } }
3432 { \int_compare_p:n { \l_tmpb_tl <= \c@jCol } }
3433 {
3434 \@@_qpoint:n { row - \l_tmpa_tl }
3435 \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
3436 \@@_qpoint:n { col - \l_tmpb_tl }
3437 \int_compare:nNnTF \l_@@_first_col_int = \l_tmpb_tl
3438 { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
3439 { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
3440 \@@_cut_on_hyphen:w #4 \q_stop
3441 \int_compare:nNnT \l_tmpa_tl > \c@iRow
3442 { \tl_set:Nx \l_tmpa_tl { \int_use:N \c@iRow } }
3443 \int_compare:nNnT \l_tmpb_tl > \c@jCol
3444 { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
3445 \@@_qpoint:n { row - \@@_succ:n \l_tmpa_tl }
3446 \dim_set:Nn \l_tmpa_dim { \pgf@y + 0.5 \arrayrulewidth }
3447 \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
3448 \dim_set:Nn \l_tmpd_dim { \pgf@x + 0.5 \arrayrulewidth }
3449 \pgfpathrectanglecorners
3450 { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
3451 { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
3452 \pgfusepathqfill
3453 }
3454 \endpgfpicture
3455 }
3456 }

```

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]`.

The last optional argument is for options. As of now, there is only one key available : `respect-blocks`.

```

3457 \keys_define:nn { NiceMatrix / rowcolors }
3458 {
3459   respect-blocks .bool_set:N = \l_@@_respect_blocks_bool ,
3460   respect-blocks .default:n = true ,
3461   unknown .code:n = \@@_error:n { Unknown-option-for-rowcolors }
3462 }
3463 \NewDocumentCommand \@@_rowcolors { 0 { } m m m 0 { } }
3464 {
3465   \keys_set:nn { NiceMatrix / rowcolors } { #5 }
3466   \bool_lazy_and:nnTF
3467     \l_@@_respect_blocks_bool
3468     { \cs_if_exist_p:c { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq } }
3469     { \@@_rowcolors_i:nnnn { #1 } { #2 } { #3 } { #4 } }
3470     {
3471       \int_step_inline:nnn { #2 } { \int_use:N \c@iRow }
3472       {
3473         \int_if_odd:nTF { ##1 }
3474         { \@@_rowcolor [ #1 ] { #3 } }
3475         { \@@_rowcolor [ #1 ] { #4 } }
3476       } { ##1 }
3477     }
3478   }
3479 }
3480 \cs_new_protected:Npn \@@_rowcolors_i:nnnn #1 #2 #3 #4
3481 {
3482   \seq_set_eq:Nc \l_tmpb_seq
3483   { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq }

```

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".

```
3484 \seq_set_filter:NNn \l_tmpa_seq \l_tmpb_seq
3485 { \@@_not_in_exterior_p:nnnn ##1 }
```

The counter `\l_tmpa_int` will be the index of the loop.

```
3486 \int_set:Nn \l_tmpa_int { #2 }
```

The boolean `\l_tmpa_bool` will indicate whereas we are in a row of the first color or of the second color.

```
3487 \bool_set_false:N \l_tmpa_bool
```

We recall that, in the code-before, `\c@iRow` is the total number of rows of the array (excepted the potential exterior rows).

```
3488 \int_do_until:nNnn \l_tmpa_int > \c@iRow
3489 {
3490   \seq_set_filter:NNn \l_tmpb_seq \l_tmpa_seq
3491   { \@@_intersect_our_row_p:nnnn ##1 }
```

We compute in `\l_tmpb_int` the last row covered by a block.

```
3492   \int_set_eq:NN \l_tmpb_int \l_tmpa_int
3493   \seq_map_inline:Nn \l_tmpb_seq { \@@_rowcolors_ii:nnnn ##1 }
3494   \bool_if:NTF \l_tmpa_bool
3495   {
3496     \@@_rowcolor [ #1 ] { #4 }
3497     { \int_use:N \l_tmpa_int - \int_use:N \l_tmpb_int }
3498     \bool_set_false:N \l_tmpa_bool
3499   }
3500   {
3501     \@@_rowcolor [ #1 ] { #3 }
3502     { \int_use:N \l_tmpa_int - \int_use:N \l_tmpb_int }
3503     \bool_set_true:N \l_tmpa_bool
3504   }
3505   \int_set:Nn \l_tmpa_int { \l_tmpb_int + 1 }
3506 }
3507 }
```

```
3508 \cs_new_protected:Npn \@@_rowcolors_ii:nnnn #1 #2 #3 #4
3509 {
3510   \int_compare:nNnT { #3 } > \l_tmpb_int
3511   { \int_set:Nn \l_tmpb_int { #3 } }
3512 }
```

```
3513 \prg_new_conditional:Nnn \@@_not_in_exterior:nnnn p
3514 {
3515   \bool_lazy_or:nnTF
3516   { \int_compare_p:nNn { #4 } = \c_zero_int }
3517   { \int_compare_p:nNn { #2 } = { \@@_succ:n { \c@jCol } } } }
3518   \prg_return_false:
3519   \prg_return_true:
3520 }
```

The following command return true when the block intersects the row `\l_tmpa_int`.

```
3521 \prg_new_conditional:Nnn \@@_intersect_our_row:nnnn p
3522 {
3523   \bool_if:nTF
3524   {
3525     \int_compare_p:n { #1 <= \l_tmpa_int }
3526     &&
3527     \int_compare_p:n { \l_tmpa_int <= #3 }
3528   }
3529   \prg_return_true:
3530   \prg_return_false:
3531 }
```

```

3532 \NewDocumentCommand \@@_chessboardcolors { 0 { } m m }
3533 {
3534   \int_step_inline:nn { \int_use:N \c@iRow }
3535   {
3536     \int_step_inline:nn { \int_use:N \c@jCol }
3537     {
3538       \int_if_even:nTF { ####1 + ##1 }
3539       { \@@_cellcolor [ #1 ] { #2 } }
3540       { \@@_cellcolor [ #1 ] { #3 } }
3541       { ##1 - ####1 }
3542     }
3543   }
3544 }

```

When the user uses the key `colortbl`-like, the following command will be linked to `\cellcolor` in the tabular.

```

3545 \NewDocumentCommand \@@_cellcolor_tabular { 0 { } m }
3546 {
3547   \tl_gput_right:Nx \g_nicematrix_code_before_tl
3548   { \cellcolor [ #1 ] { #2 } { \int_use:N \c@iRow - \int_use:N \c@jCol } }
3549 }

```

When the user uses the key `rowcolor-in-tabular`, the following command will be linked to `\rowcolor` in the tabular.

```

3550 \NewDocumentCommand \@@_rowcolor_tabular { 0 { } m }
3551 {
3552   \tl_gput_right:Nx \g_nicematrix_code_before_tl
3553   {
3554     \exp_not:N \rectanglecolor [ #1 ] { #2 }
3555     { \int_use:N \c@iRow - \int_use:N \c@jCol }
3556     { \int_use:N \c@iRow - \exp_not:n { \int_use:N \c@jCol } }
3557   }
3558 }

```

```

3559 \NewDocumentCommand \@@_columncolor_preamble { 0 { } m }
3560 {
3561   \int_compare:nNnT \c@iRow = 1
3562   {

```

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).

```

3563     \tl_gput_left:Nx \g_nicematrix_code_before_tl
3564     { \exp_not:N \columncolor [ #1 ] { #2 } { \int_use:N \c@jCol } }
3565   }
3566 }

```

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`.

```

3567 \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`.

```

3568 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix:n #1
3569 {
3570   \int_compare:nNnTF \l_@@_first_col_int = 0
3571   { \@@_OnlyMainNiceMatrix_i:n { #1 } }
3572   {
3573     \int_compare:nNnTF \c@jCol = 0
3574     {
3575       \int_compare:nNnF \c@iRow = { -1 }
3576       { \int_compare:nNnF \c@iRow = { \l_@@_last_row_int - 1 } { #1 } }
3577     }
3578     { \@@_OnlyMainNiceMatrix_i:n { #1 } }
3579   }
3580 }

```

This definition may seem complicated by we must remind that the number of row `\c@iRow` is incremented in the first cell of the row, *after* an potential vertical rule on the left side of the first cell.

The command `\@@_OnlyMainNiceMatrix_i:n` is only a short-cut which is used twice in the above command. This command must *not* be protected.

```

3581 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix_i:n #1
3582 {
3583   \int_compare:nNnF \c@iRow = 0
3584   { \int_compare:nNnF \c@iRow = \l_@@_last_row_int { #1 } }
3585 }

```

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`. `#2` is the number of consecutive occurrences of `|`.

```

3586 \cs_new_protected:Npn \@@_vline:nn #1 #2
3587 {

```

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).

```

3588   \int_compare:nNnT { #1 } < { \c@jCol + 2 }
3589   {
3590     \pgfpicture
3591     \@@_vline_i:nn { #1 } { #2 }
3592     \endpgfpicture
3593   }
3594 }
3595 \cs_new_protected:Npn \@@_vline_i:nn #1 #2
3596 {

```

`\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`.

```

3597   \tl_set:Nx \l_tmpb_tl { #1 }
3598   \tl_clear_new:N \l_tmpc_tl
3599   \int_step_variable:nNn \c@iRow \l_tmpa_tl
3600   {

```

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.

```

3601     \bool_gset_true:N \g_tmpa_bool
3602     \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
3603     { \@@_test_if_vline_in_block:nnnn ##1 }
3604     \seq_map_inline:Nn \g_@@_pos_of_xdots_seq
3605     { \@@_test_if_vline_in_block:nnnn ##1 }

```

```

3606 \clist_if_empty:NF \l_@@_except_corners_clist
3607 \@@_test_in_corner_v:
3608 \bool_if:NTF \g_tmpa_bool
3609 {
3610     \tl_if_empty:NT \l_tmpc_tl
We keep in memory that we have a rule to draw.
3611     { \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl }
3612 }
3613 {
3614     \tl_if_empty:NF \l_tmpc_tl
3615     {
3616         \@@_vline_ii:nnnn
3617         { #1 }
3618         { #2 }
3619         \l_tmpc_tl
3620         { \int_eval:n { \l_tmpa_tl - 1 } }
3621         \tl_clear:N \l_tmpc_tl
3622     }
3623 }
3624 }
3625 \tl_if_empty:NF \l_tmpc_tl
3626 {
3627     \@@_vline_ii:nnnn
3628     { #1 }
3629     { #2 }
3630     \l_tmpc_tl
3631     { \int_use:N \c@iRow }
3632     \tl_clear:N \l_tmpc_tl
3633 }
3634 }

3635 \cs_new_protected:Npn \@@_test_in_corner_v:
3636 {
3637     \int_compare:nNnTF \l_tmpb_tl = { \@@_succ:n \c@jCol }
3638     {
3639         \seq_if_in:NxT
3640         \l_@@_empty_corner_cells_seq
3641         { \l_tmpa_tl - \@@_pred:n \l_tmpb_tl }
3642         { \bool_set_false:N \g_tmpa_bool }
3643     }
3644     {
3645         \seq_if_in:NxT
3646         \l_@@_empty_corner_cells_seq
3647         { \l_tmpa_tl - \l_tmpb_tl }
3648         {
3649             \int_compare:nNnTF \l_tmpb_tl = 1
3650             { \bool_set_false:N \g_tmpa_bool }
3651             {
3652                 \seq_if_in:NxT
3653                 \l_@@_empty_corner_cells_seq
3654                 { \l_tmpa_tl - \@@_pred:n \l_tmpb_tl }
3655                 { \bool_set_false:N \g_tmpa_bool }
3656             }
3657         }
3658     }
3659 }

```

#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 number of the rows between which the rule has to be drawn.

```

3660 \cs_new_protected:Npn \@@_vline_ii:nnnn #1 #2 #3 #4
3661 {

```

```

3662 \pgfrememberpicturepositiononpagetrue
3663 \pgf@relevantforpicturesizefalse
3664 \@@_qpoint:n { row - #3 }
3665 \dim_set_eq:NN \l_tmpa_dim \pgf@y
3666 \@@_qpoint:n { col - #1 }
3667 \dim_set_eq:NN \l_tmpb_dim \pgf@x
3668 \@@_qpoint:n { row - \@@_succ:n { #4 } }
3669 \dim_set_eq:NN \l_tmpc_dim \pgf@y
3670 \bool_lazy_and:nnT
3671   { \int_compare_p:nNn { #2 } > 1 }
3672   { ! \tl_if_blank_p:V \CT@drsc@ }
3673   {
3674     \group_begin:
3675     \CT@drsc@
3676     \dim_add:Nn \l_tmpa_dim { 0.5 \arrayrulewidth }
3677     \dim_sub:Nn \l_tmpc_dim { 0.5 \arrayrulewidth }
3678     \dim_set:Nn \l_tmpd_dim
3679       { \l_tmpb_dim - ( \doublerulesep + \arrayrulewidth ) * ( #2 - 1 ) }
3680     \pgfpathrectanglecorners
3681       { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
3682       { \pgfpoint \l_tmpd_dim \l_tmpc_dim }
3683     \pgfusepathqfill
3684     \group_end:
3685   }
3686 \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
3687 \pgfpathlineto { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
3688 \prg_replicate:nn { #2 - 1 }
3689 {
3690   \dim_sub:Nn \l_tmpb_dim \arrayrulewidth
3691   \dim_sub:Nn \l_tmpb_dim \doublerulesep
3692   \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
3693   \pgfpathlineto { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
3694 }
3695 \CT@arc@
3696 \pgfsetlinewidth { 1.1 \arrayrulewidth }
3697 \pgfsetrectcap
3698 \pgfusepathqstroke
3699 }

```

The following 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 `except-corners` is not used).

```

3700 \cs_new_protected:Npn \@@_vline_i_complete:nn #1 #2
3701   { \@@_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 `except-corners` is used).

```

3702 \cs_new_protected:Npn \@@_draw_vlines:
3703   {
3704     \int_step_inline:nnn
3705       { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
3706       { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@jCol } \c@jCol }
3707       { \@@_vline:nn { ##1 } 1 }
3708   }

```

The horizontal rules

The following command will be executed in the `internal-code-after`. The row will be drawn *before* the row #1. #2 is the number of consecutive occurrences of `\Hline`.


```

3709 \cs_new_protected:Npn \@@_hline:nn #1 #2
3710 {
3711   \pgfpicture
3712   \@@_hline_i:nn { #1 } { #2 }
3713   \endpgfpicture
3714 }
3715 \cs_new_protected:Npn \@@_hline_i:nn #1 #2
3716 {

```

\l_tmpa_tl is the number of row and \l_tmpb_tl the number of column. Whe, we have found a column corresponding to a rule to draw, we note its numver in \l_tmpc_tl.

```

3717   \tl_set:Nn \l_tmpa_tl { #1 }
3718   \tl_clear_new:N \l_tmpc_tl
3719   \int_step_variable:nNn \c@jCol \l_tmpb_tl
3720   {

```

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.

```

3721     \bool_gset_true:N \g_tmpa_bool
3722     \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
3723     { \@@_test_if_hline_in_block:nnnn ##1 }
3724     \seq_map_inline:Nn \g_@@_pos_of_xdots_seq
3725     { \@@_test_if_hline_in_block:nnnn ##1 }
3726     \clist_if_empty:NF \l_@@_except_corners_clist \@@_test_in_corner_h:
3727     \bool_if:NTF \g_tmpa_bool
3728     {
3729       \tl_if_empty:NT \l_tmpc_tl

```

We keep in memory that we have a rule to draw.

```

3730       { \tl_set_eq:NN \l_tmpc_tl \l_tmpb_tl }
3731     }
3732   {
3733     \tl_if_empty:NF \l_tmpc_tl
3734     {
3735       \@@_hline_ii:nnnn
3736       { #1 }
3737       { #2 }
3738       \l_tmpc_tl
3739       { \int_eval:n { \l_tmpb_tl - 1 } }
3740       \tl_clear:N \l_tmpc_tl
3741     }
3742   }
3743 }
3744 \tl_if_empty:NF \l_tmpc_tl
3745 {
3746   \@@_hline_ii:nnnn
3747   { #1 }
3748   { #2 }
3749   \l_tmpc_tl
3750   { \int_use:N \c@jCol }
3751   \tl_clear:N \l_tmpc_tl
3752 }
3753 }

```

```

3754 \cs_new_protected:Npn \@@_test_in_corner_h:
3755 {
3756   \int_compare:nNnTF \l_tmpa_tl = { \@@_succ:n \c@iRow }
3757   {
3758     \seq_if_in:NxT
3759     \l_@@_empty_corner_cells_seq
3760     { \@@_pred:n \l_tmpa_tl - \l_tmpb_tl }

```

```

3761         { \bool_set_false:N \g_tmpa_bool }
3762     }
3763     {
3764         \seq_if_in:NxT
3765         \l_@@_empty_corner_cells_seq
3766         { \l_tmpa_tl - \l_tmpb_tl }
3767         {
3768             \int_compare:nNnTF \l_tmpa_tl = 1
3769             { \bool_set_false:N \g_tmpa_bool }
3770             {
3771                 \seq_if_in:NxT
3772                 \l_@@_empty_corner_cells_seq
3773                 { \@@_pred:n \l_tmpa_tl - \l_tmpb_tl }
3774                 { \bool_set_false:N \g_tmpa_bool }
3775             }
3776         }
3777     }
3778 }

```

#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.

```

3779 \cs_new_protected:Npn \@@_hline_ii:nnnn #1 #2 #3 #4
3780 {
3781     \pgfrememberpicturerepositiononpagetrue
3782     \pgf@relevantforpicturesizefalse
3783     \@@_qpoint:n { col - #3 }
3784     \dim_set_eq:NN \l_tmpa_dim \pgf@x
3785     \@@_qpoint:n { row - #1 }
3786     \dim_set_eq:NN \l_tmpb_dim \pgf@y
3787     \@@_qpoint:n { col - \@@_succ:n { #4 } }
3788     \dim_set_eq:NN \l_tmpc_dim \pgf@x
3789     \bool_lazy_and:nnT
3790     { \int_compare_p:nNn { #2 } > 1 }
3791     { ! \tl_if_blank_p:V \CT@drsc@ }
3792     {
3793         \group_begin:
3794         \CT@drsc@
3795         \dim_set:Nn \l_tmpd_dim
3796         { \l_tmpb_dim - ( \doublerulesep + \arrayrulewidth ) * ( #2 - 1 ) }
3797         \pgfpathrectanglecorners
3798         { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3799         { \pgfpoint \l_tmpc_dim \l_tmpd_dim }
3800         \pgfusepathqfill
3801         \group_end:
3802     }
3803     \pgfpathmoveto { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3804     \pgfpathlineto { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
3805     \prg_replicate:nn { #2 - 1 }
3806     {
3807         \dim_sub:Nn \l_tmpb_dim \arrayrulewidth
3808         \dim_sub:Nn \l_tmpb_dim \doublerulesep
3809         \pgfpathmoveto { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3810         \pgfpathlineto { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
3811     }
3812     \CT@arc@
3813     \pgfsetlinewidth { 1.1 \arrayrulewidth }
3814     \pgfsetrectcap
3815     \pgfusepathqstroke
3816 }

```

```

3817 \cs_new_protected:Npn \@@_hline_i_complete:nn #1 #2
3818 { \@@_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 `except-corners` is used).

```

3819 \cs_new_protected:Npn \@@_draw_hlines:
3820 {
3821   \int_step_inline:nnn
3822     { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
3823     { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@iRow } \c@iRow }
3824     { \@@_hline:nn { ##1 } 1 }
3825 }
```

The command `\@@_Hline:` will be linked to `\Hline` in the environments of `nicematrix`.

```

3826 \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.

```

3827 \cs_set:Npn \@@_Hline_i:n #1
3828 {
3829   \peek_meaning_ignore_spaces:NTF \Hline
3830     { \@@_Hline_ii:nn { #1 + 1 } }
3831     { \@@_Hline_iii:n { #1 } }
3832 }
3833 \cs_set:Npn \@@_Hline_ii:nn #1 #2 { \@@_Hline_i:n { #1 } }
3834 \cs_set:Npn \@@_Hline_iii:n #1
3835 {
3836   \skip_vertical:n
3837   {
3838     \arrayrulewidth * ( #1 )
3839     + \doublerulesep * ( \int_max:nn 0 { #1 - 1 } )
3840   }
3841   \tl_gput_right:Nx \g_@@_internal_code_after_tl
3842     { \@@_hline:nn { \@@_succ:n { \c@iRow } } { #1 } }
3843   \ifnum 0 = `{ \fi }
3844 }
```

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 col) would provide an horizontal rule towards the right in the block delimited by the four arguments `#1`, `#2`, `#3` and `#4`. If this rule would be in the block (it must not be drawn), the boolean `\l_tmpa_bool` is set to false.

```

3845 \cs_new_protected:Npn \@@_test_if_hline_in_block:nnnn #1 #2 #3 #4
3846 {
3847   \bool_lazy_all:nT
3848   {
3849     { \int_compare_p:nNn \l_tmpa_tl > { #1 } }
3850     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
3851     { \int_compare_p:nNn \l_tmpb_tl > { #2 - 1 } }
3852     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
3853   }
3854   { \bool_gset_false:N \g_tmpa_bool }
3855 }
```

The same for vertical rules.

```

3856 \cs_new_protected:Npn \@@_test_if_vline_in_block:nnnn #1 #2 #3 #4
3857 {
3858   \bool_lazy_all:nT
3859   {
3860     { \int_compare_p:nNn \l_tmpa_tl > { #1 - 1 } }
3861     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
3862     { \int_compare_p:nNn \l_tmpb_tl > { #2 } }
3863     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }

```

```

3864     }
3865     { \bool_gset_false:N \g_tmpa_bool }
3866 }

```

The key except-corners

When the key `except-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.

```

3867 \cs_new_protected:Npn \@@_compute_corners:
3868 {

```

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.

```

3869     \seq_clear_new:N \l_@@_empty_corner_cells_seq
3870     \clist_map_inline:Nn \l_@@_except_corners_clist
3871     {
3872         \str_case:nnF { ##1 }
3873         {
3874             { NW }
3875             { \@@_compute_a_corner:nnnnnn 1 1 1 1 \c@iRow \c@jCol }
3876             { NE }
3877             { \@@_compute_a_corner:nnnnnn 1 \c@jCol 1 { -1 } \c@iRow 1 }
3878             { SW }
3879             { \@@_compute_a_corner:nnnnnn \c@iRow 1 { -1 } 1 1 \c@jCol }
3880             { SE }
3881             { \@@_compute_a_corner:nnnnnn \c@iRow \c@jCol { -1 } { -1 } 1 1 }
3882         }
3883         { \@@_error:nn { bad~corner } { ##1 } }
3884     }
3885 }

```

“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.

```

3886 \cs_new_protected:Npn \@@_compute_a_corner:nnnnnn #1 #2 #3 #4 #5 #6
3887 {

```

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.

```

3888     \bool_set_false:N \l_tmpa_bool
3889     \int_zero_new:N \l_@@_last_empty_row_int
3890     \int_set:Nn \l_@@_last_empty_row_int { #1 }
3891     \int_step_inline:nnnn { #1 } { #3 } { #5 }
3892     {
3893         \@@_test_if_cell_in_a_block:nn { ##1 } { \int_eval:n { #2 } }
3894         \bool_lazy_or:nnTF
3895         {
3896             \cs_if_exist_p:c
3897             { pgf @ sh @ ns @ \@@_env: - ##1 - \int_eval:n { #2 } }
3898         }

```

```

3899     \l_tmpb_bool
3900     { \bool_set_true:N \l_tmpa_bool }
3901     {
3902         \bool_if:NF \l_tmpa_bool
3903         { \int_set:Nn \l_@@_last_empty_row_int { ##1 } }
3904     }
3905 }

```

Now, you determine the last empty cell in the row of number 1.

```

3906     \bool_set_false:N \l_tmpa_bool
3907     \int_zero_new:N \l_@@_last_empty_column_int
3908     \int_set:Nn \l_@@_last_empty_column_int { #2 }
3909     \int_step_inline:nnnn { #2 } { #4 } { #6 }
3910     {
3911         \@@_test_if_cell_in_a_block:nn { \int_eval:n { #1 } } { ##1 }
3912         \bool_lazy_or:nnTF
3913         \l_tmpb_bool
3914         {
3915             \cs_if_exist_p:c
3916             { pgf @ sh @ ns @ \@@_env: - \int_eval:n { #1 } - ##1 }
3917         }
3918         { \bool_set_true:N \l_tmpa_bool }
3919         {
3920             \bool_if:NF \l_tmpa_bool
3921             { \int_set:Nn \l_@@_last_empty_column_int { ##1 } }
3922         }
3923     }

```

Now, we loop over the rows.

```

3924     \int_step_inline:nnnn { #1 } { #3 } \l_@@_last_empty_row_int
3925     {

```

We treat the row number ##1 with another loop.

```

3926         \bool_set_false:N \l_tmpa_bool
3927         \int_step_inline:nnnn { #2 } { #4 } \l_@@_last_empty_column_int
3928         {
3929             \@@_test_if_cell_in_a_block:nn { ##1 } { #####1 }
3930             \bool_lazy_or:nnTF
3931             \l_tmpb_bool
3932             {
3933                 \cs_if_exist_p:c
3934                 { pgf @ sh @ ns @ \@@_env: - ##1 - #####1 }
3935             }
3936             { \bool_set_true:N \l_tmpa_bool }
3937             {
3938                 \bool_if:NF \l_tmpa_bool
3939                 {
3940                     \int_set:Nn \l_@@_last_empty_column_int { #####1 }
3941                     \seq_put_right:Nn
3942                     \l_@@_empty_corner_cells_seq
3943                     { ##1 - #####1 }
3944                 }
3945             }
3946         }
3947     }
3948 }

```

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`).

```

3949 \cs_new_protected:Npn \@@_test_if_cell_in_a_block:nn #1 #2
3950 {
3951     \int_set:Nn \l_tmpa_int { #1 }
3952     \int_set:Nn \l_tmpb_int { #2 }

```

```

3953 \bool_set_false:N \l_tmpb_bool
3954 \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
3955 { \@@_test_if_cell_in_block:nnnnnnn \l_tmpa_int \l_tmpb_int ##1 }
3956 }
3957 \cs_new_protected:Npn \@@_test_if_cell_in_block:nnnnnnn #1 #2 #3 #4 #5 #6
3958 {
3959   \int_compare:nNnT { #3 } < { \@@_succ:n { #1 } }
3960   {
3961     \int_compare:nNnT { #1 } < { \@@_succ:n { #5 } }
3962     {
3963       \int_compare:nNnT { #4 } < { \@@_succ:n { #2 } }
3964       {
3965         \int_compare:nNnT { #2 } < { \@@_succ:n { #6 } }
3966         { \bool_set_true:N \l_tmpb_bool }
3967       }
3968     }
3969   }
3970 }

```

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`.

```

3971 \cs_new:Npn \@@_hdottedline:
3972 {
3973   \noalign { \skip_vertical:N 2\l_@@_radius_dim }
3974   \@@_hdottedline_i:
3975 }

```

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

```

3976 \cs_new_protected:Npn \@@_hdottedline_i:
3977 {

```

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).

```

3978 \tl_gput_right:Nx \g_@@_internal_code_after_tl
3979 { \@@_hdottedline:n { \int_use:N \c@iRow } }
3980 }

```

The command `\@@_hdottedline:n` is the command written in the code-after that will actually draw the dotted line. Its argument is the number of the row *before* which we will draw the row.

```

3981 \AtBeginDocument
3982 {

```

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}`).

```

3983 \cs_new_protected:Npx \@@_hdottedline:n #1
3984 {
3985   \bool_set_true:N \exp_not:N \l_@@_initial_open_bool
3986   \bool_set_true:N \exp_not:N \l_@@_final_open_bool
3987   \c_@@_pgfortikzpicture_tl
3988   \@@_hdottedline_i:n { #1 }
3989   \c_@@_endpgfortikzpicture_tl
3990 }
3991 }

```

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

```

3992 \cs_new_protected:Npn \@@_hdottedline_i:n #1
3993 {
3994   \pgfrememberpicturepositiononpagetrue
3995   \@@_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).

```

3996   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
3997   \dim_sub:Nn \l_@@_y_initial_dim \l_@@_radius_dim
3998   \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim

```

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

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

```

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

```

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

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

```

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

```

$$\left[\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ \hdots & \hdots & \hdots & \hdots \\ 1 & 2 & 3 & 4 \end{array} \right]$$

```

3999   \@@_qpoint:n { col - 1 }
4000   \dim_set:Nn \l_@@_x_initial_dim
4001   {
4002     \pgf@x +

```

We do a reduction by `\arraycolsep` for the environments with delimiters (and not for the other).

```

4003     \bool_if:NTF \l_@@_NiceArray_bool \c_zero_dim \arraycolsep
4004     - \l_@@_left_margin_dim
4005   }
4006   \@@_qpoint:n { col - \@@_succ:n \c@jCol }
4007   \dim_set:Nn \l_@@_x_final_dim
4008   {
4009     \pgf@x -
4010     \bool_if:NTF \l_@@_NiceArray_bool \c_zero_dim \arraycolsep
4011     + \l_@@_right_margin_dim
4012   }

```

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.

```

4013   \tl_set:Nn \l_tmpa_tl { ( }
4014   \tl_if_eq:NnF \l_@@_left_delim_tl \l_tmpa_tl
4015   { \dim_gadd:Nn \l_@@_x_initial_dim { 0.5 \l_@@_inter_dots_dim } }
4016   \tl_set:Nn \l_tmpa_tl { ) }
4017   \tl_if_eq:NnF \l_@@_right_delim_tl \l_tmpa_tl
4018   { \dim_gsub:Nn \l_@@_x_final_dim { 0.5 \l_@@_inter_dots_dim } }

```

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

```

4019   \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
4020   \@@_draw_line:
4021 }

```

Vertical dotted lines

```

4022 \cs_new_protected:Npn \@@_vdottedline:n #1
4023 {
4024     \bool_set_true:N \l_@@_initial_open_bool
4025     \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”.

```

4026     \bool_if:NTF \c_@@_tikz_loaded_bool
4027     {
4028         \tikzpicture
4029         \@@_vdottedline_i:n { #1 }
4030         \endtikzpicture
4031     }
4032     {
4033         \pgfpicture
4034         \@@_vdottedline_i:n { #1 }
4035         \endpgfpicture
4036     }
4037 }

```

```

4038 \cs_new_protected:Npn \@@_vdottedline_i:n #1
4039 {

```

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.

```

4040     \CT@arc@
4041     \pgfrememberpicturepositiononpagetrue
4042     \@@_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).

```

4043     \dim_set:Nn \l_@@_x_initial_dim { \pgf@x - \l_@@_radius_dim }
4044     \dim_set:Nn \l_@@_x_final_dim { \pgf@x - \l_@@_radius_dim }
4045     \@@_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.

```

4046     \dim_set:Nn \l_@@_y_initial_dim { \pgf@y - 0.5 \l_@@_inter_dots_dim }
4047     \@@_qpoint:n { row - \@@_succ:n \c@iRow }
4048     \dim_set:Nn \l_@@_y_final_dim { \pgf@y + 0.5 \l_@@_inter_dots_dim }

```

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

```

4049     \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
4050     \@@_draw_line:
4051 }

```

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.

```

4052 \bool_new:N \l_@@_block_auto_columns_width_bool

```

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

```

4053 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
4054 {
4055     auto-columns-width .code:n =
4056     {

```



```

4057     \bool_set_true:N \l_@@_block_auto_columns_width_bool
4058     \dim_gzero_new:N \g_@@_max_cell_width_dim
4059     \bool_set_true:N \l_@@_auto_columns_width_bool
4060   }
4061 }

4062 \NewDocumentEnvironment { NiceMatrixBlock } { ! 0 { } }
4063 {
4064   \int_gincr:N \g_@@_NiceMatrixBlock_int
4065   \dim_zero:N \l_@@_columns_width_dim
4066   \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
4067   \bool_if:NT \l_@@_block_auto_columns_width_bool
4068   {
4069     \cs_if_exist:cT { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
4070     {
4071       \exp_args:Nnc \dim_set:Nn \l_@@_columns_width_dim
4072         { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
4073     }
4074   }
4075 }

```

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).

```

4076 {
4077   \bool_if:NT \l_@@_block_auto_columns_width_bool
4078   {
4079     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
4080     \iow_shipout:Nx \@mainaux
4081     {
4082       \cs_gset:cpn
4083         { @@_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.

```

4084       { \dim_eval:n { \g_@@_max_cell_width_dim + \arrayrulewidth } }
4085     }
4086     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
4087   }
4088 }

```

The extra nodes

First, two variants of the functions \dim_min:nn and \dim_max:nn.

```

4089 \cs_generate_variant:Nn \dim_min:nn { v n }
4090 \cs_generate_variant:Nn \dim_max:nn { v n }

```

We have three macros of creation of nodes: \@@_create_medium_nodes:, \@@_create_large_nodes: and \@@_create_medium_and_large_nodes:.

We have to compute the mathematical coordinates of the “medium nodes”. These mathematical coordinates are also used to compute the mathematical coordinates of the “large nodes”. That’s why we write a command \@@_computations_for_medium_nodes: to do these computations.

The command \@@_computations_for_medium_nodes: must be used in a {pgfpicture}.

For each row i , we compute two dimensions $l_@@_row_i_min_dim$ and $l_@@_row_i_max_dim$. The dimension $l_@@_row_i_min_dim$ is the minimal y -value of all the cells of the row i . The dimension $l_@@_row_i_max_dim$ is the maximal y -value of all the cells of the row i .

Similarly, for each column j , we compute two dimensions $l_@@_column_j_min_dim$ and $l_@@_column_j_max_dim$. The dimension $l_@@_column_j_min_dim$ is the minimal x -value of all the cells

of the column j . The dimension `l_@@_column_j_max_dim` is the maximal x -value of all the cells of the column j .

Since these dimensions will be computed as maximum or minimum, we initialize them to `\c_max_dim` or `-\c_max_dim`.

```

4091 \cs_new_protected:Npn \@@_computations_for_medium_nodes:
4092 {
4093   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4094   {
4095     \dim_zero_new:c { l_@@_row_\@@_i: _min_dim }
4096     \dim_set_eq:cN { l_@@_row_\@@_i: _min_dim } \c_max_dim
4097     \dim_zero_new:c { l_@@_row_\@@_i: _max_dim }
4098     \dim_set:cn { l_@@_row_\@@_i: _max_dim } { - \c_max_dim }
4099   }
4100   \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
4101   {
4102     \dim_zero_new:c { l_@@_column_\@@_j: _min_dim }
4103     \dim_set_eq:cN { l_@@_column_\@@_j: _min_dim } \c_max_dim
4104     \dim_zero_new:c { l_@@_column_\@@_j: _max_dim }
4105     \dim_set:cn { l_@@_column_\@@_j: _max_dim } { - \c_max_dim }
4106   }

```

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

```

4107   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4108   {
4109     \int_step_variable:nnNn
4110     \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.

```

4111     {
4112       \cs_if_exist:cT
4113       { 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`.

```

4114     {
4115       \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { south-west }
4116       \dim_set:cn { l_@@_row_\@@_i: _min_dim }
4117       { \dim_min:vn { l_@@_row _ \@@_i: _min_dim } \pgf@y }
4118       \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
4119       {
4120         \dim_set:cn { l_@@_column _ \@@_j: _min_dim }
4121         { \dim_min:vn { l_@@_column _ \@@_j: _min_dim } \pgf@x }
4122       }

```

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`.

```

4123       \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { north-east }
4124       \dim_set:cn { l_@@_row _ \@@_i: _ max_dim }
4125       { \dim_max:vn { l_@@_row _ \@@_i: _ max_dim } \pgf@y }
4126       \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
4127       {
4128         \dim_set:cn { l_@@_column _ \@@_j: _ max_dim }
4129         { \dim_max:vn { l_@@_column _ \@@_j: _ max_dim } \pgf@x }
4130       }
4131     }
4132   }
4133 }

```

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

```

4134   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4135   {
4136     \dim_compare:nNnT
4137     { \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } } = \c_max_dim

```

```

4138     {
4139         \@@_qpoint:n { row - \@@_i: - base }
4140         \dim_set:cn { l_@@_row _ \@@_i: _ max _ dim } \pgf@y
4141         \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim } \pgf@y
4142     }
4143 }
4144 \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
4145 {
4146     \dim_compare:nNnT
4147     { \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } } = \c_max_dim
4148     {
4149         \@@_qpoint:n { col - \@@_j: }
4150         \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim } \pgf@y
4151         \dim_set:cn { l_@@_column _ \@@_j: _ min _ dim } \pgf@y
4152     }
4153 }
4154 }

```

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

```

4155 \cs_new_protected:Npn \@@_create_medium_nodes:
4156 {
4157     \pgfpicture
4158     \pgfrememberpicturepositiononpagetrue
4159     \pgf@relevantforpicturesizefalse
4160     \@@_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”.

```

4161     \tl_set:Nn \l_@@_suffix_tl { -medium }
4162     \@@_create_nodes:
4163     \endpgfpicture
4164 }

```

The command `\@@_create_large_nodes:` must be used when we want to create only the “large nodes” and not the medium ones⁴³. 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:`.

```

4165 \cs_new_protected:Npn \@@_create_large_nodes:
4166 {
4167     \pgfpicture
4168     \pgfrememberpicturepositiononpagetrue
4169     \pgf@relevantforpicturesizefalse
4170     \@@_computations_for_medium_nodes:
4171     \@@_computations_for_large_nodes:
4172     \tl_set:Nn \l_@@_suffix_tl { - large }
4173     \@@_create_nodes:
4174     \endpgfpicture
4175 }
4176 \cs_new_protected:Npn \@@_create_medium_and_large_nodes:
4177 {
4178     \pgfpicture
4179     \pgfrememberpicturepositiononpagetrue
4180     \pgf@relevantforpicturesizefalse
4181     \@@_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”.

```

4182     \tl_set:Nn \l_@@_suffix_tl { - medium }

```

⁴³If we want to create both, we have to use `\@@_create_medium_and_large_nodes:`

```

4183 \@@_create_nodes:
4184 \@@_computations_for_large_nodes:
4185 \tl_set:Nn \l_@@_suffix_tl { - large }
4186 \@@_create_nodes:
4187 \endpgfpicture
4188 }

```

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.

```

4189 \cs_new_protected:Npn \@@_computations_for_large_nodes:
4190 {
4191   \int_set:Nn \l_@@_first_row_int 1
4192   \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`.

```

4193   \int_step_variable:nNn { \c@iRow - 1 } \@@_i:
4194   {
4195     \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim }
4196     {
4197       (
4198         \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } +
4199         \dim_use:c { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
4200       )
4201       / 2
4202     }
4203     \dim_set_eq:cc { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
4204     { l_@@_row _ \@@_i: _ min _ dim }
4205   }
4206   \int_step_variable:nNn { \c@jCol - 1 } \@@_j:
4207   {
4208     \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim }
4209     {
4210       (
4211         \dim_use:c { l_@@_column _ \@@_j: _ max _ dim } +
4212         \dim_use:c
4213         { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
4214       )
4215       / 2
4216     }
4217     \dim_set_eq:cc { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
4218     { l_@@_column _ \@@_j: _ max _ dim }
4219   }

```

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

```

4220   \dim_sub:cn
4221   { l_@@_column _ 1 _ min _ dim }
4222   \l_@@_left_margin_dim
4223   \dim_add:cn
4224   { l_@@_column _ \int_use:N \c@jCol _ max _ dim }
4225   \l_@@_right_margin_dim
4226 }

```

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).

```

4227 \cs_new_protected:Npn \@@_create_nodes:
4228 {
4229   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:

```

```

4230 {
4231   \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
4232   {

```

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

```

4233     \@@_pgf_rect_node:nnnnn
4234     { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
4235     { \dim_use:c { l_@@_column_ \@@_j: _min_dim } }
4236     { \dim_use:c { l_@@_row_ \@@_i: _min_dim } }
4237     { \dim_use:c { l_@@_column_ \@@_j: _max_dim } }
4238     { \dim_use:c { l_@@_row_ \@@_i: _max_dim } }
4239     \str_if_empty:NF \l_@@_name_str
4240     {
4241       \pgfnodealias
4242       { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
4243       { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
4244     }
4245   }
4246 }

```

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 .

```

4247   \seq_mapthread_function:NNN
4248   \g_@@_multicolumn_cells_seq
4249   \g_@@_multicolumn_sizes_seq
4250   \@@_node_for_multicolumn:nn
4251 }

```

```

4252 \cs_new_protected:Npn \@@_extract_coords_values: #1 - #2 \q_stop
4253 {
4254   \cs_set_nopar:Npn \@@_i: { #1 }
4255   \cs_set_nopar:Npn \@@_j: { #2 }
4256 }

```

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”).

```

4257 \cs_new_protected:Npn \@@_node_for_multicolumn:nn #1 #2
4258 {
4259   \@@_extract_coords_values: #1 \q_stop
4260   \@@_pgf_rect_node:nnnnn
4261   { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
4262   { \dim_use:c { l_@@_column_ \@@_j: _min_dim } }
4263   { \dim_use:c { l_@@_row_ \@@_i: _min_dim } }
4264   { \dim_use:c { l_@@_column_ \int_eval:n { \@@_j: +#2-1 } _max_dim } }
4265   { \dim_use:c { l_@@_row_ \@@_i: _max_dim } }
4266   \str_if_empty:NF \l_@@_name_str
4267   {
4268     \pgfnodealias
4269     { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
4270     { \int_use:N \g_@@_env_int - \@@_i: - \@@_j: \l_@@_suffix_tl }
4271   }
4272 }

```

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 in the `code-after`).

```

4273 \keys_define:nn { NiceMatrix / Block / FirstPass }
4274 {
4275   l .code:n = \tl_set:Nn \l_@@_pos_of_block_tl l ,
4276   l .value_forbidden:n = true ,
4277   r .code:n = \tl_set:Nn \l_@@_pos_of_block_tl r ,
4278   r .value_forbidden:n = true ,
4279   c .code:n = \tl_set:Nn \l_@@_pos_of_block_tl c ,
4280   c .value_forbidden:n = true ,
4281 }

```

The following command will be linked to `\Block` in the environments of `nicematrix`. We define it with `\NewDocumentCommand` of `xparse` because it has an optional argument between `<` and `>` (for TeX instructions put before the math mode of the label)

It's mandatory to use an expandable command (why?).

```

4282 \NewExpandableDocumentCommand \@@_Block: { 0 { } m D < > { } m }
4283 { \@@_Block_i #2 \q_stop { #1 } { #3 } { #4 } }

```

The first mandatory argument of `\@@_Block:` has a special syntax. It must be of the form $i-j$ where i and j are the size (in rows and columns) of the block.

```

4284 \cs_new:Npn \@@_Block_i #1-#2 \q_stop { \@@_Block_ii:nnnnn { #1 } { #2 } }

```

Now, the arguments have been extracted: `#1` is i (the number of rows of the block), `#2` is j (the number of columns of the block), `#3` is the list of key-values, `#4` are the tokens to put before the math mode and `#5` is the label of the block.

```

4285 \cs_new_protected:Npn \@@_Block_ii:nnnnn #1 #2 #3 #4 #5
4286 {
4287   \tl_if_empty:NTF \l_@@_cell_type_tl
4288   { \tl_set:Nn \l_@@_pos_of_block_tl c }
4289   { \tl_set_eq:NN \l_@@_pos_of_block_tl \l_@@_cell_type_tl }

4290   \keys_set_known:nn { NiceMatrix / Block / FirstPass } { #3 }

4291   \tl_set:Nx \l_tmpa_tl
4292   {
4293     { \int_use:N \c@iRow }
4294     { \int_use:N \c@jCol }
4295     { \int_eval:n { \c@iRow + #1 - 1 } }
4296     { \int_eval:n { \c@jCol + #2 - 1 } }
4297   }

```

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}`.

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

```

4298 \seq_gput_left:NV \g_@@_pos_of_blocks_seq \l_tmpa_tl

```

We also store a complete description of the block in the sequence `\g_@@_blocks_seq`. Of course, the sequences `\g_@@_pos_of_blocks_seq` and `\g_@@_blocks_seq` are redundant, but it's for efficiency.

In `\g_@@_blocks_seq`, each block is represented by an “object” with six components:

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

If the block is mono-column, we have a special treatment.

```

4299 \int_compare:nNnTF { #2 } = 1
4300 {
4301   \int_gincr:N \g_@@_block_box_int
4302   \cs_set_protected_nopar:Npn \diagbox ##1 ##2
4303   {
4304     \tl_gput_right:Nx \g_@@_internal_code_after_tl
4305     {
4306       \@@_actually_diagbox:nnnnnn

```

```

4307         { \int_use:N \c@iRow }
4308         { \int_use:N \c@jCol }
4309         { \int_eval:n { \c@iRow + #1 - 1 } }
4310         { \int_eval:n { \c@jCol + #2 - 1 } }
4311         { \exp_not:n { ##1 } } { \exp_not:n { ##2 } }
4312     }
4313 }
4314 \box_gclear_new:c
4315 { g_@@_block _ box _ \int_use:N \g_@@_block_box_int _ box }
4316 \hbox_gset:cn
4317 { g_@@_block _ box _ \int_use:N \g_@@_block_box_int _ box }
4318 {

```

The main aim of the following `\color_ensure_current:` is to retrieve in the box a color specified for that column in the preamble (e.g. something like `>\color{red}`}).

```

4319 \color_ensure_current:
4320 \bool_if:NTF \l_@@_NiceTabular_bool
4321 {
4322     \group_begin:
4323     \cs_set:Npn \arraystretch { 1 }
4324     \dim_set_eq:NN \extrarowheight \c_zero_dim
4325     #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).

```

4326     \bool_if:NT \g_@@_rotate_bool
4327     { \tl_set:Nn \l_@@_pos_of_block_tl c }
4328     \exp_args:Nnx \begin { tabular }
4329     { @ { } \l_@@_pos_of_block_tl @ { } }
4330     #5
4331     \end { tabular }
4332     \group_end:
4333 }
4334 {
4335     \group_begin:
4336     \cs_set:Npn \arraystretch { 1 }
4337     \dim_set_eq:NN \extrarowheight \c_zero_dim
4338     #4
4339     \bool_if:NT \g_@@_rotate_bool
4340     { \tl_set:Nn \l_@@_pos_of_block_tl c }
4341     \c_math_toggle_token
4342     \exp_args:Nnx \begin { array }
4343     { @ { } \l_@@_pos_of_block_tl @ { } }
4344     #5
4345     \end { array }
4346     \c_math_toggle_token
4347     \group_end:
4348 }
4349 }
4350 \bool_if:NT \g_@@_rotate_bool
4351 {
4352     \box_grotate:cn
4353     { g_@@_block _ box _ \int_use:N \g_@@_block_box_int _ box }
4354     { 90 }
4355     \bool_gset_false:N \g_@@_rotate_bool
4356 }
4357 \dim_gset:Nn \g_@@_blocks_width_dim
4358 {
4359     \dim_max:nn
4360     \g_@@_blocks_width_dim
4361     {

```

```

4362         \box_wd:c
4363         { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
4364     }
4365 }
4366 \seq_gput_right:Nx \g_@@_blocks_seq
4367 {
4368     \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_@@_pos_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_@@_pos_of_block_tl, which is fixed by the type of current column.

```

4369         { #3 , \l_@@_pos_of_block_tl }
4370     {
4371         \box_use_drop:c
4372         { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
4373     }
4374 }
4375 }

```

In the standard case, that is to say a \Block which is *not* mono-column.

```

4376 {
4377     \seq_gput_right:Nx \g_@@_blocks_seq
4378     {
4379         \l_tmpa_tl
4380         { #3 }
4381         \exp_not:n
4382         {
4383             {
4384                 \bool_if:NTF \l_@@_NiceTabular_bool
4385                 {
4386                     \group_begin:
4387                     \cs_set:Npn \arraystretch { 1 }
4388                     \dim_set_eq:NN \extrarowheight \c_zero_dim
4389                     #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).

```

4390         \bool_if:NT \g_@@_rotate_bool
4391         { \tl_set:Nn \l_@@_pos_of_block_tl c }
4392         \exp_args:Nnx \begin { tabular }
4393         { @ { } \l_@@_pos_of_block_tl @ { } }
4394         #5
4395         \end { tabular }
4396         \group_end:
4397     }
4398     {
4399         \group_begin:
4400         \cs_set:Npn \arraystretch { 1 }
4401         \dim_set_eq:NN \extrarowheight \c_zero_dim
4402         #4
4403         \bool_if:NT \g_@@_rotate_bool
4404         { \tl_set:Nn \l_@@_pos_of_block_tl c }
4405         \c_math_toggle_token
4406         \exp_args:Nnx \begin { array }
4407         { @ { } \l_@@_pos_of_block_tl @ { } } #5 \end { array }
4408         \c_math_toggle_token
4409         \group_end:
4410     }
4411 }
4412 }

```



```

4413     }
4414   }
4415 }

```

The key `tikz` is for Tikz options used when the PGF node of the block is created (the “normal” block node and not the “short” one nor the “medium” one). **In fact, as of now, it is *not* documented.** Is it really a good idea to provide such a key?

```

4416 \keys_define:nn { NiceMatrix / Block / SecondPass }
4417 {
4418   tikz .tl_set:N = \l_@@_tikz_tl ,
4419   tikz .value_required:n = true ,
4420   color .tl_set:N = \l_@@_color_tl ,
4421   color .value_required:n = true ,
4422   l .code:n = \tl_set:Nn \l_@@_pos_of_block_tl l ,
4423   l .value_forbidden:n = true ,
4424   r .code:n = \tl_set:Nn \l_@@_pos_of_block_tl r ,
4425   r .value_forbidden:n = true ,
4426   c .code:n = \tl_set:Nn \l_@@_pos_of_block_tl c ,
4427   c .value_forbidden:n = true ,
4428   unknown .code:n = \@@_error:n { Unknown~key~for~Block }
4429 }

```

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

```

4430 \cs_new_protected:Npn \@@_draw_blocks:
4431 { \seq_map_inline:Nn \g_@@_blocks_seq { \@@_Block_iii:nnnnnn ##1 } }
4432 \cs_new_protected:Npn \@@_Block_iii:nnnnnn #1 #2 #3 #4 #5 #6
4433 {

```

The group is for the keys.

```

4434   \group_begin:
4435   \keys_set:nn { NiceMatrix / Block / SecondPass } { #5 }
4436   \tl_if_empty:NF \l_@@_color_tl
4437   {
4438     \tl_gput_right:Nx \g_nicematrix_code_before_tl
4439     {
4440       \exp_not:N \rectanglecolor
4441       { \l_@@_color_tl }
4442       { #1 - #2 }
4443       { #3 - #4 }
4444     }
4445   }
4446   \cs_set_protected_nopar:Npn \diagbox ##1 ##2
4447   {
4448     \tl_gput_right:Nx \g_@@_internal_code_after_tl
4449     {
4450       \@@_actually_diagbox:nnnnnn
4451       { #1 } { #2 } { #3 } { #4 }
4452       { \exp_not:n { ##1 } } { \exp_not:n { ##2 } }
4453     }
4454   }
4455   \bool_lazy_or:nnTF
4456   { \int_compare_p:nNn { #3 } > \g_@@_row_total_int }
4457   { \int_compare_p:nNn { #4 } > \g_@@_col_total_int }
4458   { \msg_error:nnnn { nicematrix } { Block~too~large } { #1 } { #2 } }
4459   {
4460     \hbox_set:Nn \l_@@_cell_box { #6 }
4461     \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.

```
4462 \pgfpicture
4463 \pgfrememberpicturepositiononpagetrue
4464 \pgf@relevantforpicturesizefalse
4465 \@@_qpoint:n { row - #1 }
4466 \dim_set_eq:NN \l_tmpa_dim \pgf@y
4467 \@@_qpoint:n { col - #2 }
4468 \dim_set_eq:NN \l_tmpb_dim \pgf@x
4469 \@@_qpoint:n { row - \@@_succ:n { #3 } }
4470 \dim_set_eq:NN \l_tmpc_dim \pgf@y
4471 \@@_qpoint:n { col - \@@_succ:n { #4 } }
4472 \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.

```
4473 \begin { pgfscope }
4474 \exp_args:Nx \pgfset { \l_@@_tikz_tl }
4475 \@@_pgf_rect_node:nnnnn
4476 { \@@_env: - #1 - #2 - block }
4477 \l_tmpb_dim \l_tmpa_dim \l_tmpd_dim \l_tmpc_dim
4478 \end { pgfscope }
```

We construct the `short` node.

```
4479 \dim_set_eq:NN \l_tmpb_dim \c_max_dim
4480 \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
4481 {
```

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.

```
4482 \cs_if_exist:cT
4483 { pgf @ sh @ ns @ \@@_env: - ##1 - #2 }
4484 {
4485 \seq_if_in:NnF \g_@@_multicolumn_cells_seq { ##1 - #2 }
4486 {
4487 \pgfpointanchor { \@@_env: - ##1 - #2 } { west }
4488 \dim_set:Nn \l_tmpb_dim { \dim_min:nn \l_tmpb_dim \pgf@x }
4489 }
4490 }
4491 }
```

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.

```

4492     \dim_compare:nNnT \l_tmpb_dim = \c_max_dim
4493     {
4494         \@@_qpoint:n { col - #2 }
4495         \dim_set_eq:NN \l_tmpb_dim \pgf@x
4496     }
4497     \dim_set:Nn \l_tmpd_dim { - \c_max_dim }
4498     \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
4499     {
4500         \cs_if_exist:cT
4501         { \pgf @ sh @ ns @ \@@_env: - ##1 - #4 }
4502         {
4503             \seq_if_in:NnF \g_@@_multicolumn_cells_seq { ##1 - #2 }
4504             {
4505                 \pgfpointanchor { \@@_env: - ##1 - #4 } { east }
4506                 \dim_set:Nn \l_tmpd_dim { \dim_max:nn \l_tmpd_dim \pgf@x }
4507             }
4508         }
4509     }
4510     \dim_compare:nNnT \l_tmpd_dim = { - \c_max_dim }
4511     {
4512         \@@_qpoint:n { col - \@@_succ:n { #4 } }
4513         \dim_set_eq:NN \l_tmpd_dim \pgf@x
4514     }
4515     \@@_pgf_rect_node:nnnnn
4516     { \@@_env: - #1 - #2 - block - short }
4517     \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.

```

4518     \bool_if:NT \l_@@_medium_nodes_bool
4519     {
4520         \@@_pgf_rect_node:nnn
4521         { \@@_env: - #1 - #2 - block - medium }
4522         { \pgfpointanchor { \@@_env: - #1 - #2 - medium } { north-west } }
4523         { \pgfpointanchor { \@@_env: - #3 - #4 - medium } { south-east } }
4524     }

```

Now, we will put the label of the block beginning with the case of a `\Block` of one row.

```

4525     \int_compare:nNnTF { #1 } = { #3 }
4526     {

```

We take into account the case of a block of one row in the “first row” or the “last row”.

```

4527         \int_compare:nNnTF { #1 } = 0
4528         { \l_@@_code_for_first_row_tl }
4529         {
4530             \int_compare:nNnT { #1 } = \l_@@_last_row_int
4531             \l_@@_code_for_last_row_tl
4532         }

```

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`.

```

4533     \pgfextracty \l_tmpa_dim { \@@_qpoint:n { row - #1 - base } }

```

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

```

4534     \pgfpointanchor
4535     { \@@_env: - #1 - #2 - block - short }
4536     {
4537         \str_case:Vn \l_@@_pos_of_block_tl
4538         {
4539             c { center }
4540             l { west }

```

```

4541         r { east }
4542     }
4543 }

```

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

```

4544     \pgftransformshift { \pgfpoint \pgf@x \l_tmpa_dim }
4545     \pgfset { inner~sep = \c_zero_dim }
4546     \pgfnode
4547     { rectangle }
4548     {
4549         \str_case:Vn \l_@@_pos_of_block_tl
4550         {
4551             c { base }
4552             l { base~west }
4553             r { base~east }
4554         }
4555     }
4556     { \box_use_drop_xi:N \l_@@_cell_box } { } { }
4557 }

```

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

```

4558     {

```

If we are in the first column, we must put the block as if it was with the key `r`.

```

4559         \int_compare:nNnT \c@jCol = 0
4560         { \tl_set:Nn \l_@@_pos_of_block_tl r }
4561         \int_compare:nNnT \c@jCol = \l_@@_last_col_int
4562         { \tl_set:Nn \l_@@_pos_of_block_tl l }
4563         \pgftransformshift
4564         {
4565             \pgfpointanchor
4566             { \@@_env: - #1 - #2 - block - short }
4567             {
4568                 \str_case:Vn \l_@@_pos_of_block_tl
4569                 {
4570                     c { center }
4571                     l { west }
4572                     r { east }
4573                 }
4574             }
4575         }
4576         \pgfset { inner~sep = \c_zero_dim }
4577         \pgfnode
4578         { rectangle }
4579         {
4580             \str_case:Vn \l_@@_pos_of_block_tl
4581             {
4582                 c { center }
4583                 l { west }
4584                 r { east }
4585             }
4586         }
4587         { \box_use_drop_xii:N \l_@@_cell_box } { } { }
4588     }
4589     \endpgfpicture
4590 }
4591 \group_end:
4592 }

```

How to draw the dotted lines transparently

```

4593 \cs_set_protected:Npn \@@_renew_matrix:
4594 {
4595   \RenewDocumentEnvironment { pmatrix } { }
4596   { \pNiceMatrix }
4597   { \endpNiceMatrix }
4598   \RenewDocumentEnvironment { vmatrix } { }
4599   { \vNiceMatrix }
4600   { \endvNiceMatrix }
4601   \RenewDocumentEnvironment { Vmatrix } { }
4602   { \VNiceMatrix }
4603   { \endVNiceMatrix }
4604   \RenewDocumentEnvironment { bmatrix } { }
4605   { \bNiceMatrix }
4606   { \endbNiceMatrix }
4607   \RenewDocumentEnvironment { Bmatrix } { }
4608   { \BNiceMatrix }
4609   { \endBNiceMatrix }
4610 }

```

Automatic arrays

```

4611 \cs_new_protected:Npn \@@_set_size:n #1-#2 \q_stop
4612 {
4613   \int_set:Nn \l_@@_nb_rows_int { #1 }
4614   \int_set:Nn \l_@@_nb_cols_int { #2 }
4615 }
4616 \NewDocumentCommand \AutoNiceMatrixWithDelims { m m O { } m O { } m ! O { } }
4617 {
4618   \int_zero_new:N \l_@@_nb_rows_int
4619   \int_zero_new:N \l_@@_nb_cols_int
4620   \@@_set_size:n #4 \q_stop
4621   \begin { NiceArrayWithDelims } { #1 } { #2 }
4622     { * { \l_@@_nb_cols_int } { c } } [ #3 , #5 , #7 ]
4623   \int_compare:nNnT \l_@@_first_row_int = 0
4624     {
4625     \int_compare:nNnT \l_@@_first_col_int = 0 { & }
4626     \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
4627     \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
4628   }
4629   \prg_replicate:nn \l_@@_nb_rows_int
4630   {
4631     \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`).

```

4632   \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { { } #6 & } #6
4633   \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
4634 }
4635 \int_compare:nNnT \l_@@_last_row_int > { -2 }
4636 {
4637   \int_compare:nNnT \l_@@_first_col_int = 0 { & }
4638   \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
4639   \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
4640 }
4641 \end { NiceArrayWithDelims }
4642 }
4643 \cs_set_protected:Npn \@@_define_com:nnn #1 #2 #3
4644 {
4645   \cs_set_protected:cpn { #1 AutoNiceMatrix }
4646   {

```

```

4647     \str_gset:Nx \g_@@_name_env_str { #1 AutoNiceMatrix }
4648     \AutoNiceMatrixWithDelims { #2 } { #3 }
4649   }
4650 }

4651 \@@_define_com:nnn p ( )
4652 \@@_define_com:nnn b [ ]
4653 \@@_define_com:nnn v | |
4654 \@@_define_com:nnn V \| \|
4655 \@@_define_com:nnn B \{ \}

```

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

```

4656 \NewDocumentCommand \AutoNiceMatrix { 0 { } m 0 { } m ! 0 { } }
4657 {
4658   \group_begin:
4659     \bool_set_true:N \l_@@_NiceArray_bool
4660     \AutoNiceMatrixWithDelims . . { #2 } { #4 } [ #1 , #3 , #5 ]
4661   \group_end:
4662 }

```

The redefinition of the command `\dotfill`

```

4663 \cs_set_eq:NN \@@_old_dotfill \dotfill
4664 \cs_new_protected:Npn \@@_dotfill:
4665 {

```

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}`).

```

4666   \@@_old_dotfill
4667   \bool_if:NT \l_@@_NiceTabular_bool
4668     { \group_insert_after:N \@@_dotfill_ii: }
4669     { \group_insert_after:N \@@_dotfill_i: }
4670 }
4671 \cs_new_protected:Npn \@@_dotfill_i: { \group_insert_after:N \@@_dotfill_ii: }
4672 \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`.

```

4673 \cs_new_protected:Npn \@@_dotfill_iii:
4674 { \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`.

```

4675 \cs_new_protected:Npn \@@_diagbox:nn #1 #2
4676 {
4677   \tl_gput_right:Nx \g_@@_internal_code_after_tl
4678   {
4679     \@@_actually_diagbox:nnnnnn
4680     { \int_use:N \c_iRow }
4681     { \int_use:N \c_jCol }
4682     { \int_use:N \c_iRow }
4683     { \int_use:N \c_jCol }
4684     { \exp_not:n { #1 } }
4685     { \exp_not:n { #2 } }
4686   }

```

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 `except-corners`.

```

4687   \seq_gput_right:Nx \g_@@_pos_of_blocks_seq
4688   {
4689     { \int_use:N \c_iRow }

```

```

4690     { \int_use:N \c@jCol }
4691     { \int_use:N \c@iRow }
4692     { \int_use:N \c@jCol }
4693   }
4694 }

```

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.

```

4695 \cs_new_protected:Npn \@@_actually_diagbox:nnnnnn #1 #2 #3 #4 #5 #6
4696 {
4697   \pgfpicture
4698   \pgf@relevantforpicturesizefalse
4699   \pgfrememberpicturepositiononpagetrue
4700   \@@_qpoint:n { row - #1 }
4701   \dim_set_eq:NN \l_tmpa_dim \pgf@y
4702   \@@_qpoint:n { col - #2 }
4703   \dim_set_eq:NN \l_tmpb_dim \pgf@x
4704   \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
4705   \@@_qpoint:n { row - \@@_succ:n { #3 } }
4706   \dim_set_eq:NN \l_tmpc_dim \pgf@y
4707   \@@_qpoint:n { col - \@@_succ:n { #4 } }
4708   \dim_set_eq:NN \l_tmpd_dim \pgf@x
4709   \pgfpathlineto { \pgfpoint \l_tmpd_dim \l_tmpc_dim }
4710   {

```

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.

```

4711     \CT@arc@
4712     \pgfsetroundcap
4713     \pgfusepathqstroke
4714   }
4715   \pgfset { inner~sep = 1 pt }
4716   \pgfscope
4717   \pgftransformshift { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
4718   \pgfnode { rectangle } { south-west }
4719     { \@@_math_toggle_token: #5 \@@_math_toggle_token: } { } { }
4720   \endpgfscope
4721   \pgftransformshift { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
4722   \pgfnode { rectangle } { north-east }
4723     { \@@_math_toggle_token: #6 \@@_math_toggle_token: } { } { }
4724   \endpgfpicture
4725 }

```

The keyword `\CodeAfter`

In fact, in this subsection, we define the user command `\CodeAfter` for the case of the “normal syntax”. For the case of “light-syntax”, see the definition of the environment `{@@-light-syntax}` on p. 88.

The command `\CodeAfter` catches everything until the end of the current environment (of `nicematrix`). First, we go until the next command `\end`.

```

4726 \cs_new_protected:Npn \@@_CodeAfter:n #1 \end
4727 {
4728   \tl_gput_right:Nn \g_nicematrix_code_after_tl { #1 }
4729   \@@_CodeAfter_i:n
4730 }

```

We catch the argument of the command `\end` (in #1).

```

4731 \cs_new_protected:Npn \@@_CodeAfter_i:n #1
4732 {

```

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.

```
4733 \str_if_eq:eeTF \@currenvir { #1 }
4734 { \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`.

```
4735 {
4736   \tl_gput_right:Nn \g_nicematrix_code_after_tl { \end { #1 } }
4737   \@@_CodeAfter:n
4738 }
4739 }
```

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.

```
4740 \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.

```
4741 \bool_new:N \c_@@_footnote_bool
4742 \@@_msg_new:nnn { Unknown~option~for~package }
4743 {
4744   The~option~'\l_keys_key_tl'~is~unknown. \\
4745   If~you~go~on,~it~will~be~ignored. \\
4746   For~a~list~of~the~available~options,~type~H~<return>.
4747 }
4748 {
4749   The~available~options~are~(in~alphabetic~order):~
4750   define-L-C-R,~
4751   footnote,~
4752   footnotehyper,~
4753   renew-dots,~
4754   renew-matrix~and~
4755   transparent.
4756 }
4757 \keys_define:nn { NiceMatrix / Package }
4758 {
4759   define-L-C-R .bool_set:N = \c_@@_define_L_C_R_bool ,
4760   define-L-C-R .default:n = true ,
4761   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
4762   renew-dots .value_forbidden:n = true ,
4763   renew-matrix .code:n = \@@_renew_matrix: ,
4764   renew-matrix .value_forbidden:n = true ,
4765   transparent .meta:n = { renew-dots , renew-matrix } ,
4766   transparent .value_forbidden:n = true ,
4767   footnote .bool_set:N = \c_@@_footnote_bool ,
4768   footnotehyper .bool_set:N = \c_@@_footnotehyper_bool ,
4769   unknown .code:n = \@@_error:n { Unknown~option~for~package }
4770 }
4771 \ProcessKeysOptions { NiceMatrix / Package }
```



```

4772 \@@_msg_new:nn { footnote-with-footnotehyper-package }
4773 {
4774   You~can't~use~the~option~'footnote'~because~the~package~
4775   footnotehyper~has~already~been~loaded.~
4776   If~you~want,~you~can~use~the~option~'footnotehyper'~and~the~footnotes~
4777   within~the~environments~of~nicematrix~will~be~extracted~with~the~tools~
4778   of~the~package~footnotehyper.\\
4779   If~you~go~on,~the~package~footnote~won't~be~loaded.
4780 }
4781 \@@_msg_new:nn { footnotehyper-with-footnote-package }
4782 {
4783   You~can't~use~the~option~'footnotehyper'~because~the~package~
4784   footnote~has~already~been~loaded.~
4785   If~you~want,~you~can~use~the~option~'footnote'~and~the~footnotes~
4786   within~the~environments~of~nicematrix~will~be~extracted~with~the~tools~
4787   of~the~package~footnote.\\
4788   If~you~go~on,~the~package~footnotehyper~won't~be~loaded.
4789 }
4790 \bool_if:NT \c_@@_footnote_bool
4791 {
4792   \@ifclassloaded { beamer }
4793     { \msg_info:nn { nicematrix } { Option~incompatible~with~Beamer } }
4794     {
4795       \@ifpackageloaded { footnotehyper }
4796         { \@@_error:n { footnote-with-footnotehyper-package } }
4797         { \usepackage { footnote } }
4798     }
4799 }
4800 \bool_if:NT \c_@@_footnotehyper_bool
4801 {
4802   \@ifclassloaded { beamer }
4803     { \@@_info:n { Option~incompatible~with~Beamer } }
4804     {
4805       \@ifpackageloaded { footnote }
4806         { \@@_error:n { footnotehyper-with-footnote-package } }
4807         { \usepackage { footnotehyper } }
4808     }
4809   \bool_set_true:N \c_@@_footnote_bool
4810 }

```

The flag `\c_@@_footnote_bool` is raised and so, we will only have to test `\c_@@_footnote_bool` in order to know if we have to insert an environment `{savenotes}`.

Error messages of the package

The following command converts all the elements of a sequence (which are token lists) into strings.

```

4811 \cs_new_protected:Npn \@@_convert_to_str_seq:N #1
4812 {
4813   \seq_clear:N \l_tmpa_seq
4814   \seq_map_inline:Nn #1
4815     {
4816       \seq_put_left:Nx \l_tmpa_seq { \tl_to_str:n { ##1 } }
4817     }
4818   \seq_set_eq:NN #1 \l_tmpa_seq
4819 }

```

The following command creates a sequence of strings (`str`) from a `clist`.

```

4820 \cs_new_protected:Npn \@@_set_seq_of_str_from_clist:Nn #1 #2
4821 {
4822   \seq_set_from_clist:Nn #1 { #2 }

```

```

4823 \@@_convert_to_str_seq:N #1
4824 }
4825 \@@_set_seq_of_str_from_clist:Nn \c_@@_types_of_matrix_seq
4826 {
4827   NiceMatrix ,
4828   pNiceMatrix , bNiceMatrix , vNiceMatrix, BNiceMatrix, VNiceMatrix
4829 }

```

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`.

```

4830 \cs_new_protected:Npn \@@_error_too_much_cols:
4831 {
4832   \seq_if_in:NVTF \c_@@_types_of_matrix_seq \g_@@_name_env_str
4833   {
4834     \int_compare:nNnTF \l_@@_last_col_int = { -2 }
4835     { \@@_fatal:n { too~much~cols~for~matrix } }
4836     {
4837       \bool_if:NF \l_@@_last_col_without_value_bool
4838       { \@@_fatal:n { too~much~cols~for~matrix~with~last~col } }
4839     }
4840   }
4841   { \@@_fatal:n { too~much~cols~for~array } }
4842 }

```

The following command must *not* be protected since it's used in an error message.

```

4843 \cs_new:Npn \@@_message_hdotsfor:
4844 {
4845   \tl_if_empty:VF \g_@@_HVdotsfor_lines_tl
4846   { ~Maybe~your~use~of~\token_to_str:N \Hdotsfor\ is~incorrect.}
4847 }
4848 \@@_msg_new:nn { too~much~cols~for~matrix~with~last~col }
4849 {
4850   You~try~to~use~more~columns~than~allowed~by~your~
4851   \@@_full_name_env:.\@@_message_hdotsfor:\ The~maximal~number~of~
4852   columns~is~\int_eval:n { \l_@@_last_col_int - 1 }~(plus~the~
4853   exterior~columns).~This~error~is~fatal.
4854 }
4855 \@@_msg_new:nn { too~much~cols~for~matrix }
4856 {
4857   You~try~to~use~more~columns~than~allowed~by~your~
4858   \@@_full_name_env:.\@@_message_hdotsfor:\ Recall~that~the~maximal~
4859   number~of~columns~for~a~matrix~is~fixed~by~the~LaTeX~counter~
4860   'MaxMatrixCols'.~Its~actual~value~is~\int_use:N \c@MaxMatrixCols.~
4861   This~error~is~fatal.
4862 }

```

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`.

```

4863 \@@_msg_new:nn { too~much~cols~for~array }
4864 {
4865   You~try~to~use~more~columns~than~allowed~by~your~
4866   \@@_full_name_env:.\@@_message_hdotsfor:\ The~maximal~number~of~columns~is~
4867   \int_use:N \g_@@_static_num_of_col_int\
4868   ~(plus~the~potential~exterior~ones).~
4869   This~error~is~fatal.
4870 }
4871 \@@_msg_new:nn { last~col~not~used }
4872 {
4873   The~key~'last~col'~is~in~force~but~you~have~not~used~that~last~column~

```

```

4874     in~your~\@@_full_name_env:~However,~you~can~go~on.
4875 }
4876 \@@_msg_new:nn { columns~not~used }
4877 {
4878     The~preamble~of~your~\@@_full_name_env:\ announces~\int_use:N
4879     \g_@@_static_num_of_col_int\
4880     columns~but~you~use~only~\int_use:N \c@jCol.\
4881     However,~you~can~go~on.
4882 }
4883 \@@_msg_new:nn { in~first~col }
4884 {
4885     You~can't~use~the~command~#1 in~the~first~column~(number~0)~of~the~array.\
4886     If~you~go~on,~this~command~will~be~ignored.
4887 }
4888 \@@_msg_new:nn { in~last~col }
4889 {
4890     You~can't~use~the~command~#1 in~the~last~column~(exterior)~of~the~array.\
4891     If~you~go~on,~this~command~will~be~ignored.
4892 }
4893 \@@_msg_new:nn { in~first~row }
4894 {
4895     You~can't~use~the~command~#1 in~the~first~row~(number~0)~of~the~array.\
4896     If~you~go~on,~this~command~will~be~ignored.
4897 }
4898 \@@_msg_new:nn { in~last~row }
4899 {
4900     You~can't~use~the~command~#1 in~the~last~row~(exterior)~of~the~array.\
4901     If~you~go~on,~this~command~will~be~ignored.
4902 }
4903 \@@_msg_new:nn { bad~option~for~line~style }
4904 {
4905     Since~you~haven't~loaded~Tikz,~the~only~value~you~can~give~to~'line~style'~
4906     is~'standard'.~If~you~go~on,~this~option~will~be~ignored.
4907 }
4908 \@@_msg_new:nn { Unknown~option~for~xdots }
4909 {
4910     As~for~now~there~is~only~three~options~available~here:~'color',~'line~style'~
4911     and~'shorten'~(and~you~try~to~use~'\l_keys_key_tl').~If~you~go~on,~
4912     this~option~will~be~ignored.
4913 }
4914 \@@_msg_new:nn { Unknown~option~for~rowcolors }
4915 {
4916     As~for~now~there~is~only~one~option~available~here:~'respect~blocks'~
4917     (and~you~try~to~use~'\l_keys_key_tl').~If~you~go~on,~
4918     this~option~will~be~ignored.
4919 }
4920 \@@_msg_new:nn { ampersand~in~light~syntax }
4921 {
4922     You~can't~use~an~ampersand~(\token_to_str &)~to~separate~columns~because
4923     ~you~have~used~the~option~'light~syntax'.~This~error~is~fatal.
4924 }
4925 \@@_msg_new:nn { double~backslash~in~light~syntax }
4926 {
4927     You~can't~use~\token_to_str:N \~to~separate~rows~because~you~have~used~
4928     the~option~'light~syntax'.~You~must~use~the~character~'\l_@@_end_of_row_tl'~
4929     (set~by~the~option~'end~of~row').~This~error~is~fatal.
4930 }
4931 \@@_msg_new:nn { standard~cline~in~document }
4932 {

```

```

4933 The-key~'standard-cline'~is~available~only~in~the~preamble.\\
4934 If-you-go-on~this~command~will~be~ignored.
4935 }

4936 \@@_msg_new:nn { bad-value-for-baseline }
4937 {
4938 The-value-given-to~'baseline'~(\int_use:N \l_tmpa_int)~is~not~
4939 valid.~The-value-must-be-between~\int_use:N \l_@@_first_row_int\ and~
4940 \int_use:N \g_@@_row_total_int\ or-equal-to~'t',~'c'~or~'b'.\\
4941 If-you-go-on,~a-value-of~1~will~be~used.
4942 }

4943 \@@_msg_new:nn { empty-environment }
4944 { Your~\@@_full_name_env:\ is-empty.~This-error-is-fatal. }

4945 \@@_msg_new:nn { unknown-cell-for-line-in-code-after }
4946 {
4947 Your-command~\token_to_str:N\line\{#1\}\{#2\}~in-the~'code-after'~
4948 can't-be-executed~because~a~cell~doesn't~exist.\\
4949 If-you-go-on~this~command~will~be~ignored.
4950 }

4951 \@@_msg_new:nn { Hdotsfor~in~col~0 }
4952 {
4953 You-can't-use~\token_to_str:N \Hdotsfor\ in-an-exterior~column-of~
4954 the-array.~If-you-go-on,~the-corresponding-dotted-line-won't-be-drawn.
4955 }

4956 \@@_msg_new:nn { bad-corner }
4957 {
4958 #1~is-an-incorrect-specification~for~a~corner~(in~the~keys~
4959 'except-corners'~and~'hlines-except-corners').~The-available~
4960 values~are:~NW,~SW,~NE~and~SE.\\
4961 If-you-go-on,~this-specification-of~corner~will~be~ignored.
4962 }

4963 \@@_msg_new:nn { last-col-non-empty-for-NiceArray }
4964 {
4965 In-the~\@@_full_name_env:,~you-must-use~the-option~
4966 'last-col'~without-value.\\
4967 However,~you-can-go-on~for~this~time~
4968 (the-value~'\l_keys_value_tl'~will~be~ignored).
4969 }

4970 \@@_msg_new:nn { last-col-non-empty-for-NiceMatrixOptions }
4971 {
4972 In~\NiceMatrixoptions,~you-must-use~the-option~
4973 'last-col'~without-value.\\
4974 However,~you-can-go-on~for~this~time~
4975 (the-value~'\l_keys_value_tl'~will~be~ignored).
4976 }

4977 \@@_msg_new:nn { Block-too-large }
4978 {
4979 You-try-to~draw~a~block~in~the~cell~#1~#2~of~your~matrix~but~the~matrix~is~
4980 too-small~for~that~block. \\
4981 }

4982 \@@_msg_new:nn { unknown-column-type }
4983 {
4984 The~column~type~'#1'~in~your~\@@_full_name_env:\
4985 is-unknown. \\
4986 This-error-is-fatal.
4987 }

4988 \@@_msg_new:nn { tabularnote-forbidden }
4989 {
4990 You-can't-use~the~command~\token_to_str:N\tabularnote\
4991 ~in~a~\@@_full_name_env:~.~This-command-is-available-only~in~

```

```

4992   \{NiceTabular\},~\{NiceArray\}~and~\{NiceMatrix\}. \\
4993   If~you~go~on,~this~command~will~be~ignored.
4994 }

4995 \@@_msg_new:nn { bottomrule~without~booktabs }
4996 {
4997   You~can't~use~the~option~'tabular/bottomrule'~because~you~haven't~
4998   loaded~'booktabs'.\\
4999   If~you~go~on,~this~option~will~be~ignored.
5000 }

5001 \@@_msg_new:nn { enumitem~not~loaded }
5002 {
5003   You~can't~use~the~command~\token_to_str:N\tabularnote\
5004   ~because~you~haven't~loaded~'enumitem'.\\
5005   If~you~go~on,~this~command~will~be~ignored.
5006 }

5007 \@@_msg_new:nn { Wrong~last~row }
5008 {
5009   You~have~used~'last-row=\int_use:N \l_@@_last_row_int'~but~your~
5010   \@@_full_name_env:\ seems~to~have~\int_use:N \c@iRow \ rows.~
5011   If~you~go~on,~the~value~of~\int_use:N \c@iRow \ will~be~used~for~
5012   last~row.~You~can~avoid~this~problem~by~using~'last-row'~
5013   without~value~(more~compilations~might~be~necessary).
5014 }

5015 \@@_msg_new:nn { Yet~in~env }
5016 { Environments~of~nicematrix~can't~be~nested.\\ This~error~is~fatal. }

5017 \@@_msg_new:nn { Outside~math~mode }
5018 {
5019   The~\@@_full_name_env:\ can~be~used~only~in~math~mode~
5020   (and~not~in~\token_to_str:N \vcenter).\\
5021   This~error~is~fatal.
5022 }

5023 \@@_msg_new:nn { Bad~value~for~letter~for~dotted~lines }
5024 {
5025   The~value~of~key~'\l_keys_key_tl'~must~be~of~length~1.\\
5026   If~you~go~on,~it~will~be~ignored.
5027 }

5028 \@@_msg_new:nnn { Unknown~key~for~Block }
5029 {
5030   The~key~'\l_keys_key_tl'~is~unknown~for~the~command~\token_to_str:N
5031   \Block.\\ If~you~go~on,~it~will~be~ignored. \\
5032   For~a~list~of~the~available~keys,~type~H~<return>.
5033 }
5034 {
5035   The~available~options~are~(in~alphabetic~order):~,~,~c,~
5036   color,~l,~and~r.
5037 }

5038 \@@_msg_new:nnn { Unknown~key~for~notes }
5039 {
5040   The~key~'\l_keys_key_tl'~is~unknown.\\
5041   If~you~go~on,~it~will~be~ignored. \\
5042   For~a~list~of~the~available~keys~about~notes,~type~H~<return>.
5043 }
5044 {
5045   The~available~options~are~(in~alphabetic~order):~
5046   bottomrule,~
5047   code~after,~
5048   code~before,~
5049   enumitem~keys,~
5050   enumitem~keys~para,~
5051   para,~

```

```

5052     label-in-list,~
5053     label-in-tabular~and~
5054     style.
5055 }
5056 \@@_msg_new:nnn { Unknown~key~for~NiceMatrixOptions }
5057 {
5058     The~key~'\l_keys_key_tl'~is~unknown~for~the~command~
5059     \token_to_str:N \NiceMatrixOptions. \\
5060     If~you~go~on,~it~will~be~ignored. \\
5061     For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
5062 }
5063 {
5064     The~available~options~are~(in~alphabetic~order):~
5065     allow-duplicate-names,~
5066     cell-space-bottom-limit,~
5067     cell-space-top-limit,~
5068     code-for-first-col,~
5069     code-for-first-row,~
5070     code-for-last-col,~
5071     code-for-last-row,~
5072     create-extra-nodes,~
5073     create-medium-nodes,~
5074     create-large-nodes,~
5075     end-of-row,~
5076     first-col,~
5077     first-row,~
5078     hlines,~
5079     hvlines,~
5080     hvlines-except-corners,~
5081     last-col,~
5082     last-row,~
5083     left-margin,~
5084     letter-for-dotted-lines,~
5085     light-syntax,~
5086     notes~(several subkeys),~
5087     nullify-dots,~
5088     renew-dots,~
5089     renew-matrix,~
5090     right-margin,~
5091     small,~
5092     transparent,~
5093     vlines,~
5094     xdots/color,~
5095     xdots/shorten~and~
5096     xdots/line-style.
5097 }
5098 \@@_msg_new:nnn { Unknown~option~for~NiceArray }
5099 {
5100     The~option~'\l_keys_key_tl'~is~unknown~for~the~environment~
5101     \{NiceArray\}. \\
5102     If~you~go~on,~it~will~be~ignored. \\
5103     For~a~list~of~the~*principal*~available~options,~type~H~<return>.
5104 }
5105 {
5106     The~available~options~are~(in~alphabetic~order):~
5107     b,~
5108     baseline,~
5109     c,~
5110     cell-space-bottom-limit,~
5111     cell-space-top-limit,~
5112     code-after,~
5113     code-for-first-col,~
5114     code-for-first-row,~

```

```

5115 code-for-last-col,~
5116 code-for-last-row,~
5117 colortbl-like,~
5118 columns-width,~
5119 create-extra-nodes,~
5120 create-medium-nodes,~
5121 create-large-nodes,~
5122 extra-left-margin,~
5123 extra-right-margin,~
5124 first-col,~
5125 first-row,~
5126 hlines,~
5127 hvlines,~
5128 hvlines-except-corners,~
5129 last-col,~
5130 last-row,~
5131 left-margin,~
5132 light-syntax,~
5133 name,~
5134 notes/bottomrule,~
5135 notes/para,~
5136 nullify-dots,~
5137 renew-dots,~
5138 right-margin,~
5139 rules/color,~
5140 rules/width,~
5141 small,~
5142 t,~
5143 vlines,~
5144 xdots/color,~
5145 xdots/shorten~and~
5146 xdots/line-style.
5147 }

```

This error message is used for the set of keys `NiceMatrix/NiceMatrix` and `NiceMatrix/pNiceArray` (but not by `NiceMatrix/NiceArray` because, for this set of keys, there is also the options `t`, `c` and `b`).

```

5148 \@@_msg_new:nnn { Unknown~option~for~NiceMatrix }
5149 {
5150   The~option~'\l_keys_key_tl'~is~unknown~for~the~
5151   \@@_full_name_env:. \\\
5152   If~you~go~on,~it~will~be~ignored. \\\
5153   For~a~list~of~the~*principal*~available~options,~type~H~<return>.
5154 }
5155 {
5156   The~available~options~are~(in~alphabetic~order):~
5157   b,~
5158   baseline,~
5159   c,~
5160   cell-space-bottom-limit,~
5161   cell-space-top-limit,~
5162   code-after,~
5163   code-for-first-col,~
5164   code-for-first-row,~
5165   code-for-last-col,~
5166   code-for-last-row,~
5167   colortbl-like,~
5168   columns-width,~
5169   create-extra-nodes,~
5170   create-medium-nodes,~
5171   create-large-nodes,~
5172   extra-left-margin,~
5173   extra-right-margin,~
5174   first-col,~

```

```

5175 first-row,~
5176 hlines,~
5177 hvlines,~
5178 hvlines-except-corners,~
5179 l,~
5180 last-col,~
5181 last-row,~
5182 left-margin,~
5183 light-syntax,~
5184 name,~
5185 nullify-dots,~
5186 r,~
5187 renew-dots,~
5188 right-margin,~
5189 rules/color,~
5190 rules/width,~
5191 small,~
5192 t,~
5193 vlines,~
5194 xdots/color,~
5195 xdots/shorten~and~
5196 xdots/line-style.
5197 }

5198 \@@_msg_new:nnn { Unknown~option~for~NiceTabular }
5199 {
5200   The~option~'\l_keys_key_tl'~is~unknown~for~the~environment~
5201   \{NiceTabular\}. \\
5202   If~you~go~on,~it~will~be~ignored. \\
5203   For~a~list~of~the~*principal*~available~options,~type~H~<return>.
5204 }
5205 {
5206   The~available~options~are~(in~alphabetic~order):~
5207   b,~
5208   baseline,~
5209   c,~
5210   cell-space-bottom-limit,~
5211   cell-space-top-limit,~
5212   code-after,~
5213   code-for-first-col,~
5214   code-for-first-row,~
5215   code-for-last-col,~
5216   code-for-last-row,~
5217   colortbl-like,~
5218   columns-width,~
5219   create-extra-nodes,~
5220   create-medium-nodes,~
5221   create-large-nodes,~
5222   extra-left-margin,~
5223   extra-right-margin,~
5224   first-col,~
5225   first-row,~
5226   hlines,~
5227   hvlines,~
5228   hvlines-except-corners,~
5229   last-col,~
5230   last-row,~
5231   left-margin,~
5232   light-syntax,~
5233   name,~
5234   notes/bottomrule,~
5235   notes/para,~
5236   nullify-dots,~
5237   renew-dots,~

```



```

5238     right-margin,~
5239     rules/color,~
5240     rules/width,~
5241     t,~
5242     vlines,~
5243     xdots/color,~
5244     xdots/shorten-and~
5245     xdots/line-style.
5246 }

5247 \@@_msg_new:nnn { Duplicate-name }
5248 {
5249     The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
5250     the~same~environment~name~twice.~You~can~go~on,~but,~
5251     maybe,~you~will~have~incorrect~results~especially~
5252     if~you~use~'columns-width=auto'.~If~you~don't~want~to~see~this~
5253     message~again,~use~the~option~'allow-duplicate-names'~in~
5254     '\token_to_str:N \NiceMatrixOptions'.\\
5255     For~a~list~of~the~names~already~used,~type~H~<return>. \\
5256 }
5257 {
5258     The~names~already~defined~in~this~document~are:~
5259     \seq_use:Nnnn \g_@@_names_seq { ,~ } { ,~ } { ~and~ }.
5260 }

5261 \@@_msg_new:nn { Option-auto-for-columns-width }
5262 {
5263     You~can't~give~the~value~'auto'~to~the~option~'columns-width'~here.~
5264     If~you~go~on,~the~option~will~be~ignored.
5265 }

```

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 StackExchange⁴⁴, Tikz externalization is now deactivated in the environments of the package `nicematrix`.⁴⁵

Changes between version 2.1.2 and 2.1.3

When searching the end of a dotted line from a command like `\Cdots` issued in the “main matrix” (not in the exterior column), the cells in the exterior column are considered as outside the matrix. That means that it’s possible to do the following matrix with only a `\Cdots` command (and a single `\Vdots`).

$$\left(\begin{array}{ccc} & C_j & \\ 0 & \vdots & 0 \\ & a & \cdots \\ 0 & & 0 \end{array} \right) L_i$$

Changes between version 2.1.3 and 2.1.4

Replacement of some options `0 { }` in commands and environments defined with `xparse` by `! 0 { }` (because a recent version of `xparse` introduced the specifier `!` and modified the default behaviour of the last optional arguments).

See www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end

Changes between version 2.1.4 and 2.1.5

Compatibility with the classes `revtex4-1` and `revtex4-2`.

Option `allow-duplicate-names`.

⁴⁴cf. tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package

⁴⁵Before this version, there was an error when using `nicematrix` with Tikz externalization. In any case, it’s not possible to externalize the Tikz elements constructed by `nicematrix` because they use the options `overlay` and `remember picture`.

Changes between version 2.1.5 and 2.2

Possibility to draw horizontal dotted lines to separate rows with the command `\hdottedline` (similar to the classical command `\hline` and the command `\hdashline` of `arydshln`).

Possibility to draw vertical dotted lines to separate columns with the specifier “:” in the preamble (similar to the classical specifier “|” and the specifier “:” of `arydshln`).

Changes between version 2.2 and 2.2.1

Improvement of the vertical dotted lines drawn by the specifier “:” in the preamble.

Modification of the position of the dotted lines drawn by `\hdottedline`.

Changes between version 2.2.1 and 2.3

Compatibility with the column type `S` of `siunitx`.

Option `hlines`.

Changes between version 2.3 and 3.0

Modification of `\Hdotsfor`. Now `\Hdotsfor` erases the `\vlines` (of “|”) as `\hdotsfor` does.

Composition of exterior rows and columns on the four sides of the matrix (and not only on two sides) with the options `first-row`, `last-row`, `first-col` and `last-col`.

Changes between version 3.0 and 3.1

Command `\Block` to draw block matrices.

Error message when the user gives an incorrect value for `last-row`.

A dotted line can no longer cross another dotted line (excepted the dotted lines drawn by `\cdottedline`, the symbol “:” (in the preamble of the array) and `\line` in `code-after`).

The starred versions of `\Cdots`, `\Ldots`, etc. are now deprecated because, with the new implementation, they become pointless. These starred versions are no longer documented.

The vertical rules in the matrices (drawn by “|”) are now compatible with the color fixed by `colortbl`.

Correction of a bug: it was not possible to use the colon “:” in the preamble of an array when `pdflatex` was used with `french-babel` (because `french-babel` activates the colon in the beginning of the document).

Changes between version 3.1 and 3.2 (and 3.2a)

Option `small`.

Changes between version 3.2 and 3.3

The options `first-row`, `last-row`, `first-col` and `last-col` are now available in the environments `{NiceMatrix}`, `{pNiceMatrix}`, `{bNiceMatrix}`, etc.

The option `columns-width=auto` doesn’t need any more a second compilation.

The options `renew-dots`, `renew-matrix` and `transparent` are now available as package options (as said in the documentation).

The previous version of `nicematrix` was incompatible with a recent version of `expl3` (released 2019/09/30). This version is compatible.

Changes between version 3.3 and 3.4

Following a discussion on TeX StackExchange⁴⁶, optimization of Tikz externalization is disabled in the environments of `nicematrix` when the class `standalone` or the package `standalone` is used.

⁴⁶cf. tex.stackexchange.com/questions/510841/nicematrix-and-tikz-external-optimize

Changes between version 3.4 and 3.5

Correction on a bug on the two previous versions where the `code-after` was not executed.

Changes between version 3.5 and 3.6

LaTeX counters `iRow` and `jCol` available in the cells of the array.

Addition of `\normalbaselines` before the construction of the array: in environments like `{align}` of `amsmath` the value of `\baselineskip` is changed and if the options `first-row` and `last-row` were used in an environment of `nicematrix`, the position of the delimiters was wrong.

A warning is written in the `.log` file if an obsolete environment is used.

There is no longer artificial errors `Duplicate-name` in the environments of `amsmath`.

Changes between version 3.6 and 3.7

The four “corners” of the matrix are correctly protected against the four codes: `code-for-first-col`, `code-for-last-col`, `code-for-first-row` and `code-for-last-row`.

New command `\pAutoNiceMatrix` and its variants (suggestion of Christophe Bal).

Changes between version 3.7 and 3.8

New programming for the command `\Block` when the block has only one row. With this programming, the vertical rules drawn by the specifier “|” at the end of the block is actually drawn. In previous versions, they were not because the block of one row was constructed with `\multicolumn`. An error is raised when an obsolete environment is used.

Changes between version 3.8 and 3.9

New commands `\NiceMatrixLastEnv` and `\OnlyMainNiceMatrix`.

New options `create-medium-nodes` and `create-large-nodes`.

Changes between version 3.9 and 3.10

New option `light-syntax` (and `end-of-row`).

New option `dotted-lines-margin` for fine tuning of the dotted lines.

Changes between versions 3.10 and 3.11

Correction of a bug linked to `first-row` and `last-row`.

Changes between versions 3.11 and 3.12

Command `\rotate` in the cells of the array.

Options `vlines`, `hlines` and `hvlines`.

Option `baseline` pour `{NiceArray}` (not for the other environments).

The name of the Tikz nodes created by the command `\Block` has changed: when the command has been issued in the cell $i-j$, the name is $i-j$ -block and, if the creation of the “medium nodes” is required, a node $i-j$ -block-medium is created.

If the user try to use more columns than allowed by its environment, an error is raised by `nicematrix` (instead of a low-level error).

The package must be loaded with the option `obsolete-environments` if we want to use the deprecated environments.

Changes between versions 3.12 and 3.13

The behaviour of the command `\rotate` is improved when used in the “last row”.

The option `dotted-lines-margin` has been renamed in `xdots/shorten` and the options `xdots/color` and `xdots/line-style` have been added for a complete customization of the dotted lines.

In the environments without preamble (`{NiceMatrix}`, `{pNiceMatrix}`, etc.), it’s possible to use the options `l` (`=L`) or `r` (`=R`) to specify the type of the columns.

The starred versions of the commands `\Cdots`, `\Ldots`, `\Vdots`, `\Ddots` and `\Iddots` are deprecated since the version 3.1 of `nicematrix`. Now, one should load `nicematrix` with the option `starred-commands` to avoid an error at the compilation.

The code of `nicematrix` no longer uses Tikz but only PGF. By default, Tikz is *not* loaded by `nicematrix`.

Changes between versions 3.13 and 3.14

Correction of a bug (question 60761504 on [stackoverflow](#)).

Better error messages when the user uses `&` or `\\` when `light-syntax` is in force.

Changes between versions 3.14 and 3.15

It’s possible to put labels on the dotted lines drawn by `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots`, `\Hdotsfor` and the command `\line` in the `code-after` with the tokens `_` and `^`.

The option `baseline` is now available in all the environments of `nicematrix`. Before, it was available only in `{NiceArray}`.

New keyword `\CodeAfter` (in the environments of `nicematrix`).

Changes between versions 3.15 and 4.0

New environment `{NiceTabular}`

Commands to color cells, row and columns with a perfect result in the PDF.

Changes between versions 4.0 and 4.1

New keys `cell-space-top-limit` and `cell-space-bottom-limit`

New command `\diagbox`

The key `hvline` don’t draw rules in the blocks (commands `\Block`) and in the virtual blocks corresponding to the dotted lines.

Changes between versions 4.1 and 4.2

It’s now possible to write `\begin{pNiceMatrix}a&b\\c&d\end{pNiceMatrix}^2` 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` can take in as value 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.

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>\@@_Block:</code> 1057, 4282	<code>\@@_Hdotsfor_i</code> 3102, 3109
<code>\@@_Block_i</code> 4283, 4284	<code>\@@_Hline:</code> 1052, 3826
<code>\@@_Block_ii:nnnnn</code> 4284, 4285	<code>\@@_Hline_i:n</code> 3826, 3827, 3833
<code>\@@_Block_iii:nnnnnn</code> 4431, 4432	<code>\@@_Hline_ii:nn</code> 3830, 3833
<code>\@@_Cdots</code> 982, 1047, 2961	<code>\@@_Hline_iii:n</code> 3831, 3834
<code>\g_@@_Cdots_lines_tl</code> 1073, 2292	<code>\@@_Hspace:</code> 1053, 3044
<code>\@@_Cell:</code> 206, 761, 1471, 1518, 1535, 2092, 3061, 3062, 3063, 3064, 3065, 3066, 3067, 3068, 3069, 3070, 3071, 3072	<code>\@@_Iddots</code> 985, 1050, 3014
<code>\@@_CodeAfter:n</code> 1061, 4726, 4737	<code>\g_@@_Iddots_lines_tl</code> 1077, 2291
<code>\@@_CodeAfter_i:n</code> 4729, 4731	<code>\@@_Ldots</code> 981, 986, 1046, 2946
<code>\@@_Ddots</code> 984, 1049, 2991	<code>\g_@@_Ldots_lines_tl</code> 1074, 2293
<code>\g_@@_Ddots_lines_tl</code> 1076, 2290	<code>\l_@@_Matrix_bool</code> 250, 1278, 1384, 2083
<code>\g_@@_HVDotsfor_lines_tl</code> 1078, 2288, 3111, 3187, 4845	<code>\l_@@_NiceArray_bool</code> 247, 343, 1119, 1230, 1303, 1415, 1427, 2059, 3705, 3706, 3822, 3823, 4003, 4010, 4659
<code>\@@_Hdotsfor:</code> 987, 1054, 3096	<code>\g_@@_NiceMatrixBlock_int</code> 240, 4064, 4069, 4072, 4083
<code>\@@_Hdotsfor:nnnn</code> 3113, 3125	<code>\l_@@_NiceTabular_bool</code> 162, 248, 768, 918, 1095, 1174, 1176, 1335, 1339, 1416, 1428, 2110, 2119, 4320, 4384, 4667

\@@_OnlyMainNiceMatrix:n	1059, 3568	\l_@@_code_for_last_row_tl	511, 784, 4531
\@@_OnlyMainNiceMatrix_i:n	3571, 3578, 3581	\g_@@_col_total_int	766, 1069, 1271, 1356, 1365, 1915, 1916, 1952, 1956, 1961, 1962, 2018, 2127, 2130, 2135, 2142, 2186, 2629, 3092, 3093, 3246, 4100, 4110, 4144, 4231, 4457
\@@_Vdots	983, 1048, 2976	\l_@@_color_tl	290, 4420, 4436, 4441
\g_@@_Vdots_lines_tl	1075, 2289	\@@_colortbl_like:	989, 1063
\@@_Vdotsfor:	1055, 3185	\l_@@_colortbl_like_bool	423, 582, 1063, 1403
\@@_Vdotsfor:nnnn	3189, 3200	\c_@@_colortbl_loaded_bool	94, 98, 1014
\@@_W:	1388, 1457	\@@_columncolor	1172, 3353
\@@_actually_diagbox:nnnnnn	4306, 4450, 4679, 4695	\@@_columncolor_preamble	993, 3559
\@@_actually_draw_Cdots:	2539, 2543	\c_@@_columncolor_regex	215, 1406
\@@_actually_draw_Ddots:	2665, 2669	\l_@@_columns_width_dim	241, 567, 689, 1884, 1890, 4065, 4071
\@@_actually_draw_Iddots:	2716, 2720	\g_@@_com_or_env_str	261, 264
\@@_actually_draw_Ldots:	2497, 2501, 3176	\@@_computations_for_large_nodes:	4171, 4184, 4189
\@@_actually_draw_Vdots:	2592, 2596, 3251	\@@_computations_for_medium_nodes:	4091, 4160, 4170, 4181
\@@_adapt_S_column:	176, 191, 1094	\@@_compute_a_corner:nnnnnn	3875, 3877, 3879, 3881, 3886
\@@_adjust_width_box:	839, 853, 1544, 1995, 2039	\@@_compute_corners:	2223, 3867
\@@_after_array:	1377, 2123	\@@_construct_preamble:n	1243, 1381
\@@_analyze_end:Nn	1771, 1816	\@@_convert_to_str_seq:N	4811, 4823
\l_@@_argspec_tl	2944, 2945, 2946, 2961, 2976, 2991, 3014, 3107, 3108, 3109, 3183, 3184, 3185, 3261, 3262, 3263	\@@_create_col_nodes:	1775, 1803, 1822
\@@_array:	913, 1772, 1799	\@@_create_large_nodes:	2209, 4165
\l_@@_auto_columns_width_bool	448, 566, 1883, 1887, 4059	\@@_create_medium_and_large_nodes:	2206, 4176
\l_@@_baseline_str	437, 438, 559, 560, 561, 562, 926, 1305, 1587, 1599, 1604, 1606, 1611, 1616, 1695, 1696, 1700, 1705, 1707, 1712	\@@_create_medium_nodes:	2207, 4155
\@@_begin_of_NiceMatrix:nn	2081, 2102	\@@_create_nodes:	4162, 4173, 4183, 4186, 4227
\@@_begin_of_row:	765, 789, 1973	\@@_create_row_node:	929, 961, 998
\l_@@_block_auto_columns_width_bool	1108, 1888, 4052, 4057, 4067, 4077	\@@_cut_on_hyphen:w	3312, 3334, 3335, 3368, 3369, 3398, 3429, 3440
\g_@@_block_box_int	294, 1088, 4301, 4315, 4317, 4353, 4363, 4372	\g_@@_ddots_int	2196, 2689, 2690
\g_@@_blocks_seq	286, 1110, 2237, 4366, 4377, 4431	\@@_def_env:nnn	2065, 2076, 2077, 2078, 2079, 2080
\g_@@_blocks_width_dim	244, 841, 844, 845, 4357, 4360	\@@_define_L_C_R:	228, 1242
\c_@@_booktabs_loaded_bool	36, 42, 997, 1664	\c_@@_define_L_C_R_bool	227, 1242, 4759
\l_@@_cell_box	767, 813, 815, 821, 827, 830, 834, 843, 844, 854, 855, 856, 857, 859, 862, 864, 866, 883, 999, 1185, 1187, 1534, 1545, 1974, 1998, 2001, 2003, 2019, 2042, 2046, 4460, 4556, 4587, 4674	\@@_define_com:nnn	4643, 4651, 4652, 4653, 4654, 4655
\l_@@_cell_space_bottom_limit_dim	426, 493, 857	\g_@@_delta_x_one_dim	2198, 2692, 2702
\l_@@_cell_space_top_limit_dim	425, 491, 855	\g_@@_delta_x_two_dim	2200, 2743, 2753
\l_@@_cell_type_tl	242, 243, 1471, 1536, 4287, 4289	\g_@@_delta_y_one_dim	2199, 2694, 2702
\@@_cellcolor	1168, 3389, 3539, 3540	\g_@@_delta_y_two_dim	2201, 2745, 2753
\@@_cellcolor_tabular	991, 3545	\@@_diagbox:nn	1062, 4675
\g_@@_cells_seq	1810, 1811, 1812, 1814	\@@_dotfill:	4664
\@@_chessboardcolors	1173, 3532	\@@_dotfill_i:	4669, 4671
\@@_cline	141, 1045	\@@_dotfill_ii:	4668, 4671, 4672
\@@_cline_i:nn	142, 143, 155, 158	\@@_dotfill_iii:	4672, 4673
\@@_cline_i:w	143, 144	\@@_double_int_eval:n	3257, 3271, 3272
\l_@@_code_before_bool	275, 556, 583, 933, 1115, 1125, 1830, 1847, 1865, 1896, 1922, 1949, 2169, 2269	\g_@@_dp_ante_last_row_dim	792, 1030
\l_@@_code_before_tl	274, 555, 1116, 1175	\g_@@_dp_last_row_dim	792, 793, 1033, 1034, 1186, 1187, 1322
\l_@@_code_for_first_col_tl	505, 1985	\g_@@_dp_row_zero_dim	812, 813, 1024, 1025, 1315, 1689, 1728
\l_@@_code_for_first_row_tl	509, 777, 4528	\@@_draw_Cdots:nnn	2525
\l_@@_code_for_last_col_tl	507, 2028	\@@_draw_Ddots:nnn	2657
		\@@_draw_Iddots:nnn	2708
		\@@_draw_Ldots:nnn	2483
		\@@_draw_Vdots:nnn	2577
		\@@_draw_blocks:	2237, 4430
		\@@_draw_dotted_lines:	2222, 2277
		\@@_draw_dotted_lines_i:	2280, 2284

\l_@@_draw_first_bool .	293, 3006, 3029, 3040	\l_@@_final_i_int	2212, 2305, 2310, 2313, 2338,
\@@_draw_hlines:	2234, 3819		2346, 2350, 2359, 2367, 2447, 2477, 2517,
\@@_draw_line:	2523,		2614, 2681, 2732, 3130, 3158, 3226, 3236, 3238
	2575, 2655, 2706, 2757, 2759, 3310, 4020, 4050	\l_@@_final_j_int	2213, 2306, 2311, 2318, 2323, 2329, 2339,
\@@_draw_line_ii:nn	3290, 3294		2347, 2351, 2360, 2368, 2448, 2478, 2514,
\@@_draw_line_iii:nn	3297, 3301		2554, 2683, 2734, 3151, 3161, 3163, 3205, 3234
\@@_draw_non_standard_dotted_line: . .	2765, 2767	\l_@@_final_open_bool	2215, 2312,
\@@_draw_non_standard_dotted_line:n . .	2770, 2773		2316, 2319, 2326, 2332, 2336, 2352, 2512,
\@@_draw_standard_dotted_line: .	2764, 2793		2552, 2561, 2572, 2599, 2612, 2620, 2641,
\@@_draw_standard_dotted_line_i: .	2858, 2862		2679, 2730, 2866, 2881, 2912, 2913, 3128,
\@@_draw_vlines:	2235, 3702		3152, 3164, 3203, 3227, 3239, 3280, 3986, 4025
\g_@@_empty_cell_bool	282, 861, 868,	\@@_find_extremities_of_line:nnnn . . .	2300, 2487, 2529, 2581, 2661, 2712
	2008, 2054, 2959, 2974, 2989, 3012, 3035, 3046	\l_@@_first_col_int	129, 142,
\l_@@_empty_corner_cells_seq	2228,		297, 298, 501, 739, 765, 1285, 1410, 1825,
	3640, 3646, 3653, 3759, 3765, 3772, 3869, 3942		1845, 2180, 3325, 3376, 3408, 3437, 3570,
\@@_end_Cell:	208, 848, 1473,		4100, 4110, 4144, 4192, 4231, 4625, 4631, 4637
	1524, 1540, 2092, 3061, 3062, 3063, 3064,	\l_@@_first_row_int	295, 296, 502, 743, 1067,
	3065, 3066, 3067, 3068, 3069, 3070, 3071, 3072		1313, 1618, 1686, 1714, 1725, 2178, 4093,
\l_@@_end_of_row_tl	456, 457, 499, 1795, 1796, 4928		4107, 4134, 4191, 4229, 4480, 4498, 4623, 4939
\c_@@_endpgfortikzpicture_tl	51, 55, 2281, 3298, 3989	\c_@@_footnote_bool	1084, 1379, 4741, 4767, 4790, 4809
\c_@@_enumitem_loaded_bool	37, 45, 316, 610, 615, 626, 631	\c_@@_footnotehyper_bool	4740, 4768, 4800
\@@_env:	235, 239,	\@@_full_name_env:	262, 4851, 4858, 4866, 4874,
	798, 804, 884, 890, 938, 944, 950, 1139,		4878, 4944, 4965, 4984, 4991, 5010, 5019, 5151
	1140, 1146, 1147, 1154, 1155, 1165, 1831,	\@@_hdottedline:	1051, 3971
	1834, 1836, 1852, 1858, 1861, 1870, 1876,	\@@_hdottedline:n	3979, 3983
	1879, 1901, 1907, 1910, 1927, 1933, 1939,	\@@_hdottedline_i:	3974, 3976
	1952, 1956, 1962, 2248, 2358, 2426, 2465,	\@@_hdottedline_i:n	3988, 3992
	2476, 3139, 3157, 3214, 3232, 3283, 3285,	\@@_hline:nn	3709, 3824, 3842
	3304, 3307, 3897, 3916, 3934, 4113, 4115,	\@@_hline_i:nn	2232, 3712, 3715
	4123, 4234, 4243, 4261, 4476, 4483, 4487,	\@@_hline_i_complete:nn	2232, 3817
	4501, 4505, 4516, 4521, 4522, 4523, 4535, 4566	\@@_hline_ii:nnnn	3735, 3746, 3779, 3818
\g_@@_env_int	234, 235, 237, 1107,	\l_@@_hlines_bool	443, 513, 518, 548, 962, 2234
	1113, 1117, 1127, 1131, 1134, 1143, 1144,	\g_@@_ht_last_row_dim	794, 1031, 1032, 1184, 1185, 1321
	1151, 1152, 1195, 1198, 1213, 1216, 2134,	\g_@@_ht_row_one_dim	820, 821, 1028, 1029
	2155, 2173, 2176, 2189, 2265, 3468, 3483, 4270	\g_@@_ht_row_zero_dim	814, 815, 1026, 1027, 1316, 1688, 1727
\@@_error:n	24, 319, 344, 468,	\@@_i:	4093, 4095,
	478, 636, 677, 688, 697, 702, 720, 727, 735,		4096, 4097, 4098, 4107, 4113, 4115, 4116,
	741, 746, 757, 759, 1266, 1276, 1281, 1621,		4117, 4118, 4123, 4124, 4125, 4126, 4134,
	1669, 1717, 3099, 3461, 4428, 4769, 4796, 4806		4137, 4139, 4140, 4141, 4193, 4195, 4198,
\@@_error:nn	25, 574,		4199, 4203, 4204, 4229, 4234, 4236, 4238,
	2949, 2952, 2964, 2967, 2979, 2982, 2995,		4242, 4243, 4254, 4261, 4263, 4265, 4269, 4270
	2996, 3001, 3002, 3018, 3019, 3024, 3025, 3883	\g_@@_iddots_int	2197, 2740, 2741
\@@_error:nnn	26, 3288	\l_@@_in_env_bool	246, 343, 1098, 1099
\@@_error_too_much_cols:	1438, 4830	\c_@@_in_preamble_bool	33, 34, 35, 606, 622
\@@_everycr:	955, 1019, 1022	\@@_info:n	4803
\@@_everycr_i:	955, 956	\l_@@_initial_i_int	2210,
\l_@@_except_corners_clist	444, 542, 546, 3606, 3726, 3870		2303, 2378, 2381, 2406, 2414, 2418, 2427,
\l_@@_exterior_arraycolsep_bool	439, 685, 1418, 1430		2435, 2445, 2466, 2508, 2563, 2565, 2608,
\l_@@_extra_left_margin_dim	454, 534, 1246, 2006		2673, 2724, 3129, 3130, 3140, 3208, 3218, 3220
\l_@@_extra_right_margin_dim	455, 535, 1258, 2050, 2632	\l_@@_initial_j_int	2211, 2304, 2379, 2386,
\@@_extract_coords_values:	4252, 4259		2391, 2397, 2407, 2415, 2419, 2428, 2436,
\@@_fatal:n	27, 255,		2446, 2467, 2505, 2547, 2622, 2624, 2629,
	1098, 1780, 1784, 1786, 1819, 4835, 4838, 4841		2675, 2726, 3133, 3143, 3145, 3204, 3205, 3216
\@@_fatal:nn	28, 1464		

<code>\l_@@_initial_open_bool</code>	<code>\l_@@_max_delimiter_width_bool</code>
.... 2214, 2380, 2384, 2387, 2394, 2400, 459, 496, 1349
2404, 2420, 2503, 2545, 2560, 2570, 2599,	<code>\c_@@_max_l_dim</code>
2606, 2618, 2671, 2722, 2864, 2911, 3127,	2856, 2861
3134, 3146, 3202, 3209, 3221, 3279, 3985, 4024	<code>\l_@@_medium_nodes_bool</code> 450, 524, 2203, 4518
<code>\@@_instruction_of_type:nnn</code>	<code>\@@_message_hdotsfor:</code> 4843, 4851, 4858, 4866
..... 894, 2954, 2969, 2984, 3006, 3029	<code>\@@_msg_new:nn</code>
<code>\l_@@_inter_dots_dim</code> 29, 4772, 4781, 4848, 4855, 4863,
. 427, 428, 2219, 2869, 2876, 2887, 2895,	4871, 4876, 4883, 4888, 4893, 4898, 4903,
2902, 2907, 2919, 2927, 4015, 4018, 4046, 4048	4908, 4914, 4920, 4925, 4931, 4936, 4943,
<code>\g_@@_internal_code_after_tl</code> 269, 1507,	4945, 4951, 4956, 4963, 4970, 4977, 4982,
1561, 2238, 2239, 3841, 3978, 4304, 4448, 4677	4988, 4995, 5001, 5007, 5015, 5017, 5023, 5261
<code>\@@_intersect_our_row:nnnn</code>	<code>\@@_msg_new:nnn</code>
3521	30,
<code>\@@_intersect_our_row_p:nnnn</code>	4742, 5028, 5038, 5056, 5098, 5148, 5198, 5247
3491	<code>\@@_msg_redirect_name:nn</code>
<code>\@@_j:</code>	31, 691
4100, 4102,	<code>\@@_multicolumn:nnn</code>
4103, 4104, 4105, 4110, 4113, 4115, 4118,	1056, 3050
4120, 4121, 4123, 4126, 4128, 4129, 4144,	<code>\g_@@_multicolumn_cells_seq</code>
4147, 4149, 4150, 4151, 4206, 4208, 4211,	... 1065, 3080, 4118, 4126, 4248, 4485, 4503
4213, 4217, 4218, 4231, 4234, 4235, 4237,	<code>\g_@@_multicolumn_sizes_seq</code> 1066, 3082, 4249
4242, 4243, 4255, 4261, 4262, 4264, 4269, 4270	<code>\g_@@_name_env_str</code>
<code>\l_@@_l_dim</code>	260,
.... 2842, 2843, 2856, 2857, 2869, 2875,	265, 266, 1092, 1093, 1818, 2060, 2061,
2886, 2894, 2902, 2907, 2919, 2920, 2927, 2928	2069, 2070, 2099, 2108, 2116, 2271, 4647, 4832
<code>\l_@@_large_nodes_bool</code> 451, 525, 2205, 2209	<code>\l_@@_name_str</code>
<code>\g_@@_last_col_found_bool</code>	449, 576, 800, 803,
305,	886, 889, 946, 949, 1193, 1202, 1205, 1211,
1072, 1272, 1353, 1914, 1943, 2016, 2126, 2183	1220, 1223, 1835, 1836, 1860, 1861, 1878,
<code>\l_@@_last_col_int</code>	1879, 1909, 1910, 1935, 1938, 1958, 1961,
303,	2137, 2141, 2158, 2162, 4239, 4242, 4266, 4269
304, 678, 713, 715, 728, 742, 758, 1137,	<code>\g_@@_names_seq</code>
1209, 1215, 1222, 1275, 1422, 2088, 2090,	245, 573, 575, 5259
2127, 2130, 2182, 2586, 2627, 2951, 2966,	<code>\l_@@_nb_cols_int</code>
3002, 3025, 4561, 4627, 4633, 4639, 4834, 4852 4614, 4619, 4622, 4626, 4632, 4638
<code>\l_@@_last_col_without_value_bool</code> ...	<code>\l_@@_nb_rows_int</code>
..... 302, 712, 2128, 4837	4613, 4618, 4629
<code>\l_@@_last_empty_column_int</code>	<code>\@@_newcolumntype</code>
..... 3907, 3908, 3921, 3927, 3940	972, 1387, 1388
<code>\l_@@_last_empty_row_int</code>	<code>\@@_node_for_multicolumn:nn</code> 4250, 4257
..... 3889, 3890, 3903, 3924	<code>\@@_node_for_the_cell:</code> 865, 870, 2002, 2051
<code>\l_@@_last_row_int</code> .. 299, 300, 503, 782,	<code>\@@_node_position:</code> .. 1146, 1148, 1154, 1156
828, 966, 1135, 1180, 1190, 1197, 1204,	<code>\@@_not_in_exterior:nnnn</code>
1260, 1264, 1267, 1284, 1319, 1797, 1798,	3513
1981, 1982, 2025, 2026, 2149, 2492, 2534,	<code>\@@_not_in_exterior_p:nnnn</code>
2981, 2996, 3019, 3576, 3584, 4530, 4635, 5009	3485
<code>\l_@@_last_row_without_value_bool</code> ...	<code>\l_@@_notes_above_space_dim</code> 445, 446
..... 301, 1192, 1262, 2147	<code>\l_@@_notes_bottomrule_bool</code>
<code>\l_@@_left_delim_dim</code> 594, 731, 752, 1662
..... 1228, 1232, 1237, 1763, 2004	<code>\l_@@_notes_code_after_tl</code>
<code>\l_@@_left_delim_tl</code>	592, 1671
1086, 4014	<code>\l_@@_notes_code_before_tl</code>
<code>\l_@@_left_margin_dim</code>	590, 1643
..... 452, 528, 1245, 2005, 4004, 4222	<code>\@@_notes_label_in_list:n</code> 312, 331, 339, 602
<code>\l_@@_letter_for_dotted_lines_str</code> ...	<code>\@@_notes_label_in_tabular:n</code> . 311, 352, 599
..... 696, 704, 705, 1462	<code>\l_@@_notes_para_bool</code> .. 588, 729, 750, 1647
<code>\l_@@_light_syntax_bool</code>	<code>\@@_notes_style:n</code>
..... 436, 497, 1248, 1253, 2150 310, 313, 331, 339, 355, 360, 596
<code>\@@_light_syntax_i</code>	<code>\l_@@_nullify_dots_bool</code>
1788, 1791 447, 523, 2958, 2973, 2988, 3011, 3034
<code>\@@_line</code>	<code>\l_@@_number_of_notes_int</code> 309, 346, 356, 366
2252, 3263	<code>\@@_old_CT@arc@</code>
<code>\@@_line_i:nn</code>	1100, 2273
3270, 3277	<code>\@@_old_arraycolsep_dim</code>
<code>\@@_line_with_light_syntax:n</code> ... 1802, 1806	283
<code>\@@_line_with_light_syntax_i:n</code>	<code>\@@_old_cdots</code>
..... 1801, 1807, 1808	1039, 2973
<code>\@@_math_toggle_token:</code>	<code>\@@_old_ddots</code>
161, 850, 1975, 1992, 2020, 2036, 4719, 4723	1041, 3011
<code>\g_@@_max_cell_width_dim</code>	<code>\@@_old_dotfill</code>
..... 858, 859, 1109, 1889, 4058, 4084	4663, 4666, 4674
	<code>\@@_old_dotfill:</code>
	1060
	<code>\l_@@_old_iRow_int</code>
	270, 1001, 2297
	<code>\@@_old_ialign:</code>
	928, 1035, 2236
	<code>\@@_old_iddots</code>
	1042, 3034
	<code>\l_@@_old_jCol_int</code>
	271, 1004, 2298
	<code>\@@_old_ldots</code>
	1038, 2958
	<code>\@@_old_multicolumn</code>
	3049, 3056
	<code>\@@_old_pgful@check@rerun</code>
	87, 91
	<code>\@@_old_vdots</code>
	1040, 2988

\l_@@_parallelize_diags_bool	440, 441, 520, 2194, 2687, 2738	\\@@_renew_matrix:	681, 4593, 4763
\\@@_patch_preamble:n	1400, 1442, 1481, 1510, 1563, 1575	\\l_@@_respect_blocks_bool	3459, 3467
\\@@_patch_preamble_i:n	1446, 1447, 1448, 1467	\\@@_restore_iRow_jCol:	2272, 2295
\\@@_patch_preamble_ii:nn	1449, 1450, 1451, 1478	\\@@_revtex_array:	905, 916
\\@@_patch_preamble_iii:n	1452, 1483, 1491	\\c_@@_revtex_bool	58, 60, 63, 915
\\@@_patch_preamble_iii_i:n	1486, 1488	\\l_@@_right_delim_dim	1229, 1233, 1239, 1766, 2048
\\@@_patch_preamble_iv:nnn	1453, 1454, 1455, 1513	\\l_@@_right_delim_tl	1087, 4017
\\@@_patch_preamble_ix:n	1568, 1578	\\l_@@_right_margin_dim	453, 530, 1257, 2049, 2631, 4011, 4225
\\@@_patch_preamble_v:nnnn	1456, 1457, 1529	\\@@_rotate:	1058, 3256
\\@@_patch_preamble_vi:n	1458, 1551	\\g_@@_rotate_bool	251, 837, 852, 1543, 1994, 2038, 3256, 4326, 4339, 4350, 4355, 4390, 4403, 4461
\\@@_patch_preamble_vii:n	1463, 1557	\\@@_rotate_cell_box:	825, 852, 1543, 1994, 2038, 4461
\\@@_patch_preamble_viii:n	1476, 1527, 1549, 1555, 1565, 1581	\\g_@@_row_of_col_done_bool	273, 959, 1091, 1844
\\@@_pgf_rect_node:nnn	399, 4520	\\g_@@_row_total_int	1068, 1283, 1619, 1715, 2149, 2156, 2163, 2179, 3171, 4093, 4107, 4134, 4229, 4456, 4480, 4498, 4940
\\@@_pgf_rect_node:nnnn	374, 4233, 4260, 4475, 4515	\\@@_rowcolor	1170, 3317, 3474, 3475, 3496, 3501
\\c_@@_pgfortikzpicture_tl	50, 54, 2279, 3296, 3987	\\@@_rowcolor_tabular	992, 3550
\\@@_picture_position:	1140, 1148, 1156	\\@@_rowcolors	1171, 3463
\\l_@@_pos_of_block_tl ...	291, 292, 4275, 4277, 4279, 4288, 4289, 4327, 4329, 4340, 4343, 4369, 4391, 4393, 4404, 4407, 4422, 4424, 4426, 4537, 4549, 4560, 4562, 4568, 4580	\\@@_rowcolors_i:nnnn	3469, 3480
\\g_@@_pos_of_blocks_seq	287, 1111, 2190, 2226, 3083, 3602, 3722, 3954, 4298, 4687	\\@@_rowcolors_ii:nnnn	3493, 3508
\\g_@@_pos_of_xdots_seq	288, 1112, 2227, 2443, 3604, 3724	\\g_@@_rows_seq .	1794, 1796, 1798, 1800, 1802
\\@@_pre_array:	995, 1227	\\l_@@_rules_color_tl ..	272, 482, 1123, 1124
\\c_@@_preamble_first_col_tl	1411, 1969	\\@@_set_CT@arc@:	163, 1124
\\c_@@_preamble_last_col_tl	1423, 2012	\\@@_set_CT@arc@_i:	164, 165
\\g_@@_preamble_tl	1385, 1395, 1398, 1408, 1411, 1420, 1423, 1432, 1437, 1469, 1480, 1493, 1515, 1531, 1553, 1559, 1572, 1580, 1772, 1799	\\@@_set_CT@arc@_ii:	164, 167
\\@@_pred:n	130, 160, 2090, 3641, 3654, 3760, 3773	\\@@_set_final_coords:	2456, 2481
\\@@_provide_pgfsyspdfmark: .	216, 225, 1083	\\@@_set_final_coords_from_anchor:n ..	2472, 2520, 2558, 2602, 2617, 2686, 2737
\\@@_put_box_in_flow:	1351, 1583, 1765	\\@@_set_initial_coords:	2451, 2470
\\@@_put_box_in_flow_bis:nn	1350, 1732	\\@@_set_initial_coords_from_anchor:n .	2461, 2511, 2551, 2601, 2611, 2678, 2729
\\@@_put_box_in_flow_i:	1589, 1591	\\@@_set_seq_of_str_from_clist:Nn	4820, 4825
\\@@_qpoint:n	238, 1594, 1596, 1608, 1624, 1680, 1682, 1698, 1709, 1720, 2505, 2508, 2514, 2517, 2547, 2554, 2563, 2565, 2608, 2614, 2622, 2624, 2673, 2675, 2681, 2683, 2724, 2726, 2732, 2734, 3304, 3307, 3324, 3328, 3341, 3343, 3360, 3362, 3375, 3379, 3399, 3405, 3407, 3411, 3434, 3436, 3445, 3447, 3664, 3666, 3668, 3783, 3785, 3787, 3995, 3999, 4006, 4042, 4045, 4047, 4139, 4149, 4465, 4467, 4469, 4471, 4494, 4512, 4533, 4700, 4702, 4705, 4707	\\@@_set_size:n	4611, 4620
\\l_@@_radius_dim	431, 432, 1560, 2218, 2521, 2522, 2936, 3973, 3997, 4043, 4044	\\c_@@_siunitx_loaded_bool	169, 173, 178, 196
\\l_@@_real_left_delim_dim	1734, 1749, 1764	\\l_@@_small_bool	679, 718, 724, 744, 771, 1007, 1976, 2021, 2216
\\l_@@_real_right_delim_dim	1735, 1761, 1767	\\@@_standard_cline	126, 1044
\\@@_rectanglecolor	1169, 3422	\\@@_standard_cline:w	126, 127
\\@@_renew_NC@rewrite@S:	197, 199, 1071	\\l_@@_standard_cline_bool ..	424, 489, 1043
\\@@_renew_dots:	979, 1064	\\c_@@_standard_tl	434, 435, 2763, 4019, 4049
\\l_@@_renew_dots_bool	521, 1064, 4761	\\g_@@_static_num_of_col_int	289, 1280, 1401, 4867, 4879
		\\l_@@_stop_loop_bool	2307, 2308, 2340, 2353, 2362, 2375, 2376, 2408, 2421, 2430
		\\@@_succ:n	155, 159, 938, 944, 1508, 1596, 1927, 1933, 1938, 1939, 1952, 1956, 1961, 1962, 2184, 2514, 2554, 2565, 2614, 2624, 2681, 2683, 2726, 2732, 3328, 3341, 3362, 3379, 3405, 3411, 3445, 3447, 3517, 3637, 3668, 3706, 3756, 3787, 3823, 3842, 3959, 3961, 3963, 3965, 4006, 4047, 4199, 4203, 4213, 4217, 4469, 4471, 4512, 4705, 4707
		\\l_@@_suffix_tl	4161, 4172, 4182, 4185, 4234, 4242, 4243, 4261, 4269, 4270
		\\c_@@_table_collect_begin_tl .	186, 188, 206
		\\c_@@_table_print_tl	189, 190, 208

<code>\l_@@_tabular_width_dim</code>	249, 921, 923, 1434, 2117
<code>\l_@@_tabularnote_tl</code>	308, 733, 754, 1636, 1644
<code>\g_@@_tabularnotes_seq</code>	307, 347, 1650, 1656, 1673
<code>\@@_test_if_cell_in_a_block:nn</code>	3893, 3911, 3929, 3949
<code>\@@_test_if_cell_in_block:nnnnnnn</code> ...	3955, 3957
<code>\@@_test_if_hline_in_block:nnnn</code>	3723, 3725, 3845
<code>\@@_test_if_math_mode:</code>	252, 1097, 2071
<code>\@@_test_if_vline_in_block:nnnn</code>	3603, 3605, 3856
<code>\@@_test_in_corner_h:</code>	3726, 3754
<code>\@@_test_in_corner_v:</code>	3607, 3635
<code>\l_@@_the_array_box</code>	1241, 1244, 1637, 1639, 1640
<code>\c_@@_tikz_loaded_bool</code>	38, 49, 1160, 2240, 4026
<code>\l_@@_tikz_tl</code>	4418, 4474
<code>\@@_true_c:</code>	207, 1458
<code>\l_@@_type_of_col_tl</code> ..	716, 717, 2100, 2102
<code>\c_@@_types_of_matrix_seq</code>	4825, 4832
<code>\@@_update_for_first_and_last_row:</code> ..	808, 860, 1182, 1996, 2040
<code>\@@_use_arraybox_with_notes:</code> ...	1310, 1693
<code>\@@_use_arraybox_with_notes_b:</code> ..	1307, 1677
<code>\@@_use_arraybox_with_notes_c:</code>	1308, 1338, 1632, 1691, 1730
<code>\@@_vdottedline:n</code>	1562, 4022
<code>\@@_vdottedline_i:n</code>	4029, 4034, 4038
<code>\@@_vline:nn</code>	1508, 3586, 3707
<code>\@@_vline_i:nn</code>	2231, 3591, 3595
<code>\@@_vline_i_complete:nn</code>	2231, 3700
<code>\@@_vline_ii:nnnn</code> ...	3616, 3627, 3660, 3701
<code>\l_@@_vlines_bool</code>	442, 514, 517, 547, 1393, 1417, 1429, 1570, 2235
<code>\@@_w:</code>	1387, 1456
<code>\g_@@_width_first_col_dim</code>	285, 1090, 1297, 1301, 1369, 1839, 1997, 1998
<code>\g_@@_width_last_col_dim</code>	284, 1089, 1373, 1948, 2041, 2042
<code>\l_@@_x_final_dim</code>	278, 2458, 2515, 2516, 2555, 2556, 2604, 2626, 2634, 2638, 2642, 2644, 2649, 2651, 2684, 2693, 2701, 2735, 2744, 2752, 2790, 2804, 2813, 2849, 2901, 2917, 3308, 4007, 4018, 4044
<code>\l_@@_x_initial_dim</code>	276, 2453, 2506, 2507, 2548, 2549, 2604, 2625, 2626, 2633, 2638, 2642, 2644, 2646, 2649, 2651, 2676, 2693, 2701, 2727, 2744, 2752, 2781, 2803, 2813, 2849, 2901, 2915, 2917, 2935, 2937, 3305, 4000, 4015, 4043
<code>\l_@@_xdots_color_tl</code>	458, 471, 2496, 2538, 2590, 2591, 2664, 2715, 2771, 3175, 3250, 3267
<code>\l_@@_xdots_down_tl</code> ...	475, 2787, 2797, 2832
<code>\l_@@_xdots_line_style_tl</code>	433, 435, 467, 2763, 2771, 4019, 4049
<code>\l_@@_xdots_shorten_dim</code>	429, 430, 473, 2220, 2778, 2779, 2875, 2886, 2894
<code>\l_@@_xdots_up_tl</code>	476, 2783, 2796, 2822
<code>\l_@@_y_final_dim</code>	279, 2459, 2518, 2522, 2567, 2571, 2573, 2615, 2682, 2695, 2698, 2733, 2746, 2749, 2790, 2804, 2812, 2851, 2906, 2925, 3309, 3998, 4048
<code>\l_@@_y_initial_dim</code>	277, 2454, 2509, 2521, 2566, 2567, 2571, 2573, 2609, 2674, 2695, 2700, 2725, 2746, 2751, 2781, 2803, 2812, 2851, 2906, 2923, 2925, 2935, 2938, 3306, 3996, 3997, 3998, 4046
<code>\</code>	1785, 1807, 4627, 4633, 4639, 4744, 4745, 4778, 4787, 4880, 4885, 4890, 4895, 4900, 4927, 4933, 4940, 4948, 4960, 4966, 4973, 4980, 4985, 4992, 4998, 5004, 5016, 5020, 5025, 5031, 5040, 5041, 5059, 5060, 5101, 5102, 5151, 5152, 5201, 5202, 5254, 5255
<code>\{</code>	266, 2078, 4655, 4947, 4992, 5101, 5201
<code>\}</code>	266, 2078, 4655, 4947, 4992, 5101, 5201
<code>\ </code>	2080, 4654
<code>_</code>	4846, 4851, 4858, 4866, 4867, 4878, 4879, 4939, 4940, 4944, 4953, 4984, 4990, 5003, 5010, 5011, 5019
A	
<code>\aboverulesep</code>	1666
<code>\addtocounter</code>	364
<code>\alpha</code>	310
<code>\arraycolsep</code>	529, 531, 533, 920, 1010, 1232, 1233, 1337, 1341, 4003, 4010
<code>\arrayrulecolor</code>	101
<code>\arrayrulewidth</code>	134, 139, 151, 484, 799, 937, 939, 945, 967, 1332, 1344, 1396, 1501, 1573, 1685, 1724, 1851, 1853, 1859, 1869, 1871, 1877, 1900, 1902, 1908, 1926, 1928, 1934, 3326, 3327, 3329, 3342, 3344, 3361, 3363, 3377, 3378, 3380, 3404, 3406, 3409, 3410, 3412, 3435, 3438, 3439, 3446, 3448, 3676, 3677, 3679, 3690, 3696, 3796, 3807, 3813, 3838, 4084
<code>\arraystretch</code>	1009, 4323, 4336, 4387, 4400
<code>\AtBeginDocument</code>	35, 39, 79, 95, 170, 194, 314, 608, 624, 2275, 2942, 3105, 3181, 3259, 3292, 3981
<code>\AutoNiceMatrix</code>	4656
<code>\AutoNiceMatrixWithDelims</code> ...	4616, 4648, 4660
B	
<code>\baselineskip</code>	104, 111
<code>\bgroup</code>	1085
<code>\Block</code>	1057, 5031
<code>\BNiceMatrix</code>	4608
<code>\bNiceMatrix</code>	4605
bool commands:	
<code>\bool_do_until:Nn</code>	2308, 2376
<code>\bool_gset_false:N</code>	837, 868, 1072, 1091, 2008, 2054, 3854, 3865, 4355
<code>\bool_gset_true:N</code>	1844, 2016, 2959, 2974, 2989, 3012, 3035, 3046, 3256, 3601, 3721
<code>\bool_if:NTF</code>	162, 178, 610, 615, 626, 631, 768, 771, 852, 933, 959, 962, 997, 1007, 1063, 1064, 1084, 1098, 1108, 1125, 1160, 1174, 1176, 1242, 1262, 1278, 1353, 1379, 1403, 1543,

<code>\dim_use:N</code>	1294, 1362, 4137, 4147, 4198, 4199, 4211, 4212, 4235, 4236, 4237, 4238, 4262, 4263, 4264, 4265
<code>\dim_zero_new:N</code>	4095, 4097, 4102, 4104
<code>\c_max_dim</code>	4096, 4098, 4103, 4105, 4137, 4147, 4479, 4492, 4497, 4510
<code>\l_tmpc_dim</code> 280, 3326, 3327, 3346, 3377, 3378, 3382, 3409, 3410, 3414, 3438, 3439, 3450, 3669, 3677, 3682, 3687, 3693, 3788, 3799, 3804, 3810, 4470, 4477, 4517, 4706, 4709, 4717
<code>\l_tmpd_dim</code>	281, 3344, 3346, 3380, 3383, 3412, 3415, 3448, 3451, 3678, 3682, 3795, 3799, 4472, 4477, 4497, 4506, 4510, 4513, 4517, 4708, 4709, 4721
<code>\dotfill</code>	1060, 4663
<code>\dots</code>	986
<code>\doublerulesep</code>	1502, 3679, 3691, 3796, 3808, 3839
<code>\doublerulesepcolor</code>	108
<code>\draw</code>	2775
E	
<code>\egroup</code>	1378
else commands:	
<code>\else:</code>	254
<code>\endarray</code>	1776, 1804
<code>\endBNiceMatrix</code>	4609
<code>\endbNiceMatrix</code>	4606
<code>\endNiceArray</code>	2113, 2122
<code>\endNiceArrayWithDelims</code>	2064, 2074
<code>\endpgfscope</code>	2837, 4720
<code>\endpNiceMatrix</code>	4597
<code>\endsavenotes</code>	1379
<code>\endtabularnotes</code>	1657
<code>\endVNiceMatrix</code>	4603
<code>\endvNiceMatrix</code>	4600
<code>\everycr</code>	137, 156, 1022
exp commands:	
<code>\exp_after:wN</code>	203, 1124, 1400
<code>\exp_args:Ne</code>	3059
<code>\exp_args:NNc</code>	4071
<code>\exp_args:NNe</code>	3055
<code>\exp_args:Nne</code>	2102
<code>\exp_args:NNV</code>	1796, 2946, 2961, 2976, 2991, 3014, 3109, 3185, 3263
<code>\exp_args:NNv</code>	1116
<code>\exp_args:Nnx</code>	4328, 4342, 4392, 4406
<code>\exp_args:No</code>	2770
<code>\exp_args:NV</code>	1772, 1799, 1801
<code>\exp_args:Nx</code>	4474
<code>\exp_not:N</code>	50, 51, 54, 55, 1495, 3554, 3564, 3985, 3986, 4440
<code>\exp_not:n</code>	902, 3119, 3195, 3556, 4311, 4381, 4452, 4684, 4685
<code>\ExplSyntaxOff</code>	223, 2145, 2166, 2192, 2268, 4086
<code>\ExplSyntaxOn</code>	220, 2131, 2152, 2171, 2261, 4079
<code>\extrarowheight</code>	4324, 4337, 4388, 4401
F	
<code>\fi</code>	116, 1391, 3826, 3843
fi commands:	
<code>\fi:</code>	256
<code>\fontdimen</code>	1625
fp commands:	
<code>\fp_eval:n</code>	2808
<code>\fp_to_dim:n</code>	2845
<code>\futurelet</code>	121
G	
group commands:	
<code>\group_insert_after:N</code>	4668, 4669, 4671, 4672
H	
<code>\halign</code>	1036
<code>\hbox</code>	931, 1333, 1849, 1867, 1894, 1898, 1924
hbox commands:	
<code>\hbox:n</code>	70, 72, 75
<code>\hbox_gset:Nn</code>	4316
<code>\hbox_overlap_left:n</code>	1828, 1999
<code>\hbox_overlap_right:n</code>	367, 1945, 2044
<code>\hbox_set:Nn</code> 350, 1236, 1238, 1325, 1736, 1751, 4460
<code>\hbox_set:Nw</code>	767, 1244, 1534, 1974, 2019
<code>\hbox_set_end:</code>	851, 1259, 1542, 1993, 2037
<code>\hbox_to_wd:nn</code>	392, 417
<code>\Hdotsfor</code>	1054, 4846, 4953
<code>\hdotsfor</code>	987
<code>\hdottedline</code>	1051
<code>\heavyrulewidth</code>	1667
<code>\hfil</code>	1457
<code>\hfill</code>	134, 151
<code>\Hline</code>	1052, 3829
<code>\hline</code>	114
<code>\hrule</code>	118, 134, 151, 967, 1667
<code>\hskip</code>	117
<code>\Hspace</code>	1053
<code>\hspace</code>	3047
<code>\hss</code>	1457
I	
<code>\ialign</code>	928, 1012, 1035, 2236
<code>\iddots</code>	1050, 3018, 3019, 3024, 3025
<code>\iddots</code>	65, 985, 1042
if commands:	
<code>\if_mode_math:</code>	254
<code>\ifnum</code>	116, 3826, 3843
<code>\ifstandalone</code>	1104
int commands:	
<code>\int_case:nnTF</code>	2993, 2999, 3016, 3022
<code>\int_compare:nNnTF</code> 129, 130, 146, 764, 765, 775, 782, 818, 828, 964, 966, 1135, 1137, 1180, 1190, 1209, 1260, 1264, 1275, 1280, 1284, 1285, 1645, 1686, 1725, 1797, 1825, 2022, 2318, 2325, 2329, 2331, 2386, 2393, 2397, 2399, 2492, 2534, 2586, 2627, 2629, 3078, 3092, 3171, 3246, 3339, 3373, 3441, 3443, 3510, 3561, 3575, 3576, 3583, 3584, 3588, 3959, 3961, 3963, 3965, 4530, 4559, 4561, 4623, 4625, 4627, 4631, 4633, 4635, 4637, 4639
<code>\int_compare_p:n</code> 3401, 3402, 3431, 3432, 3525, 3527
<code>\int_do_until:nNnn</code>	3488
<code>\int_gadd:Nn</code>	3091
<code>\int_gincr:N</code>	763, 791, 1107, 1475, 1526, 1548, 1554, 1920, 2017, 2689, 2740, 4064, 4301
<code>\int_if_even:nTF</code>	3538

<code>\int_incr:N</code>	346, 1485	<code>\newlist</code>	322, 333
<code>\int_step_inline:nn</code>	3534, 3536	<code>\NiceArray</code>	2111, 2120
<code>\int_step_inline:nnnn</code> 3891, 3909, 3924, 3927		<code>\NiceArrayWithDelims</code>	2062, 2072
<code>\c_zero_int</code>	3516	nicematrix commands:	
iow commands:		<code>\g_nicematrix_code_after_tl</code>	
<code>\iow_now:Nn</code>	82,	268, 579, 1793, 2254, 2255, 4728, 4736
218, 2171, 2172, 2174, 2192, 2261, 2262, 2268		<code>\g_nicematrix_code_before_tl</code>	
<code>\iow_shipout:Nn</code>	2131, 2132, 2139,	1079, 2257, 2266, 3547, 3552, 3563, 4438
2145, 2152, 2153, 2160, 2166, 4079, 4080, 4086		<code>\NiceMatrixLastEnv</code>	236
<code>\item</code>	1650, 1656	<code>\NiceMatrixOptions</code>	706, 5059, 5254
K			
<code>\kern</code>	75	<code>\NiceMatrixoptions</code>	4972
keys commands:		<code>\noalign</code> 104, 111, 116, 139, 955, 1018, 3826, 3973	
<code>\keys_define:nn</code>	460,	<code>\nobreak</code>	336
480, 487, 540, 586, 638, 674, 708, 722,		<code>\normalbaselines</code>	1006
737, 748, 3038, 3457, 4053, 4273, 4416, 4757		<code>\nulldelimiterspace</code>	1750, 1762
<code>\l_keys_key_tl</code>	4744, 4911,	<code>\numexpr</code>	159, 160
4917, 5025, 5030, 5040, 5058, 5100, 5150, 5200		O	
<code>\keys_set:nn</code>	495, 700, 707, 1120,	<code>\omit</code>	129, 1827, 1843, 1919
1121, 2101, 2109, 2118, 2495, 2537, 2589,		<code>\OnlyMainNiceMatrix</code>	1059, 3567
2663, 2714, 3174, 3249, 3266, 3465, 4066, 4435		P	
<code>\keys_set_known:nn</code>	3005, 3028, 4290	<code>\par</code>	1644, 1652
<code>\l_keys_value_tl</code>	4968, 4975, 5249	peek commands:	
L			
<code>\Ldots</code>	1046, 2949, 2952	<code>\peek_meaning:NTF</code>	164, 348, 975
<code>\ldots</code>	981, 1038	<code>\peek_meaning_ignore_spaces:NTF</code> 1771, 3829	
<code>\leaders</code>	134, 151	<code>\peek_meaning_remove_ignore_spaces:NTF</code> 154	
<code>\left</code>	1328, 1739, 1754	<code>\peek_remove_spaces:n</code>	3076
legacy commands:		<code>\pgfextracty</code>	4533
<code>\legacy_if:nTF</code>	570	<code>\pgfgetlastxy</code>	410
<code>\line</code>	2252, 4947	<code>\pgfpathcircle</code>	2934
M			
<code>\makebox</code>	1545	<code>\pgfpathlineto</code>	3687, 3693, 3804, 3810, 4709
<code>\mathinner</code>	67	<code>\pgfpathmoveto</code>	3686, 3692, 3803, 3809, 4704
mode commands:		<code>\pgfpathrectanglecorners</code>	
<code>\mode_leave_vertical:</code>	1096, 1520	3345, 3381, 3413, 3449, 3680, 3797
msg commands:		<code>\pgfpointhead</code>	408
<code>\msg_error:nn</code>	24	<code>\pgfpointanchor</code>	239, 2463, 2474,
<code>\msg_error:nnn</code>	25	4115, 4123, 4487, 4505, 4522, 4523, 4534, 4565	
<code>\msg_error:nnnn</code>	26, 4458	<code>\pgfpointdiff</code>	409, 1148, 1156
<code>\msg_fatal:nn</code>	27	<code>\pgfpointlineatime</code>	2802
<code>\msg_fatal:nnn</code>	28	<code>\pgfpointorigin</code>	1834, 1957
<code>\msg_info:nn</code>	4793	<code>\pgfpointscale</code>	408
<code>\msg_new:nnn</code>	29	<code>\pgfpointshapeborder</code>	3304, 3307
<code>\msg_new:nnnn</code>	30	<code>\pgfrememberpicturepositiononpagetrue</code>	
<code>\msg_redirect_name:nnn</code>	32	796, 874, 943, 1833, 1857, 1875, 1906,
<code>\multicolumn</code>	1056, 3049, 3053, 3101, 3122	1932, 1955, 2286, 2761, 2839, 3303, 3662,	
<code>\multispan</code>	130, 131, 147, 148	3781, 3994, 4041, 4158, 4168, 4179, 4463, 4699	
<code>\myfiledate</code>	6	<code>\pgfscope</code>	2799, 4716
<code>\myfileversion</code>	7	<code>\pgfset</code>	377, 402, 875, 4474, 4545, 4576, 4715
N			
<code>\newcolumntype</code>	230, 231, 232	<code>\pgfsetbaseline</code>	873
<code>\newcounter</code>	306	<code>\pgfsetlinewidth</code>	3696, 3813
<code>\NewDocumentCommand</code>		<code>\pgfsetrectcap</code>	3697, 3814
.	318, 341, 706, 2946, 2961, 2976, 2991,	<code>\pgfsetroundcap</code>	4712
3014, 3109, 3185, 3263, 3317, 3353, 3389,		<code>\pgfsyspdfmark</code>	221, 222
3422, 3463, 3532, 3545, 3550, 3559, 4616, 4656		<code>\pgftransformrotate</code>	2806
<code>\NewDocumentEnvironment</code>	1081,	<code>\pgftransformshift</code>	
1769, 1778, 2057, 2067, 2097, 2106, 2114, 4062		383, 408, 2800, 4544, 4563, 4717, 4721
<code>\NewExpandableDocumentCommand</code>	236, 4282	<code>\pgfusepath</code>	2826, 2836
		<code>\pgfusepathqfill</code>	
		2940, 3349, 3385, 3418, 3452, 3683, 3800
		<code>\pgfusepathqstroke</code>	3698, 3815, 4713
		<code>\phantom</code>	2958, 2973, 2988, 3011, 3034
		<code>\pNiceMatrix</code>	4596

prg commands:

- \prg_do_nothing: 182, 191, 197, 225, 477, 1018, 2253
- \prg_new_conditional:Nnn 3513, 3521
- \prg_replicate:nn 356, 1915, 1916, 3122, 3688, 3805, 4626, 4629, 4632, 4638
- \prg_return_false: 3518, 3530
- \prg_return_true: 3519, 3529
- \ProcessKeysOptions 4771
- \ProvideDocumentCommand 65
- \ProvidesExplPackage 4

Q

\quad 337

quark commands:

- \q_stop 126, 127, 143, 144, 165, 167, 1124, 1400, 1459, 1788, 1791, 3257, 3271, 3272, 3312, 3334, 3335, 3368, 3369, 3398, 3429, 3440, 4252, 4259, 4283, 4284, 4611, 4620

R

- \rectanglecolor 1169, 2259, 3554, 4440
- \refstepcounter 365

regex commands:

- \regex_const:Nn 215
- \regex_replace_all:NnN 1405
- \relax 159, 160
- \renewcommand 201
- \RenewDocumentEnvironment 4595, 4598, 4601, 4604, 4607
- \RequirePackage 1, 3, 9, 10, 11
- \right 1346, 1746, 1758
- \rotate 1058
- \rowcolor 992, 1170
- \rowcolors 1171

S

- \savenotes 1084
- \scriptstyle 771, 1976, 2021, 2783, 2787, 2822, 2832

seq commands:

- \seq_clear:N 4813
- \seq_clear_new:N 2173, 3869
- \seq_count:N 1798
- \seq_gclear:N 1110, 1111, 1112, 1673
- \seq_gclear_new:N ... 1065, 1066, 1794, 1810
- \seq_gpop_left:NN 1800, 1812
- \seq_gput_left:Nn ... 575, 3080, 3082, 4298
- \seq_gput_right:Nn 347, 2443, 3083, 4366, 4377, 4687
- \seq_gset_from_clist:Nn 2176, 2188
- \seq_gset_split:Nnn 1796, 1811
- \seq_if_empty:NnTF 2237
- \seq_if_empty_p:N 2226, 2227, 2228
- \seq_if_exist:NnTF 1127
- \seq_if_in:NnTF .. 573, 3639, 3645, 3652, 3758, 3764, 3771, 4118, 4126, 4485, 4503, 4832
- \seq_item:Nn 1131, 1134, 1143, 1144, 1151, 1152
- \seq_map_function:NN 1802
- \seq_map_inline:Nn ... 1650, 1656, 1814, 3493, 3602, 3604, 3722, 3724, 3954, 4431, 4814
- \seq_mapthread_function:NNN 4247
- \seq_new:N 245, 286, 287, 288, 307
- \seq_put_left:Nn 4816

- \seq_put_right:Nn 3941
- \seq_set_eq:NN 3482, 4818
- \seq_set_filter:NNn 3484, 3490
- \seq_set_from_clist:Nn 4822
- \seq_use:Nnnn 2190, 5259
- \l_tmpa_seq 3484, 3490, 4813, 4816, 4818
- \l_tmpb_seq 3482, 3484, 3490, 3493
- \setlist 323, 334, 611, 616, 627, 632

skip commands:

- \skip_gadd:Nn 1891
- \skip_gset:Nn 1882
- \skip_gset_eq:NN 1889, 1890
- \skip_horizontal:N 135, 152, 1245, 1246, 1257, 1258, 1287, 1301, 1336, 1337, 1340, 1341, 1373, 1374, 1396, 1560, 1573, 1763, 1764, 1766, 1767, 1838, 1839, 1851, 1853, 1869, 1871, 1893, 1900, 1902, 1921, 1926, 1928, 1947, 1948, 2004, 2005, 2006, 2009, 2043, 2048, 2049, 2050
- \skip_horizontal:n ... 368, 1290, 1358, 1497
- \skip_vertical:N 139, 937, 939, 1331, 1332, 1343, 1344, 1641, 1666, 3973
- \skip_vertical:n 833, 3836
- \g_tmpa_skip 1882, 1889, 1890, 1891, 1893, 1921
- \c_zero_skip 1023
- \space 265, 266
- \stepcounter 354, 359

str commands:

- \c_backslash_str 265
- \c_colon_str 705
- \str_case:nn ... 3059, 4537, 4549, 4568, 4580
- \str_case:nnTF 1305, 1444, 1611, 3872
- \str_count:N 1606, 1707
- \str_gclear:N 2271
- \str_gset:Nn 1093, 2061, 2070, 2099, 2108, 2116, 4647
- \str_if_empty:NnTF 800, 886, 946, 1092, 1193, 1211, 1835, 1860, 1878, 1909, 1935, 1958, 2060, 2069, 2137, 2158, 4239, 4266
- \str_if_eq:nnTF 90, 264, 926, 1462, 1490, 1567, 1587, 1695, 1818, 4733
- \str_if_eq_p:nn 466
- \str_if_in:NnTF 1599, 1700
- \str_new:N 260, 437, 449, 704
- \str_range:Nnn 1603, 1704
- \str_set:Nn 572, 696
- \str_set_eq:NN 576, 705
- \l_tmpa_str 572, 573, 575, 576
- \strut 1650, 1656

T

- \tabcolsep 919, 1336, 1340
- \tabskip 1023
- \tabularnote 318, 341, 348, 4990, 5003
- \tabularnotes 1655

TeX and L^AT_EX 2_ε commands:

- \@BTnormal 998
- \@acol 909
- \@acoll 907
- \@acolr 908
- \@array@array 911
- \@arrayacol 907, 908, 909
- \@arstrutbox 793, 794, 833, 1025, 1027, 1029, 1032, 1034, 1521, 1524

\@currenvir	4733	\tl_gclear_new:N	1073, 1074, 1075, 1076, 1077, 1078, 1079
\@gobblethree	222	\tl_gput_left:Nn	896, 1411, 1420, 3563
\@halignto	910, 922, 923	\tl_gput_right:Nn	896, 1423, 1432, 1436, 1469, 1480, 1493, 1507, 1515, 1531, 1553, 1559, 1561, 1572, 1580, 1793, 3111, 3187, 3547, 3552, 3841, 3978, 4304, 4438, 4448, 4677, 4728, 4736
\@height	119, 134, 151	\tl_gset:Nn	184, 188, 190, 1385, 1395, 2264
\@ifclassloaded	59, 62, 4792, 4802	\tl_if_blank:nTF	3319, 3355, 3391, 3424
\@ifnextchar	1070	\tl_if_blank_p:n	3672, 3791
\@ifpackageloaded	41, 44, 47, 81, 97, 172, 4795, 4805	\tl_if_empty:Nn	1123, 1644, 2257, 3336, 3337, 3370, 3371, 3610, 3614, 3625, 3729, 3733, 3744, 4287, 4436
\@mainaux	82, 218, 2131, 2132, 2139, 2145, 2152, 2153, 2160, 2166, 2171, 2172, 2174, 2192, 2261, 2262, 2268, 4079, 4080, 4086	\tl_if_empty:nTF	553, 676, 710, 726, 740, 756, 1780, 1807, 2496, 2538, 2590, 2664, 2715, 3175, 3250, 3267, 3323, 3359, 3395, 3428, 4845
\@tabarray	924	\tl_if_empty_p:N	2796, 2797
\@tempswafalse	1391	\tl_if_empty_p:n	1636
\@tempswatru	1390	\tl_if_eq:NNTF	2763, 4014, 4017
\@temptokena	181, 184, 203, 205, 1389, 1400	\tl_if_eq:nnTF	565, 687, 1783, 1785
\@whilesw	1391	\tl_if_exist:Nn	1113
\@width	119	\tl_if_in:NnTF	3333, 3367
\@xhline	122	\tl_if_single_token:nTF	695
\bBigg@	1236, 1238	\tl_item:Nn	187, 188, 190
\c@MaxMatrixCols	2089, 4860	\tl_lower_case:n	3059
\c@tabularnote	1635, 1645, 1674	\tl_map_inline:nn	1781
\col@sep	919, 920, 1287, 1295, 1369, 1374, 1838, 1891, 1947, 2009, 2043, 2507, 2516, 2549, 2556	\tl_new:N	186, 189, 242, 268, 269, 272, 274, 290, 291, 308, 433, 456, 458
\CT@arc	101, 102	\tl_put_left:Nn	998
\CT@arc@	100, 105, 120, 133, 150, 166, 168, 1100, 1667, 2273, 3695, 3812, 4040, 4711	\tl_put_right:Nn	555, 1116, 1182
\CT@drs	108, 109	\tl_range:nnn	90
\CT@drsc@	107, 112, 3672, 3675, 3791, 3794	\tl_set:Nn	187, 3338, 3340, 3372, 3374, 3442, 3444, 3597, 4291
\CT@everycr	1016	\tl_set_eq:NN	435, 3611, 3730, 4019, 4049, 4289
\CT@row@color	1018	\tl_set_rescan:Nnn	1795, 2945, 3108, 3184, 3262
\if@tempswa	1391	\tl_to_str:n	4816
\NC@	974	\g_tmpa_tl	184, 187, 190
\NC@find	182, 210	\l_tmpb_tl	3315, 3337, 3338, 3339, 3340, 3341, 3371, 3372, 3373, 3374, 3379, 3402, 3407, 3408, 3411, 3432, 3436, 3437, 3443, 3444, 3447, 3597, 3637, 3641, 3647, 3649, 3654, 3719, 3730, 3739, 3760, 3766, 3773, 3851, 3852, 3862, 3863
\NC@list	1391	\l_tmppc_tl	3598, 3610, 3611, 3614, 3619, 3621, 3625, 3630, 3632, 3718, 3729, 3730, 3733, 3738, 3740, 3744, 3749, 3751
\NC@rewrite@S	183, 201		
\new@ifnextchar	1070	token commands:	
\newcol@	976, 977	\token_to_str	4922
\nicematrix@redefine@check@rerun	82, 85	\token_to_str:N	4846, 4927, 4947, 4953, 4990, 5003, 5020, 5030, 5059, 5254
\pgf@relevantforpicturesizefalse	2287, 2762, 2840, 2931, 3322, 3358, 3394, 3427, 3663, 3782, 4159, 4169, 4180, 4464, 4698		
\pgfsys@getposition	1140, 1146, 1154	U	
\pgfsys@markposition	938, 1139, 1831, 1852, 1870, 1901, 1927, 1951	\unskip	1660
\pgfutil@check@rerun	87, 88	use commands:	
\reserved@a	121	\use:N	899, 1198, 1205, 1216, 1223, 1249, 1250, 1254, 1255, 2084, 2104
\tikz@library@external@loaded	1101	\use:n	3268, 3567
tex commands:		\usepackage	4797, 4807
\tex_mkern:D	69, 71, 73, 76	\usepgfmodule	2
\tex_the:D	205		
\textfont	1625		
\textit	310		
\textsuperscript	311, 312		
\the	159, 160, 1391, 1400		
\thetabularnote	313		
\tikzexternaldisable	1103		
\tikzset	1105, 1162, 2242		
tl commands:			
\tl_clear:N	3621, 3632, 3740, 3751		
\tl_clear_new:N	3598, 3718		
\tl_const:Nn	50, 51, 54, 55, 434, 1969, 2012		
\tl_gclear:N	1398, 2239, 2255		

V	
vbox commands:	
\ vbox:n	75
\ vbox_set_top:Nn	830
\ vbox_to_ht:nn	388, 415, 1742, 1755
\ vbox_to_zero:n	832
\ vcenter	1329, 1740, 5020
\ Vdots	1048, 2979, 2982
\ vdots	983, 1040
\ Vdotsfor	1055
\ vfill	391, 417
\ vline	120
\ VNiceMatrix	4602
\ vNiceMatrix	4599
\ vrule	118
\ vskip	117
\ vtop	935
X	
\ xglobal	778, 785, 1986, 2029

Contents

1	The environments of this package	2
2	The vertical space between the rows	2
3	The vertical position of the arrays	3
4	The blocks	4
4.1	General case	4
4.2	The mono-column blocks	5
4.3	A small remark	5
5	The rules	6
5.1	Some differences with the classical environments	6
5.1.1	The vertical rules	6
5.1.2	The command \ cline	6
5.2	The thickness and the color of the rules	7
5.3	The keys hlines and vlines	7
5.4	The key hvlines	7
5.5	The key hvlines-except-corners	8
5.6	The command \ diagbox	8
5.7	Dotted rules	9
6	The color of the rows and columns	9
6.1	Use of colortbl	9
6.2	The tools of nicematrix in the code-before	10
6.3	Color tools with the syntax of colortbl	12
7	The width of the columns	13
8	The exterior rows and columns	14
9	The continuous dotted lines	16
9.1	The option nullify-dots	17
9.2	The commands \ Hdotsfor and \ Vdotsfor	17
9.3	How to generate the continuous dotted lines transparently	18
9.4	The labels of the dotted lines	19
9.5	Customization of the dotted lines	19
9.6	The dotted lines and the rules	20
10	The code-after	20

11	The notes in the tabulars	21
11.1	The footnotes	21
11.2	The notes of tabular	21
11.3	Customisation of the tabular notes	23
11.4	Use of <code>{NiceTabular}</code> with <code>threeparttable</code>	25
12	Other features	25
12.1	Use of the column type <code>S</code> of <code>siunitx</code>	25
12.2	Alignment option in <code>{NiceMatrix}</code>	25
12.3	The command <code>\rotate</code>	25
12.4	The option <code>small</code>	26
12.5	The counters <code>iRow</code> and <code>jCol</code>	26
12.6	The option <code>light-syntax</code>	27
12.7	The environment <code>{NiceArrayWithDelims}</code>	28
13	Use of Tikz with <code>nicematrix</code>	28
13.1	The nodes corresponding to the contents of the cells	28
13.2	The “medium nodes” and the “large nodes”	29
13.3	The “row-nodes” and the “col-nodes”	30
14	API for the developpers	31
15	Technical remarks	32
15.1	Definition of new column types	32
15.2	Diagonal lines	32
15.3	The “empty” cells	33
15.4	The option <code>exterior-arraycolsep</code>	33
15.5	Incompatibilities	33
16	Examples	34
16.1	Notes in the tabulars	34
16.2	Dotted lines	35
16.3	Dotted lines which are no longer dotted	37
16.4	Width of the columns	37
16.5	How to highlight cells of the matrix	38
16.6	Direct use of the Tikz nodes	41
17	Implementation	42
18	History	161
	Index	166