

The package `nicematrix`*

F. Pantigny
fpantigny@wanadoo.fr

July 15, 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

In this 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 5.0 of `nicematrix`, at the date of 2020/07/15.

1 The environments of this package

The package `nicematrix` defines the following new environments.

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

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

However, in `{NiceArray}` (and its variants), the columns of type `w` (e.g.: `wc{1cm}`) are composed in math mode whereas, in `{array}` of `array`, they are composed in text mode.

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

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

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

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 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`²).

```
\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

4 The blocks

In the environments of `nicematrix`, it's possible to use the command `\Block` in order to place an element in the center of a rectangle of merged cells of the array. The command `\Block` don't create space by itself.

The command `\Block` must be used in the upper leftmost cell of the array with two arguments. The first argument is the size of the block with the syntax `i-j` where `i` is the number of rows of the block and `j` its number of columns. The second argument is the content of the block.

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

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

```

\begin{NiceTabular}{cccc}
rose      & tulipe & marguerite & dahlia \\
violette  & \Block{2-2}{\LARGE\color{blue} fleurs} & & souci \\
pervenche & & lys \\
arum      & iris & jacinthe & muguet
\end{NiceTabular}

```

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

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

```

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

```

	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

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

```

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

```

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

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

```

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

```

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

5 The rules

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

5.1 The thickness and the color of the rules

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

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

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

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

rose	tulipe	lys
arum	iris	violette
muguet	dahlia	souci

If one wishes to define new specifiers for columns in order to draw vertical rules (for example with a specific color or thicker than the standard rules), he should consider the command `\OnlyMainNiceMatrix` described on page 27.

5.2 A remark about `\cline`

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

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

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

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

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

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

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

5.3 The keys `hlines` and `vlines`

The key `hlines` draws all the horizontal rules and the key `vlines` draws all the vertical rules. In fact, in the environments with delimiters (as `{pNiceMatrix}` or `{bNiceArray}`) the exterior rules are not drawn (as expected).

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

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

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

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

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

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

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

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

5.4 The key `hvlines`

The key `hvlines` draws all the vertical and horizontal rules *excepted in the blocks*.³

```
\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 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}
```

³In fact, when the key `hvlines` (or the key `hvlines-except-corners` described just after) is in force, the rules are also not drawn in the virtual blocks delimited by cells relied par dotted lines (cf. p. 17).

				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.

```
\begin{NiceTabular}{*{6}{c}}%
[hvlines-except-corners, 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.⁴

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

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

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

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)
```

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

⁴The author of this document considers that type of construction as graphically poor.

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 rule⁵. In `nicematrix`, the dotted lines drawn by `\hdottedline` and “.” do likewise.

6 The color of the rows and columns

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

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

These commands are independent of `colortbl`.⁷

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

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

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

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

a	b	c
e	f	g
h	i	j

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

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

a	b	c
e	f	g
h	i	j

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

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

⁶There is also a key `code-after` : see p. 17.

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


```

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

```

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}

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

```

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

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

```

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

```

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

We have used the key `r` which aligns all the columns rightwards (cf. p. 22).

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

⁸The command `\rowcolors` of `color` 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.

7 The width of the columns

In the environments with an explicit preamble (like `{NiceTabular}`, `{NiceArray}`, etc.), it's possible to fix the width of a given column with the standard letters `w` and `W` of the package `array`. In `{NiceTabular}`, the cells of such columns are composed in text mode but, in `{NiceArray}`, `{pNiceArray}`, etc., they are composed in math mode (whereas, in `{array}` of `array`, they are composed in text mode).

```

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

```

Paris	New York	Madrid
Berlin	London	Roma
Rio	Tokyo	Oslo

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

```

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

```

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

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

It's possible to give the special value `auto` to the option `columns-width`: all the columns of the array will have a width equal to the widest cell of the array.⁹

```

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

```

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

⁹The 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`¹⁰. The environment `{NiceMatrixBlock}` has no direct link with the command `\Block` presented previously in this document (cf. p. 3).

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

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

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}$
```

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

The dotted lines have been drawn with the tools presented p. 13.

¹⁰At 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. 24) 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 27.
- 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. 10) 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`.¹¹

Each of them must be used alone in the cell of the array and it draws a dotted line between the first non-empty cells¹² 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`.¹³

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

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

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

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

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

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

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

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

In the first column of this exemple, there are two instructions `\Vdots` but only one dotted line is drawn (there is no overlapping graphic objects in the resulting PDF¹⁴).

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

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

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

¹¹The 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`.

¹²The precise definition of a “non-empty cell” is given below (cf. p. 28).

¹³It'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. 16.

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

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.¹⁵

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

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

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

9.1 The option `nullify-dots`

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

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

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

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

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

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

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

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

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

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

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

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

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

9.2 The command `\Hdotsfor`

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

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

¹⁵In `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. 10

```


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


```

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

```


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


```

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

9.3 How to generate the continuous dotted lines transparently

The package `nicematrix` provides an option called `transparent` for using existing code transparently in the environments of the `amsmath`: `{matrix}`, `{pmatrix}`, `{bmatrix}`, etc. In fact, this option is an alias for the conjunction of two options: `renew-dots` and `renew-matrix`.¹⁶

- The option `renew-dots`
With this option, the commands `\ldots`, `\cdots`, `\vdots`, `\ddots`, `\iddots`¹¹ and `\hdotsfor` are redefined within the environments provided by `nicematrix` and behave like `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots` and `\Hdotsfor`; the command `\dots` (“automatic dots” of `amsmath`) is also redefined to behave like `\Ldots`.
- The option `renew-matrix`
With this option, the environment `{matrix}` is redefined and behave like `{NiceMatrix}`, and so on for the five variants.

Therefore, with the option `transparent`, a classical code gives directly the output of `nicematrix`.

```


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


```

9.4 The labels of the dotted lines

The commands `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots` and `\Hdotsfor` (and the command `\line` in the `code-after` which is described p. 17) 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`.

```


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


```

¹⁶The options `renew-dots`, `renew-matrix` and `transparent` can be fixed with the command `\NiceMatrixOptions` like the other options. However, they can also be fixed as options of the command `\usepackage` (it's an exception for these three specific options.)

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. 17) 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. 11.

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).¹⁷

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

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

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

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

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

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

¹⁷The 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.

9.6 The dotted lines and the key hvlines

We have said (cf. p. 6) that the key `hvlines` draws all the horizontal and vertical rules, excepted in the blocks. In fact, when this key is in force, the rules are also not drawn in the virtual blocks delimited by cells relied par dotted lines.

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

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

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.¹⁸

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}
```

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

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`.¹⁹ For an example, cf. p. 33.

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`.

¹⁸There is also a key `code-before` described p. 8.

¹⁹In some circumstances, one must put `\omit \CodeAfter`. `\omit` is a keyword of TeX which cancels the pattern of the current cell.

11.2 The notes of tabular

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

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

Last name	First name	Birth day
Achard ^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.
- 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`).
- 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. 19. This table has been composed with the following code.

Table 1: Utilisation of `\tabularnote`^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

^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.

```

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

```

11.3 Customisation of the tabular notes

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

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

For sake of commodity, it is also possible to set these keys in `\NiceMatrixOptions` via a key `notes` which takes 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: `\emph{\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 = 0pt`

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. 19).

- 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. 29.

11.4 Utilisation of `{NiceTabular}` with `threeparttable`

If you wish to use the environment `{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}}
\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`.

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

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

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` (equivalent at L and R) which generate all the columns aligned leftwards (or rightwards).

```


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


```

There is also a key `S` which sets all the columns all type `S` of `siunitx` (if this package is loaded).²⁰

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}

```

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

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`).

```


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


```

²⁰This is a part of the functionality provided by the environments `{pmatrix*}`, `{bmatrix*}`, etc. of `mathtools`.

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

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 column²¹. Of course, the user must not change the value of these counters which are used internally by `nicematrix`.

In the `code-before` (cf. p. 8) and in the `code-after` (cf. p. 17), `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).

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

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

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

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

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

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

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

²¹We recall that the exterior “first row” (if it exists) has the number 0 and that the exterior “first column” (if it exists) has also the number 0.

12.6 The option `light-syntax`

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

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

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

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

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

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

When the option `light-syntax` is used, it is not possible to put verbatim material (for example with the command `\verb`) in the cells of the array.²²

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}$
```

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

13 Utilisation 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.

²²The 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.


```

 $\begin{pNiceMatrix}[name=mymatrix]$ 
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
 $\end{pNiceMatrix}$ 
\CodeAfter
\tikz \draw (2-2) circle (2mm) ;

```

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

In the following example, we have underlined all the nodes of the matrix (we explain below the technic used : cf. p. 33).

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

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`.²³

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

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

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

The names of the “large nodes” are constructed by adding the suffix “-large” to the names of the “normal nodes”. In the following example, we have underlined the “large nodes”. We consider that this example is self-explanatory.²⁴

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

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

²³There is also an option `create-extra-nodes` which is an alias for the conjunction of `create-medium-nodes` and `create-large-nodes`.

²⁴There is no “large nodes” created in the exterior rows and columns (for these rows and columns, cf. p. 11).

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`.²⁵

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

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

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

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

Here is an array composed with the following code:

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

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

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

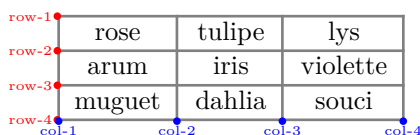
fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

Here are the “large nodes” of this array (without utilisation 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`.



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)`.

²⁵The 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.

```

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

```

```

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 Technical remarks

14.1 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 job²⁶:

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

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

```

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

```

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

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.

²⁶The command `\vrule` is a TeX (and not LaTeX) command.

14.2 Diagonal lines

By default, all the diagonal lines²⁷ of a same array are “parallelized”. That means that the first diagonal line is drawn and, then, the other lines are drawn parallel to the first one (by rotation around the left-most extremity of the line). That’s why the position of the instructions `\Ddots` in the array can have a marked effect on the final result.

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

Example with parallelization (default):

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

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

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

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

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

The same example without parallelization:

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

14.3 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`.

²⁷We speak of the lines created by `\Ddots` and not the lines created by a command `\line` in `code-after`.

14.4 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 idea²⁸. The environment `{matrix}` of `amsmath` and its variants (`{pmatrix}`, `{vmatrix}`, etc.) of `amsmath` prefer to delete these spaces with explicit instructions `\hskip -\arraycolsep`²⁹. 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).

14.5 Incompatibilities

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

15 Examples

15.1 Notes in the tabulars

The tools provided by `nicematrix` for the composition of the tabular notes have been presented in the section 11 p. 17.

Let's consider that we wish to number the notes of a tabular with stars.³⁰

First, we write a command `\stars` similar the well-known commands `\arabic`, `\alph`, `\Alph`, etc. which produce a number of stars equal to its argument³¹

```
\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 =
    {
```

²⁸In 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).*

²⁹And 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

³⁰Of course, it's realistic only when there is very few notes in the tabular.

³¹In fact: the value of its argument.

```

        widest* = \value{tabularnote} ,
        align = right
    }
}

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

```

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

*Achard is an old family of the Poitou.

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

15.2 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}

```

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

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}
```

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

An example with `\Hdotsfor`:

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

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

An example for the resultant of two polynomials:

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

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

An example for a linear system:

```

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

```

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

15.3 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}

```

$$\begin{array}{c} \xrightarrow[n \text{ columns}]{} \\ \updownarrow n \text{ rows} \end{array} \left(\begin{array}{cccc} 1 & 1 & 1 & \cdots & 1 \\ 1 & 1 & 1 & & 1 \\ 1 & 1 & 1 & & 1 \\ 1 & 1 & 1 & & 1 \\ 1 & 1 & 1 & \cdots & 1 \end{array} \right)$$

15.4 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}

```

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

15.5 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} & \\\
\end{pNiceArray}

```

```

\CodeAfter
\begin{tikzpicture}[name suffix = -large,
    every node/.style = {draw,inner sep = 0 pt}]
    \node [fit = (1-1)] {} ;
    \node [fit = (2-2)] {} ;
    \node [fit = (3-3)] {} ;
    \node [fit = (4-4)] {} ;
\end{tikzpicture}
\end{pNiceArray}$

```

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

We should remark that the rules we have drawn are drawn *after* the construction of the array and thus, they don't spread the cells of the array. We recall that, on the other side, the command `\hline`, the specifier “|” and the options `hlines`, `vlines` and `hvlines` spread the cells.³²

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}}

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

```

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

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

```

³²For the command `\cline`, see the remark p. 5.

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`.

```

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

```

$$\left(\begin{array}{c|c} \text{A} & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} \\ \hline 0 \dots \dots \dots 0 & 0 \end{array} \right)$$

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 = example-,
                             highlight/.style = {fill = red!15,
                                                  blend mode = multiply,
                                                  inner sep = 0pt,
                                                  fit = #1}}}

```

```

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

```

We obtain the following matrix.

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

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

```

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

```

We obtain the following matrix.

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

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}
```

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

15.6 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}
& & \\
& & \end{array}
```

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

```
\begin{bNiceArray}>{\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} \\ \\
```

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}

```

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

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

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

Technical definitions

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

23 \bool_new:N \c_@@_booktabs_loaded_bool
24 \bool_new:N \c_@@_enumitem_loaded_bool
25 \bool_new:N \c_@@_tikz_loaded_bool
26 \AtBeginDocument
27   {
28     \ifpackageloaded { booktabs }
29       { \bool_set_true:N \c_@@_booktabs_loaded_bool }
30     { }
31     \ifpackageloaded { enumitem }
32       { \bool_set_true:N \c_@@_enumitem_loaded_bool }
33     { }
34     \ifpackageloaded { tikz }
35     { }
```

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.

```

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

```

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.

```

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

```

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.

```

52 \ProvideDocumentCommand \iddots { }
53 {
54   \mathinner
55   {
56     \tex_mkern:D 1 mu
57     \box_move_up:nn { 1 pt } { \hbox:n { . } }
58     \tex_mkern:D 2 mu
59     \box_move_up:nn { 4 pt } { \hbox:n { . } }
60     \tex_mkern:D 2 mu
61     \box_move_up:nn { 7 pt }
62     { \vbox:n { \kern 7 pt \hbox:n { . } } }
63     \tex_mkern:D 1 mu
64   }
65 }

```

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.

```

66 \AtBeginDocument
67 {
68   \ifpackageloaded { booktabs }
69   { \iow_now:Nn \@mainaux \nicematrix@redefine@check@rerun }
70   { }
71 }
72 \cs_set_protected:Npn \nicematrix@redefine@check@rerun
73 {
74   \cs_set_eq:NN \c_@@_old_pgful@check@rerun \pgfutil@check@rerun

```

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

```

75   \cs_set_protected:Npn \pgfutil@check@rerun ##1 ##2
76   {
77     \str_if_eq:eeF { nm- } { \tl_range:nnn { ##1 } 1 3 }

```

```

78         { \@@_old_pgful@check@rerun { ##1 } { ##2 } }
79     }
80 }

```

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

```

81 \bool_new:N \c_@@_colortbl_loaded_bool
82 \AtBeginDocument
83 {
84     \ifpackageloaded { colortbl }
85     { \bool_set_true:N \c_@@_colortbl_loaded_bool }
86     {

```

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.

```

87     \cs_set_protected:Npn \CT@arc@ { }
88     \cs_set:Npn \arrayrulecolor #1 # { \CT@arc@ { #1 } }
89     \cs_set:Npn \CT@arc@ #1 #2
90     {
91         \dim_compare:nNnT \baselineskip = \c_zero_dim \noalign
92         { \cs_gset:Npn \CT@arc@ { \color #1 { #2 } } }
93     }
94     \cs_set:Npn \hline
95     {
96         \noalign { \ifnum 0 = ` } \fi
97         \cs_set_eq:NN \hskip \vskip
98         \cs_set_eq:NN \vrule \hrule
99         \cs_set_eq:NN \@width \@height
100        { \CT@arc@ \vline }
101        \futurelet \reserved@a
102        \@xhline
103    }
104 }
105 }

```

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.

```

106 \cs_set:Npn \@@_standard_cline #1 { \@@_standard_cline:w #1 \q_stop }
107 \cs_set:Npn \@@_standard_cline:w #1-#2 \q_stop
108 {
109     \int_compare:nNnT \l_@@_first_col_int = 0 { \omit & }
110     \int_compare:nNnT { #1 } > 1 { \multispan { \@@_pred:n { #1 } } & }
111     \multispan { \int_eval:n { #2 - #1 + 1 } }
112     { \CT@arc@ \leaders \hrule \@height \arrayrulewidth \hfill }

```

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

```

113 \everycr { }
114 \cr
115 \noalign { \skip_vertical:N -\arrayrulewidth }
116 }

```

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

```

117 \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`.

```

118 { \@@_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*.

```

119 \cs_set:Npn \@@_cline_i:nn #1 #2 { \@@_cline_i:w #1-#2 \q_stop }
120 \cs_set:Npn \@@_cline_i:w #1-#2-#3 \q_stop

```



```
121 {
```

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).

```
122 \int_compare:nNtT { #1 } < { #2 }
123 { \multispan { \int_eval:n { #2 - #1 } } & }
124 \multispan { \int_eval:n { #3 - #2 + 1 } }
125 { \CT@arc@ \leaders \hrule \@height \arrayrulewidth \hfill }
```

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

```
126 \peek_meaning_remove_ignore_spaces:NTF \cline
127 { & \@@_cline_i:en { \@@_succ:n { #3 } } }
128 { \everycr { } \cr }
129 }
130 \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.

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

The following command is a small shortcut.

```
133 \cs_new:Npn \@@_math_toggle_token:
134 { \bool_if:NF \l_@@_NiceTabular_bool \c_math_toggle_token }

135 \cs_new_protected:Npn \@@_set_CT@arc@:
136 { \peek_meaning:NTF [ \@@_set_CT@arc@_i: \@@_set_CT@arc@_ii: }
137 \cs_new_protected:Npn \@@_set_CT@arc@_i: [ #1 ] #2 \q_stop
138 { \cs_set:Npn \CT@arc@ { \color [ #1 ] { #2 } } }
139 \cs_new_protected:Npn \@@_set_CT@arc@_ii: #1 \q_stop
140 { \cs_set:Npn \CT@arc@ { \color { #1 } } }

141 \cs_new:Npn \@@_tab_or_array_colsep:
142 { \bool_if:NTF \l_@@_NiceTabular_bool \tabcolsep \arraycolsep }
```

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.

```
143 \bool_new:N \c_@@_siunitx_loaded_bool
144 \AtBeginDocument
145 {
146   \ifpackageloaded { siunitx }
147   { \bool_set_true:N \c_@@_siunitx_loaded_bool }
148   { }
149 }
```

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:`.

```

150 \cs_set_protected:Npn \@@_adapt_S_column:
151 {
152   \bool_if:NT \c_@@_siunitx_loaded_bool
153   {
154     \group_begin:
155     \@temptokena = { }

```

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

```

156   \cs_set_eq:NN \NC@find \prg_do_nothing:
157   \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:N`).

```

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

```

165   \cs_gset_eq:NN \@@_adapt_S_column: \prg_do_nothing:
166   }
167 }

```

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

```

168 \AtBeginDocument
169 {
170   \bool_if:nTF { ! \c_@@_siunitx_loaded_bool }
171   { \cs_set_eq:NN \@@_renew_NC@rewrite@S: \prg_do_nothing: }
172   {
173     \cs_new_protected:Npn \@@_renew_NC@rewrite@S:
174     {
175       \renewcommand*{\NC@rewrite@S}[1] []
176       {
177         \@temptokena \exp_after:wN
178         {
179           \tex_the:D \@temptokena
180           > { \@@_Cell: \c_@@_table_collect_begin_tl S {##1} }

```

`\@@_true_c:` will be replaced statically by `c` at the end of the construction of the preamble.

```

181         \@@_true_c:
182         < { \c_@@_table_print_tl \@@_end_Cell: }
183     }
184     \NC@find
185 }
186 }
187 }
188 }
```

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.

```

189 \bool_new:N \c_@@_define_L_C_R_bool
190 \cs_new_protected:Npn \@@_define_L_C_R:
191 {
192     \newcolumntype L l
193     \newcolumntype C c
194     \newcolumntype R r
195 }
196 % \end{document}
197 %
198 % \bigskip
199 % The following counter will count the environments {NiceArray}. The value of
200 % this counter will be used to prefix the names of the Tikz nodes created in the
201 % array.
202 % \begin{macrocode}
203 \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).

```

204 \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.

```

205 \NewExpandableDocumentCommand \NiceMatrixLastEnv { }
206 { \int_use:N \g_@@_env_int }
```

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

```

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

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

```

209 \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).

```

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

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

```
211 \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.

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

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

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

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

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

```
215 \cs_new_protected:Npn \@@_test_if_math_mode:
216 {
217   \if_mode_math: \else:
218     \@@_fatal:n { Outside-math-mode }
219   \fi:
220 }
```

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

```
221 \colorlet { nicematrix-last-col } { . }
222 \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`).

```
223 \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*.

```
224 \str_set:Nn \g_@@_com_or_env_str { environment }
```

The following command will be able to reconstruct the full name of the current command or environment (despite its name which contains `env`). This command must *not* be protected since it will be used in error messages.

```
225 \cs_new:Npn \@@_full_name_env:
226 {
227   \str_if_eq:VnTF \g_@@_com_or_env_str { command }
228     { command \space \c_backslash_str \g_@@_name_env_str }
229     { environment \space \{ \g_@@_name_env_str \} }
230 }
```

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`).

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

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

```
232 \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.

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

```
235 \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).

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

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

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

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

```
238 \dim_new:N \l_@@_x_initial_dim
239 \dim_new:N \l_@@_y_initial_dim
240 \dim_new:N \l_@@_x_final_dim
241 \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`).

```
242 \dim_zero_new:N \l_tmpc_dim
243 \dim_zero_new:N \l_tmpd_dim
```

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

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

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

```
245 \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”.

```
246 \dim_new:N \g_@@_width_last_col_dim
247 \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.

```
248 \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}`.

```
249 \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 required by the keys `hvlines` or `hvlines-except-corners`.

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}`.

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

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

Variables for the exterior rows and columns

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

• First row

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

```
251 \int_new:N \l_@@_first_row_int
252 \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.

```
253 \int_new:N \l_@@_first_col_int
254 \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).

```
255 \int_new:N \l_@@_last_row_int
256 \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

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

Idem for `\l_@@_last_col_without_value_bool`

```
258 \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 there is a last column but we don’t know its value because the user has used the option `last-col` without value (it’s possible in an environment without preamble like `{pNiceMatrix}`). A value of 0 means that the option `last-col` has been used in an environment with preamble (like `{pNiceArray}`).

```
259 \int_new:N \l_@@_last_col_int
260 \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.

```
261 \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.

```
262 \newcounter { tablarnote }
```

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

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

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).

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

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

```
265 \cs_new:Npn \@@_notes_style:n #1 { \emph { \alph { #1 } } }
```

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

```
266 \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`.

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

We define `\thetablarnote` 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`.

```
268 \cs_set:Npn \thetablarnote { { \@@_notes_style:n { tablarnote } } }
```

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`).

```

269 \AtBeginDocument
270 {
271   \bool_if:nTF { ! \c_@@_enumitem_loaded_bool }
272   {
273     \NewDocumentCommand \tabularnote { m }
274     { \@@_error:n { enumitem~not~loaded } }
275   }
276   {

```

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.

```

277   \newlist { tabularnotes } { enumerate } { 1 }
278   \setlist [ tabularnotes ]
279   {
280     noitemsep , leftmargin = * , align = left , labelsep = 0pt ,
281     label =
282     \@@_notes_label_in_list:n { \@@_notes_style:n { tabularnotesi } } ,
283   }
284   \newlist { tabularnotes* } { enumerate* } { 1 }
285   \setlist [ tabularnotes* ]
286   {
287     afterlabel = \nobreak ,
288     itemjoin = \quad ,
289     label =
290     \@@_notes_label_in_list:n { \@@_notes_style:n { tabularnotes*i } }
291   }

```

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

```

292   \NewDocumentCommand \tabularnote { m }
293   {
294     \bool_if:nTF { ! \l_@@_NiceArray_bool && \l_@@_in_env_bool }
295     { \@@_error:n { tabularnote~forbidden } }
296     {

```

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

```

297     \int_incr:N \l_@@_number_of_notes_int

```

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

```

298     \seq_gput_right:Nx \g_@@_tabularnotes_seq { #1 }
299     \peek_meaning:NF \tabularnote
300     {

```

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.

```

301     \hbox_set:Nn \l_tmpa_box
302     {

```

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

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

```

303         \@@_notes_label_in_tabular:n
304         {
305             \stepcounter { tabularnote }
306             \@@_notes_style:n { tabularnote }
307             \prg_replicate:nn { \l_@@_number_of_notes_int - 1 }
308             {
309                 ,
310                 \stepcounter { tabularnote }
311                 \@@_notes_style:n { tabularnote }
312             }
313         }
314     }

```

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.

```

315         \addtocounter { tabularnote } { -1 }
316         \refstepcounter { tabularnote }
317         \int_zero:N \l_@@_number_of_notes_int
318         \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).

```

319         \skip_horizontal:n { \box_wd:N \l_tmpa_box }
320     }
321 }
322 }
323 }
324 }

```

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.

```

325 \cs_new_protected:Npn \@@_pgf_rect_node:nnnnn #1 #2 #3 #4 #5
326 {
327     \begin { pgfscope }
328     \pgfset
329     {
330         outer~sep = \c_zero_dim ,
331         inner~sep = \c_zero_dim ,
332         minimum~size = \c_zero_dim
333     }
334     \pgftransformshift { \pgfpoint { 0.5 * ( #2 + #4 ) } { 0.5 * ( #3 + #5 ) } }
335     \pgfnode
336     { rectangle }
337     { center }
338     {
339         \vbox_to_ht:nn
340         { \dim_abs:n { #5 - #3 } }
341         {
342             \vfill
343             \hbox_to_wd:nn { \dim_abs:n { #4 - #2 } } { }
344         }
345     }
346     { #1 }
347     { }

```

```

348 \end { pgfscope }
349 }

```

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.

```

350 \cs_new_protected:Npn \@@_pgf_rect_node:nnn #1 #2 #3
351 {
352   \begin { pgfscope }
353   \pgfset
354   {
355     outer-sep = \c_zero_dim ,
356     inner-sep = \c_zero_dim ,
357     minimum-size = \c_zero_dim
358   }
359   \pgftransformshift { \pgfpointscale { 0.5 } { \pgfpointadd { #2 } { #3 } } }
360   \pgfpointdiff { #3 } { #2 }
361   \pgfgetlastxy \l_tmpa_dim \l_tmpb_dim
362   \pgfnode
363   { rectangle }
364   { center }
365   {
366     \vbox_to_ht:nn
367     { \dim_abs:n \l_tmpb_dim }
368     { \vfill \hbox_to_wd:nn { \dim_abs:n \l_tmpa_dim } { } }
369   }
370   { #1 }
371   { }
372   \end { pgfscope }
373 }

```

The options

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

```

374 \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`).

```

375 \dim_new:N \l_@@_cell_space_top_limit_dim
376 \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.

```

377 \dim_new:N \l_@@_inter_dots_dim
378 \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).

```

379 \dim_new:N \l_@@_xdots_shorten_dim
380 \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.

```

381 \dim_new:N \l_@@_radius_dim
382 \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.

```
383 \tl_new:N \l_@@_xdots_line_style_tl
384 \tl_const:Nn \c_@@_standard_tl { standard }
385 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
```

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

```
386 \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).

```
387 \str_new:N \l_@@_baseline_str
388 \str_set:Nn \l_@@_baseline_str c
```

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

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

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

```
390 \bool_new:N \l_@@_parallelize_diags_bool
391 \bool_set_true:N \l_@@_parallelize_diags_bool
```

The flag `\l_@@_hlines_bool` corresponds to the key `hlines`, the flag `\l_@@_vlines_bool` to the key `vlines` and the flag `hvlines` to the key `hvlines`. Since version 4.1, the key `hvlines` is no longer a mere alias for the conjunction of `hlines` and `vlines`. Indeed, with `hvlines`, the vertical and horizontal rules are *not* drawn within the blocks (created by `\Block`).

```
392 \bool_new:N \l_@@_hlines_bool
393 \bool_new:N \l_@@_vlines_bool
394 \bool_new:N \l_@@_hvlines_bool
```

The flag `\l_@@_hlines_except_corners_bool` corresponds to the key `hlines-except-corners`.

```
395 \bool_new:N \l_@@_hvlines_except_corners_bool
396 \dim_new:N \l_@@_notes_above_space_dim
397 \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.).

```
398 \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).

```
399 \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.

```
400 \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”.

```
401 \bool_new:N \l_@@_medium_nodes_bool
402 \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`).

```
403 \dim_new:N \l_@@_left_margin_dim
404 \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`.

```
405 \dim_new:N \l_@@_extra_left_margin_dim
406 \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.

```
407 \tl_new:N \l_@@_end_of_row_tl
408 \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 “:”.

```
409 \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.

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

```
411 \keys_define:nn { NiceMatrix / xdots }
412 {
413   line-style .code:n =
414   {
415     \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.

```
416     { \cs_if_exist_p:N \tikzpicture }
417     { \str_if_eq_p:nn { #1 } { standard } }
418     { \tl_set:Nn \l_@@_xdots_line_style_tl { #1 } }
419     { \@@_error:n { bad-option-for~line-style } }
420   } ,
421   line-style .value_required:n = true ,
422   color .tl_set:N = \l_@@_xdots_color_tl ,
423   color .value_required:n = true ,
424   shorten .dim_set:N = \l_@@_xdots_shorten_dim ,
425   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 `_`.

```
426   down .tl_set:N = \l_@@_xdots_down_tl ,
427   up .tl_set:N = \l_@@_xdots_up_tl ,
428   unknown .code:n = \@@_error:n { Unknown-option-for~xdots }
429 }
```

The following keys are for the tabular notes (specified by the command `\tabularnote` inside `{NiceTabular}` and `{NiceArray}`).

```

430 \keys_define:nn { NiceMatrix / rules }
431 {
432   color .tl_set:N = \l_@@_rules_color_tl ,
433   color .value_required:n = true ,
434   width .dim_set:N = \arrayrulewidth ,
435   width .value_required:n = true
436 }

```

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

```

437 \keys_define:nn { NiceMatrix / Global }
438 {
439   standard-cline .bool_set:N = \l_@@_standard_cline_bool ,
440   standard-cline .default:n = true ,
441   cell-space-top-limit .dim_set:N = \l_@@_cell_space_top_limit_dim ,
442   cell-space-top-limit .value_required:n = true ,
443   cell-space-bottom-limit .dim_set:N = \l_@@_cell_space_bottom_limit_dim ,
444   cell-space-bottom-limit .value_required:n = true ,
445   xdots .code:n = \keys_set:nn { NiceMatrix / xdots } { #1 } ,
446   max-delimiter-width .bool_set:N = \l_@@_max_delimiter_width_bool ,
447   light-syntax .bool_set:N = \l_@@_light_syntax_bool ,
448   light-syntax .default:n = true ,
449   end-of-row .tl_set:N = \l_@@_end_of_row_tl ,
450   end-of-row .value_required:n = true ,
451   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
452   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
453   last-row .int_set:N = \l_@@_last_row_int ,
454   last-row .default:n = -1 ,
455   code-for-first-col .tl_set:N = \l_@@_code_for_first_col_tl ,
456   code-for-first-col .value_required:n = true ,
457   code-for-last-col .tl_set:N = \l_@@_code_for_last_col_tl ,
458   code-for-last-col .value_required:n = true ,
459   code-for-first-row .tl_set:N = \l_@@_code_for_first_row_tl ,
460   code-for-first-row .value_required:n = true ,
461   code-for-last-row .tl_set:N = \l_@@_code_for_last_row_tl ,
462   code-for-last-row .value_required:n = true ,
463   hlines .bool_set:N = \l_@@_hlines_bool ,
464   vlines .bool_set:N = \l_@@_vlines_bool ,
465   hvlines .code:n =
466   {
467     \bool_set_true:N \l_@@_hvlines_bool
468     \bool_set_true:N \l_@@_vlines_bool
469     \bool_set_true:N \l_@@_hlines_bool
470   } ,
471   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.

```

472   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
473   renew-dots .value_forbidden:n = true ,
474   nullify-dots .bool_set:N = \l_@@_nullify_dots_bool ,
475   create-medium-nodes .bool_set:N = \l_@@_medium_nodes_bool ,
476   create-large-nodes .bool_set:N = \l_@@_large_nodes_bool ,
477   create-extra-nodes .meta:n =
478   { create-medium-nodes , create-large-nodes } ,
479   left-margin .dim_set:N = \l_@@_left_margin_dim ,
480   left-margin .default:n = \arraycolsep ,
481   right-margin .dim_set:N = \l_@@_right_margin_dim ,

```

```

482 right-margin .default:n = \arraycolsep ,
483 margin .meta:n = { left-margin = #1 , right-margin = #1 } ,
484 margin .default:n = \arraycolsep ,
485 extra-left-margin .dim_set:N = \l_@@_extra_left_margin_dim ,
486 extra-right-margin .dim_set:N = \l_@@_extra_right_margin_dim ,
487 extra-margin .meta:n =
488   { extra-left-margin = #1 , extra-right-margin = #1 } ,
489 extra-margin .value_required:n = true
490 }

```

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

```

491 \keys_define:nn { NiceMatrix / Env }
492 {
493   hvlines-except-corners .bool_set:N = \l_@@_hvlines_except_corners_bool ,
494   hvlines-except-corners .default:n = true ,
495   code-before .code:n =
496     {
497       \tl_if_empty:nF { #1 }
498       {
499         \tl_set:Nn \l_@@_code_before_tl { #1 }
500         \bool_set_true:N \l_@@_code_before_bool
501       }
502     } ,

```

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

```

503   c .code:n = \str_set:Nn \l_@@_baseline_str c ,
504   t .code:n = \str_set:Nn \l_@@_baseline_str t ,
505   b .code:n = \str_set:Nn \l_@@_baseline_str b ,
506   baseline .tl_set:N = \l_@@_baseline_str ,
507   baseline .value_required:n = true ,
508   columns-width .code:n =
509     \str_if_eq:nnTF { #1 } { auto }
510     { \bool_set_true:N \l_@@_auto_columns_width_bool }
511     { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,
512   columns-width .value_required:n = true ,
513   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.

```

514   \legacy_if:nF { measuring@ }
515   {
516     \str_set:Nn \l_tmpa_str { #1 }
517     \seq_if_in:NVTf \g_@@_names_seq \l_tmpa_str
518     { \@@_error:nn { Duplicate-name } { #1 } }
519     { \seq_gput_left:Nv \g_@@_names_seq \l_tmpa_str }
520     \str_set_eq:NN \l_@@_name_str \l_tmpa_str
521   } ,
522   name .value_required:n = true ,
523   code-after .tl_gset:N = \g_@@_code_after_tl ,
524   code-after .value_required:n = true ,
525 }
526 \keys_define:nn { NiceMatrix / notes }
527 {
528   para .bool_set:N = \l_@@_notes_para_bool ,
529   para .default:n = true ,
530   code-before .tl_set:N = \l_@@_notes_code_before_tl ,
531   code-before .value_required:n = true ,
532   code-after .tl_set:N = \l_@@_notes_code_after_tl ,
533   code-after .value_required:n = true ,
534   bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,

```

```

535 bottomrule .default:n = true ,
536 style .code:n = \cs_set:Nn \@@_notes_style:n { #1 } ,
537 style .value_required:n = true ,
538 label-in-tabular .code:n =
539   \cs_set:Nn \@@_notes_label_in_tabular:n { #1 } ,
540 label-in-tabular .value_required:n = true ,
541 label-in-list .code:n =
542   \cs_set:Nn \@@_notes_label_in_list:n { #1 } ,
543 label-in-list .value_required:n = true ,
544 enumitem-keys .code:n =
545   {
546     \bool_if:NTF \c_@@_in_preamble_bool
547     {
548       \AtBeginDocument
549       {
550         \bool_if:NT \c_@@_enumitem_loaded_bool
551         { \setlist* [ tabularnotes ] { #1 } }
552       }
553     }
554     {
555       \bool_if:NT \c_@@_enumitem_loaded_bool
556       { \setlist* [ tabularnotes ] { #1 } }
557     }
558   } ,
559 enumitem-keys .value_required:n = true ,
560 enumitem-keys-para .code:n =
561   {
562     \bool_if:NTF \c_@@_in_preamble_bool
563     {
564       \AtBeginDocument
565       {
566         \bool_if:NT \c_@@_enumitem_loaded_bool
567         { \setlist* [ tabularnotes* ] { #1 } }
568       }
569     }
570     {
571       \bool_if:NT \c_@@_enumitem_loaded_bool
572       { \setlist* [ tabularnotes* ] { #1 } }
573     }
574   } ,
575 enumitem-keys-para .value_required:n = true ,
576 unknown .code:n = \@@_error:n { Unknown~key~for~notes }
577 }

```

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

```

578 \keys_define:nn { NiceMatrix }
579 {
580   NiceMatrixOptions .inherit:n =
581   {
582     NiceMatrix / Global ,
583   } ,
584   NiceMatrixOptions / xdots .inherit:n = NiceMatrix / xdots ,
585   NiceMatrixOptions / rules .inherit:n = NiceMatrix / rules ,
586   NiceMatrixOptions / notes .inherit:n = NiceMatrix / notes ,
587   NiceMatrix .inherit:n =
588   {
589     NiceMatrix / Global ,
590     NiceMatrix / Env ,
591   } ,
592   NiceMatrix / xdots .inherit:n = NiceMatrix / xdots ,
593   NiceMatrix / rules .inherit:n = NiceMatrix / rules ,
594   NiceTabular .inherit:n =

```

```

595     {
596         NiceMatrix / Global ,
597         NiceMatrix / Env
598     } ,
599     NiceTabular / xdots .inherit:n = NiceMatrix / xdots ,
600     NiceTabular / rules .inherit:n = NiceMatrix / rules ,
601     NiceArray .inherit:n =
602     {
603         NiceMatrix / Global ,
604         NiceMatrix / Env ,
605     } ,
606     NiceArray / xdots .inherit:n = NiceMatrix / xdots ,
607     NiceArray / rules .inherit:n = NiceMatrix / rules ,
608     pNiceArray .inherit:n =
609     {
610         NiceMatrix / Global ,
611         NiceMatrix / Env ,
612     } ,
613     pNiceArray / xdots .inherit:n = NiceMatrix / xdots ,
614     pNiceArray / rules .inherit:n = NiceMatrix / rules ,
615 }

```

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

```

616 \keys_define:nn { NiceMatrix / NiceMatrixOptions }
617 {
618     last-col .code:n = \tl_if_empty:nF { #1 }
619                     { \@@_error:n { last-col-non-empty-for-NiceMatrixOptions } }
620                     \int_zero:N \l_@@_last_col_int ,
621     small .bool_set:N = \l_@@_small_bool ,
622     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.

```

623     renew-matrix .code:n = \@@_renew_matrix: ,
624     renew-matrix .value_forbidden:n = true ,
625     transparent .meta:n = { renew-dots , renew-matrix } ,
626     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}`.

```

627     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.

```

628     columns-width .code:n =
629     \str_if_eq:nnTF { #1 } { auto }
630     { \@@_error:n { Option-auto-for-columns-width } }
631     { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,

```

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

```

632     allow-duplicate-names .code:n =
633     \@@_msg_redirect_name:nn { Duplicate-name } { none } ,
634     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`).

```

635   letter-for-dotted-lines .code:n =
636   {
637     \int_compare:nTF { \tl_count:n { #1 } = 1 }
638     { \str_set:Nx \l_@@_letter_for_dotted_lines_str { #1 } }
639     { \@@_error:n { Bad-value-for-letter-for-dotted-lines } }
640   } ,
641   letter-for-dotted-lines .value_required:n = true ,
642   notes .code:n = \keys_set:nn { NiceMatrix / notes } { #1 } ,
643   notes .value_required:n = true ,
644   unknown .code:n = \@@_error:n { Unknown-key-for-NiceMatrixOptions }
645 }

646 \str_new:N \l_@@_letter_for_dotted_lines_str
647 \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.

```

648 \NewDocumentCommand \NiceMatrixOptions { m }
649 { \keys_set:nn { NiceMatrix / NiceMatrixOptions } { #1 } }

```

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

```

650 \keys_define:nn { NiceMatrix / NiceMatrix }
651 {
652   last-col .code:n = \tl_if_empty:nTF {#1}
653   {
654     \bool_set_true:N \l_@@_last_col_without_value_bool
655     \int_set:Nn \l_@@_last_col_int { -1 }
656   }
657   { \int_set:Nn \l_@@_last_col_int { #1 } } ,
658   l .code:n = \tl_set:Nn \l_@@_type_of_col_tl l ,
659   r .code:n = \tl_set:Nn \l_@@_type_of_col_tl r ,
660   L .code:n = \tl_set:Nn \l_@@_type_of_col_tl l ,
661   R .code:n = \tl_set:Nn \l_@@_type_of_col_tl r ,
662   S .code:n = \bool_if:NTF \c_@@_siunitx_loaded_bool
663   { \tl_set:Nn \l_@@_type_of_col_tl S }
664   { \@@_error:n { option-S-without-siunitx } } ,
665   small .bool_set:N = \l_@@_small_bool ,
666   small .value_forbidden:n = true ,
667   unknown .code:n = \@@_error:n { Unknown-option-for-NiceMatrix }
668 }

```

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

```

669 \keys_define:nn { NiceMatrix / NiceArray }
670 {

```

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.

```

671   small .bool_set:N = \l_@@_small_bool ,
672   small .value_forbidden:n = true ,
673   last-col .code:n = \tl_if_empty:nF { #1 }
674   { \@@_error:n { last-col-non-empty-for-NiceArray } }
675   \int_zero:N \l_@@_last_col_int ,
676   notes / para .bool_set:N = \l_@@_notes_para_bool ,
677   notes / para .default:n = true ,
678   notes / bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
679   notes / bottomrule .default:n = true ,
680   unknown .code:n = \@@_error:n { Unknown-option-for-NiceArray }
681 }

```

```

682 \keys_define:nn { NiceMatrix / pNiceArray }
683 {
684   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
685   last-col .code:n = \tl_if_empty:nF {#1}
686     { \@@_error:n { last-col~non-empty~for~NiceArray } }
687     \int_zero:N \l_@@_last_col_int ,
688   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
689   small .bool_set:N = \l_@@_small_bool ,
690   small .value_forbidden:n = true ,
691   unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix }
692 }

```

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

```

693 \keys_define:nn { NiceMatrix / NiceTabular }
694 {
695   notes / para .bool_set:N = \l_@@_notes_para_bool ,
696   notes / para .default:n = true ,
697   notes / bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
698   notes / bottomrule .default:n = true ,
699   last-col .code:n = \tl_if_empty:nF {#1}
700     { \@@_error:n { last-col~non-empty~for~NiceArray } }
701     \int_zero:N \l_@@_last_col_int ,
702   unknown .code:n = \@@_error:n { Unknown~option~for~NiceTabular }
703 }

```

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}).

```

704 \cs_new_protected:Npn \@@_Cell:
705 {

```

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

```

706   \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.

```

707   \int_compare:nNnT \c@jCol = 1
708     { \int_compare:nNnT \l_@@_first_col_int = 1 \@@_begin_of_row: }
709   \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).

```

710   \hbox_set:Nw \l_@@_cell_box
711   \bool_if:NF \l_@@_NiceTabular_bool
712   {
713     \c_math_toggle_token
714     \bool_if:NT \l_@@_small_bool \scriptstyle
715   }

```

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.

```

716   \int_compare:nNnTF \c@iRow = 0
717   {
718     \int_compare:nNnT \c@jCol > 0

```

```

719     {
720         \l_@@_code_for_first_row_tl
721         \xglobal \colorlet { nicematrix-first-row } { . }
722     }
723 }
724 {
725     \int_compare:nNnT \c@iRow = \l_@@_last_row_int
726     {
727         \l_@@_code_for_last_row_tl
728         \xglobal \colorlet { nicematrix-last-row } { . }
729     }
730 }
731 }

```

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.

```

732 \cs_new_protected:Npn \@@_begin_of_row:
733 {
734     \int_gincr:N \c@iRow
735     \dim_gset_eq:NN \g_@@_dp_ante_last_row_dim \g_@@_dp_last_row_dim
736     \dim_gset:Nn \g_@@_dp_last_row_dim { \box_dp:N \@arstrutbox }
737     \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
738     \pgfpicture
739     \pgfrememberpicturepositiononpagetrue
740     \pgfcoordinate
741     { \@@_env: - row - \int_use:N \c@iRow - base }
742     { \pgfpoint \c_zero_dim { 0.5 \arrayrulewidth } }
743     \str_if_empty:NF \l_@@_name_str
744     {
745         \pgfnodealias
746         { \l_@@_name_str - row - \int_use:N \c@iRow - base }
747         { \@@_env: - row - \int_use:N \c@iRow - base }
748     }
749     \endpgfpicture
750 }

```

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.

```

751 \cs_new_protected:Npn \@@_update_for_first_and_last_row:
752 {
753     \int_compare:nNnTF \c@iRow = 0
754     {
755         \dim_gset:Nn \g_@@_dp_row_zero_dim
756         { \dim_max:nn \g_@@_dp_row_zero_dim { \box_dp:N \l_@@_cell_box } }
757         \dim_gset:Nn \g_@@_ht_row_zero_dim
758         { \dim_max:nn \g_@@_ht_row_zero_dim { \box_ht:N \l_@@_cell_box } }
759     }
760     {
761         \int_compare:nNnT \c@iRow = 1
762         {
763             \dim_gset:Nn \g_@@_ht_row_one_dim
764             { \dim_max:nn \g_@@_ht_row_one_dim { \box_ht:N \l_@@_cell_box } }
765         }
766     }
767 }
768 \cs_new_protected:Npn \@@_end_Cell:
769 {
770     \@@_math_toggle_token:
771     \hbox_set_end:

```

```

772 \box_set_ht:Nn \l_@@_cell_box
773 { \box_ht:N \l_@@_cell_box + \l_@@_cell_space_top_limit_dim }
774 \box_set_dp:Nn \l_@@_cell_box
775 { \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”).

```

776 \dim_gset:Nn \g_@@_max_cell_width_dim
777 { \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”.

```

778 \@@_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.

```

779 \bool_if:NTF \g_@@_empty_cell_bool
780 { \box_use_drop:N \l_@@_cell_box }
781 {
782   \dim_compare:nNnTF { \box_wd:N \l_@@_cell_box } > \c_zero_dim
783     \@@_node_for_the_cell:
784     { \box_use_drop:N \l_@@_cell_box }
785 }
786 \bool_gset_false:N \g_@@_empty_cell_bool
787 }

```

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

```

788 \cs_new_protected:Npn \@@_node_for_the_cell:
789 {
790   \pgfpicture
791   \pgfsetbaseline \c_zero_dim
792   \pgfrememberpicturepositiononpagetrue
793   \pgfset
794   {
795     inner~sep = \c_zero_dim ,
796     minimum~width = \c_zero_dim
797   }
798   \pgfnode
799   { rectangle }
800   { base }
801   { \box_use_drop:N \l_@@_cell_box }
802   { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
803   { }
804   \str_if_empty:NF \l_@@_name_str
805   {
806     \pgfnodealias
807     { \l_@@_name_str - \int_use:N \c@iRow - \int_use:N \c@jCol }

```

```

808         { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
809     }
810     \endpgfpicture
811 }

```

The first argument of the following command `\@@_instruction_of_type:nn` defined below is the type of the instruction (`Cdots`, `Vdots`, `Ddots`, etc.). The second argument is the list of options. This command writes in the corresponding `\g_@@_type_lines_tl` the instruction which will actually draw the line after the construction of the matrix.

For example, for the following matrix,

```

\begin{pNiceMatrix}
1 & 2 & 3 & 4 \\
5 & \Cdots & & 6 \\
7 & \Cdots & [color=red]
\end{pNiceMatrix}

```

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

the content of `\g_@@_Cdots_lines_tl` will be:

```

\@@_draw_Cdots:nnn {2}{2}{}
\@@_draw_Cdots:nnn {3}{2}{color=red}

```

```

812 \cs_new_protected:Npn \@@_instruction_of_type:nn #1 #2
813 {

```

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).

```

814     \tl_gput_right:cx
815     { g_@@_ #1 _ lines _ tl }
816     {
817         \use:c { @@ _ draw _ #1 : nnn }
818         { \int_use:N \c@iRow }
819         { \int_use:N \c@jCol }
820         { \exp_not:n { #2 } }
821     }
822 }

```

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`.

```

823 \cs_new_protected:Npn \@@_array:
824 {
825     \bool_if:NTF \c_@@_revtex_bool
826     {
827         \cs_set_eq:NN \@acoll \@arrayacol
828         \cs_set_eq:NN \@acolr \@arrayacol
829         \cs_set_eq:NN \@acol \@arrayacol
830         \cs_set:Npn \@halignto { }
831         \@array@array
832     }
833     \array

```

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

```

834     [ \str_if_eq:VnTF \l_@@_baseline_str c c t ]
835 }

```

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.

```

836 \cs_set_eq:NN \@@_old_ialign: \ialign

```

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

```
837 \cs_new_protected:Npn \@@_create_row_node:
838 {
```

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

```
839   \hbox
840   {
841     \bool_if:NT \l_@@_code_before_bool
842     {
843       \vtop
844       {
845         \skip_vertical:N 0.5\arrayrulewidth
846         \pgfsys@markposition { \@@_env: - row - \@@_succ:n \c@iRow }
847         \skip_vertical:N -0.5\arrayrulewidth
848       }
849     }
850     \pgfpicture
851     \pgfrememberpicturerepositiononpagetrue
852     \pgfcoordinate { \@@_env: - row - \@@_succ:n \c@iRow }
853     { \pgfpoint \c_zero_dim { - 0.5 \arrayrulewidth } }
854     \str_if_empty:NF \l_@@_name_str
855     {
856       \pgfnodealias
857       { \l_@@_name_str - row - \int_use:N \c@iRow }
858       { \@@_env: - row - \int_use:N \c@iRow }
859     }
860     \endpgfpicture
861   }
862 }
```

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

```
863 \cs_new:Npn \@@_everycr: { \noalign { \@@_everycr_i: } }
864 \cs_new_protected:Npn \@@_everycr_i:
865 {
866   \int_gzero:N \c@jCol
867   \bool_if:NF \g_@@_row_of_col_done_bool
868   {
869     \@@_create_row_node:
```

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

```
870     \bool_if:NT \l_@@_hlines_bool
871     {
```

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.

```
872     \int_compare:nNnT \c@iRow > { -1 }
873     {
874       \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@`.

```
875       { \hrule height \arrayrulewidth width \c_zero_dim }
876     }
877   }
878 }
879 }
```

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

```
880 \cs_set_protected:Npn \@@_newcolumntype #1
```

```

881 {
882   % \cs_if_free:cT { NC @ find @ #1 }
883   % { \NC@list \exp_after:wN { \the \NC@list \NC@do #1 } }
884   \cs_set:cpn { NC @ find @ #1 } ##1 #1 { \NC@ { ##1 } }
885   \peek_meaning:NTF [
886     { \newcol@ #1 }
887     { \newcol@ #1 [ 0 ] }
888   }

```

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

```

889 \cs_new_protected:Npn \@@_define_columntype:nn #1 #2
890 {

```

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

```

891   \@@_newcolumntype #1 [ 1 ]
892   {
893     > {
894       \@@_Cell:
895       \begin { minipage } [ #2 ] { ##1 }
896       \mode_leave_vertical: \box_use:N \@arstrutbox
897     }

```

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

```

898       c
899       < { \box_use:N \@arstrutbox \end { minipage } \@@_end_Cell: }
900     }
901   }

```

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

```

902 \cs_new_protected:Npn \@@_pre_array:
903 {

```

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 redefinition³⁵.

```

904   \bool_if:NT \c_@@_booktabs_loaded_bool
905   { \tl_put_left:Nn \@BTnormal \@@_create_row_node: }
906   \box_clear_new:N \l_@@_cell_box
907   \cs_if_exist:NT \theiRow
908   { \int_set_eq:NN \l_@@_old_iRow_int \c@iRow }
909   \int_gzero_new:N \c@iRow
910   \cs_if_exist:NT \thejCol
911   { \int_set_eq:NN \l_@@_old_jCol_int \c@jCol }
912   \int_gzero_new:N \c@jCol
913   \normalbaselines

```

If the option `small` is used, we have to do some tuning. In particular, we change the value of `\arraystretch` (this parameter is used in the construction of `\@arstrutbox` in the beginning of `{array}`).

```

914   \bool_if:NT \l_@@_small_bool
915   {

```

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

```

916     \cs_set:Npn \arraystretch { 0.47 }
917     \dim_set:Nn \arraycolsep { 1.45 pt }
918 }

```

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`.

```

919     \cs_set:Npn \ialign
920     {
921         \bool_if:NT \l_@@_NiceTabular_bool
922         { \dim_set_eq:NN \arraycolsep \@@_old_arraycolsep_dim }
923         \bool_if:NTF \c_@@_colortbl_loaded_bool
924         {
925             \CT@everycr
926             {
927                 \noalign { \cs_gset_eq:NN \CT@row@color \prg_do_nothing: }
928                 \@@_everycr:
929             }
930         }
931         { \everycr { \@@_everycr: } }
932         \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`.

```

933     \dim_gzero_new:N \g_@@_dp_row_zero_dim
934     \dim_gset:Nn \g_@@_dp_row_zero_dim { \box_dp:N \@arstrutbox }
935     \dim_gzero_new:N \g_@@_ht_row_zero_dim
936     \dim_gset:Nn \g_@@_ht_row_zero_dim { \box_ht:N \@arstrutbox }
937     \dim_gzero_new:N \g_@@_ht_row_one_dim
938     \dim_gset:Nn \g_@@_ht_row_one_dim { \box_ht:N \@arstrutbox }
939     \dim_gzero_new:N \g_@@_dp_ante_last_row_dim
940     \dim_gzero_new:N \g_@@_ht_last_row_dim
941     \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
942     \dim_gzero_new:N \g_@@_dp_last_row_dim
943     \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`.

```

944     \cs_set_eq:NN \ialign \@@_old_ialign:
945     \halign
946 }

```

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).

```

947     \cs_set_eq:NN \@@_old_ldots \ldots
948     \cs_set_eq:NN \@@_old_cdots \cdots
949     \cs_set_eq:NN \@@_old_vdots \vdots
950     \cs_set_eq:NN \@@_old_ddots \ddots
951     \cs_set_eq:NN \@@_old_iddots \iddots
952     \cs_set_eq:NN \firsthline \hline
953     \cs_set_eq:NN \lasthline \hline
954     \bool_if:NTF \l_@@_standard_cline_bool
955     { \cs_set_eq:NN \cline \@@_standard_cline }
956     { \cs_set_eq:NN \cline \@@_cline }
957     \cs_set_eq:NN \Ldots \@@_Ldots
958     \cs_set_eq:NN \Cdots \@@_Cdots

```

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


```

959 \cs_set_eq:NN \Vdots \@@_Vdots
960 \cs_set_eq:NN \Ddots \@@_Ddots
961 \cs_set_eq:NN \Iddots \@@_Iddots
962 \cs_set_eq:NN \hdottedline \@@_hdottedline:
963 \cs_set_eq:NN \Hspace \@@_Hspace:
964 \cs_set_eq:NN \Hdotsfor \@@_Hdotsfor:
965 \cs_set_eq:NN \Vdotsfor \@@_Vdotsfor:
966 \cs_set_eq:NN \multicolumn \@@_multicolumn:nnn
967 \cs_set_eq:NN \Block \@@_Block:
968 \cs_set_eq:NN \rotate \@@_rotate:
969 \cs_set_eq:NN \OnlyMainNiceMatrix \@@_OnlyMainNiceMatrix:n
970 \cs_set_eq:NN \dotfill \@@_dotfill:
971 \cs_set_eq:NN \CodeAfter \@@_CodeAfter:n
972 \cs_set_eq:NN \diagbox \@@_diagbox:nn
973 \bool_if:NT \l_@@_renew_dots_bool
974 {
975   \cs_set_eq:NN \ldots \@@_Ldots
976   \cs_set_eq:NN \cdots \@@_Cdots
977   \cs_set_eq:NN \vdots \@@_Vdots
978   \cs_set_eq:NN \ddots \@@_Ddots
979   \cs_set_eq:NN \iddots \@@_Iddots
980   \cs_set_eq:NN \dots \@@_Ldots
981   \cs_set_eq:NN \hdotsfor \@@_Hdotsfor:
982 }

```

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).

```

983 \seq_gclear_new:N \g_@@_multicolumn_cells_seq
984 \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).

```

985 \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`).

```

986 \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.

```

987 \int_gzero_new:N \g_@@_col_total_int
988 \cs_set_eq:NN \@ifnextchar \new@ifnextchar

```

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

```

989 \@@_define_columntype:nn p t
990 \@@_define_columntype:nn m c
991 \@@_define_columntype:nn b b

```

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.

```

992 \@@_newcolumntype w [ 2 ]
993 {
994   > {
995     \hbox_set:Nw \l_@@_cell_box
996     \@@_Cell:
997   }

```

`\@@_true_c`: will be replaced statically by `c` at the end of the construction of the preamble.

```

998      \@@_true_c:
999      < {
1000          \@@_end_Cell:
1001          \hbox_set_end:

```

The construction with `\@@_foldcase`: (defined below) should be replaced by `\str_foldcase:n` when that command of `expl3` will be widespread. That construction is only for giving the user the ability to write `wC{1cm}` instead of `wc{1cm}` for homogeneity with the letters `L`, `C` and `R` used elsewhere in the preamble instead of `l`, `c` and `r`.

```

1002      \tl_set:Nn \l_tmpa_tl { ##1 }
1003      \makebox [ ##2 ] [ \@@_foldcase: ]
1004      { \box_use_drop:N \l_@@_cell_box }
1005  }
1006  }
1007  \@@_newcolumnntype W [ 2 ]
1008  {
1009      > {
1010          \hbox_set:Nw \l_@@_cell_box
1011          \@@_Cell:
1012      }

```

`\@@_true_c`: will be replaced statically by `c` at the end of the construction of the preamble.

```

1013      \@@_true_c:
1014      < {
1015          \@@_end_Cell:
1016          \hbox_set_end:
1017          \cs_set_eq:NN \hss \hfil
1018          \tl_set:Nn \l_tmpa_tl { ##1 }
1019          \makebox [ ##2 ] [ \@@_foldcase: ]
1020          { \box_use_drop:N \l_@@_cell_box }
1021      }
1022  }

```

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

```

1023      \tl_set_rescan:Nno
1024      \l_@@_letter_for_dotted_lines_str { } \l_@@_letter_for_dotted_lines_str
1025      \exp_args:NV \newcolumnntype \l_@@_letter_for_dotted_lines_str
1026      {
1027          !
1028      {

```

The following code because we want the dotted line to have exactly the same position as a vertical rule drawn by “|” (considering the rule having a width equal to the diameter of the dots).

```

1029      \int_compare:nNnF \c@iRow = 0
1030      {
1031          \int_compare:nNnF \c@iRow = \l_@@_last_row_int
1032          { \skip_horizontal:N 2\l_@@_radius_dim }
1033      }

```

Consider the following code:

```

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

```

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

```

1034         \int_compare:nNnT \c@jCol > \g_@@_last_vdotted_col_int
1035         {
1036             \int_gset_eq:NN \g_@@_last_vdotted_col_int \c@jCol
1037             \tl_gput_right:Nx \g_@@_internal_code_after_tl

```

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

```

1038             { \@@_vdottedline:n { \int_use:N \c@jCol } }
1039         }
1040     }
1041 }
1042 \int_gzero_new:N \g_@@_last_vdotted_col_int
1043 \@@_renew_NC@rewrite@S:
1044 \int_gset:Nn \g_@@_last_vdotted_col_int { -1 }
1045 \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.

```

1046 \tl_gclear_new:N \g_@@_Cdots_lines_tl
1047 \tl_gclear_new:N \g_@@_Ldots_lines_tl
1048 \tl_gclear_new:N \g_@@_Vdots_lines_tl
1049 \tl_gclear_new:N \g_@@_Ddots_lines_tl
1050 \tl_gclear_new:N \g_@@_Iddots_lines_tl
1051 \tl_gclear_new:N \g_@@_HVDotsfor_lines_tl
1052 }

```

The following command will be deleted when we will consider `\str_lowercase:n` as widespread.

```

1053 \cs_set:Npn \@@_foldcase:
1054 {
1055     \tl_case:NnF \l_tmpa_tl
1056     {
1057         C { c }
1058         L { l }
1059         R { r }
1060     }
1061     { \l_tmpa_tl }
1062 }

```

The environment `{NiceArrayWithDelims}`

```

1063 \NewDocumentEnvironment { NiceArrayWithDelims } { m m O { } m ! O { } }
1064 {
1065     \bool_if:NT \c_@@_footnote_bool { \begin { 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.

```

1066 \bgroup
1067 \tl_set:Nn \l_@@_left_delim_tl { #1 }
1068 \tl_set:Nn \l_@@_right_delim_tl { #2 }
1069 \bool_gset_false:N \g_@@_row_of_col_done_bool
1070 \str_if_empty:NT \g_@@_name_env_str
1071 { \str_gset:Nn \g_@@_name_env_str { NiceArrayWithDelims } }
1072 \@@_adapt_S_column:
1073 \bool_if:NTF \l_@@_NiceTabular_bool
1074     \mode_leave_vertical:
1075     \@@_test_if_math_mode:

```

```

1076 \bool_if:NT \l_@@_in_env_bool { \@@_fatal:n { Yet~in~env } }
1077 \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.

```

1078 \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).

```

1079 \cs_if_exist:NT \tikz@library@external@loaded
1080 {
1081   \tikzset { external / export = false }
1082   \cs_if_exist:NT \ifstandalone
1083     { \tikzset { external / optimize = false } }
1084 }

```

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

```

1085 \int_gincr:N \g_@@_env_int
1086 \bool_if:NF \l_@@_block_auto_columns_width_bool
1087 { \dim_gzero_new:N \g_@@_max_cell_width_dim }

```

We do a redefinition of `\@arrayrule` because we want that the vertical rules drawn by `|` in the preamble of the array don't extend in the potential exterior rows.

```

1088 \cs_set_protected:Npn \@arrayrule { \@addtopreamble \@@_vline: }

```

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.

```

1089 \seq_clear:N \g_@@_blocks_seq
1090 \seq_clear:N \g_@@_pos_of_blocks_seq

```

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

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`.

```

1091 \bool_if:NTF \l_@@_NiceArray_bool
1092 { \keys_set:nn { NiceMatrix / NiceArray } }
1093 { \keys_set:nn { NiceMatrix / pNiceArray } }
1094 { #3 , #5 }

1095 \tl_if_empty:NF \l_@@_rules_color_tl
1096 { \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.

```

1097 \bool_if:NT \l_@@_code_before_bool
1098 {
1099   \seq_if_exist:cT { @@_size_ \int_use:N \g_@@_env_int _ seq }
1100   {

```

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).

```

1101 \int_zero_new:N \c@iRow
1102 \int_set:Nn \c@iRow
1103 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 2 }
1104 \int_zero_new:N \c@jCol
1105 \int_set:Nn \c@jCol
1106 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 4 }

```

³⁷e.g. `\color[rgb]{0.5,0.5,0}`

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.

```

1107         \int_compare:nNnF \l_@@_last_row_int = { -2 }
1108         { \int_decr:N \c@iRow }
1109         \int_compare:nNnF \l_@@_last_col_int = { -2 }
1110         { \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.

```

1111         \pgfsys@markposition { \@@_env: - position }
1112         \pgfsys@getposition { \@@_env: - position } \@@_picture_position:
1113         \pgfpicture

```

First, the creation of the `row` nodes.

```

1114         \int_step_inline:nnn
1115         { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 1 }
1116         { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 2 + 1 }
1117         {
1118             \pgfsys@getposition { \@@_env: - row - ##1 } \@@_node_position:
1119             \pgfcoordinate { \@@_env: - row - ##1 }
1120             { \pgfpointdiff \@@_picture_position: \@@_node_position: }
1121         }

```

Now, the creation of the `col` nodes.

```

1122         \int_step_inline:nnn
1123         { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 3 }
1124         { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 4 + 1 }
1125         {
1126             \pgfsys@getposition { \@@_env: - col - ##1 } \@@_node_position:
1127             \pgfcoordinate { \@@_env: - col - ##1 }
1128             { \pgfpointdiff \@@_picture_position: \@@_node_position: }
1129         }
1130         \endpgfpicture
1131         \group_begin:
1132         \bool_if:NT \c_@@_tikz_loaded_bool
1133         {
1134             \tikzset
1135             {
1136                 every~picture / .style =
1137                 { overlay , name~prefix = \@@_env: - }
1138             }
1139         }
1140         \cs_set_eq:NN \cellcolor \@@_cellcolor
1141         \cs_set_eq:NN \rectanglecolor \@@_rectanglecolor
1142         \cs_set_eq:NN \rowcolor \@@_rowcolor
1143         \cs_set_eq:NN \rowcolors \@@_rowcolors
1144         \cs_set_eq:NN \columncolor \@@_columncolor
1145         \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`.

```

1146         \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
1147         \l_@@_code_before_tl
1148         \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
1149         \group_end:
1150     }
1151 }

```

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).

```

1152         \int_compare:nNnT \l_@@_last_row_int > { -2 }
1153         {
1154             \tl_put_right:Nn \@@_update_for_first_and_last_row:
1155             {

```

```

1156         \dim_gset:Nn \g_@@_ht_last_row_dim
1157         { \dim_max:nn \g_@@_ht_last_row_dim { \box_ht:N \l_@@_cell_box } }
1158         \dim_gset:Nn \g_@@_dp_last_row_dim
1159         { \dim_max:nn \g_@@_dp_last_row_dim { \box_dp:N \l_@@_cell_box } }
1160     }
1161 }
1162 \int_compare:nNnT \l_@@_last_row_int = { -1 }
1163 {
1164     \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.

```

1165     \str_if_empty:NTF \l_@@_name_str
1166     {
1167         \cs_if_exist:cT { @@_last_row_ \int_use:N \g_@@_env_int }
1168         {
1169             \int_set:Nn \l_@@_last_row_int
1170             { \use:c { @@_last_row_ \int_use:N \g_@@_env_int } }
1171         }
1172     }
1173     {
1174         \cs_if_exist:cT { @@_last_row_ \l_@@_name_str }
1175         {
1176             \int_set:Nn \l_@@_last_row_int
1177             { \use:c { @@_last_row_ \l_@@_name_str } }
1178         }
1179     }
1180 }

```

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).

```

1181     \int_compare:nNnT \l_@@_last_col_int = { -1 }
1182     {
1183         \str_if_empty:NTF \l_@@_name_str
1184         {
1185             \cs_if_exist:cT { @@_last_col_ \int_use:N \g_@@_env_int }
1186             {
1187                 \int_set:Nn \l_@@_last_col_int
1188                 { \use:c { @@_last_col_ \int_use:N \g_@@_env_int } }
1189             }
1190         }
1191         {
1192             \cs_if_exist:cT { @@_last_col_ \l_@@_name_str }
1193             {
1194                 \int_set:Nn \l_@@_last_col_int
1195                 { \use:c { @@_last_col_ \l_@@_name_str } }
1196             }
1197         }
1198     }

```

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

```

1199     \@@_pre_array:

```

We compute the width of the two delimiters.

```

1200     \dim_zero_new:N \l_@@_left_delim_dim
1201     \dim_zero_new:N \l_@@_right_delim_dim
1202     \bool_if:NTF \l_@@_NiceArray_bool
1203     {
1204         \dim_gset:Nn \l_@@_left_delim_dim { 2 \arraycolsep }
1205         \dim_gset:Nn \l_@@_right_delim_dim { 2 \arraycolsep }
1206     }
1207     {

```

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

```

1208     \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 #1 $ }
1209     \dim_set:Nn \l_@@_left_delim_dim { \box_wd:N \l_tmpa_box }
1210     \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 #2 $ }
1211     \dim_set:Nn \l_@@_right_delim_dim { \box_wd:N \l_tmpa_box }
1212 }

```

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

```

1213     \box_clear_new:N \l_@@_the_array_box

```

We construct the preamble of the array in `\l_tmpa_tl`.

```

1214     \tl_set:Nn \l_tmpa_tl { #4 }

```

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).

```

1215     \bool_if:NT \c_@@_define_L_C_R_bool \@@_define_L_C_R:

```

We will do an “expansion”³⁸ of the preamble (with the tools of `array` itself) and a replacement of the specifiers `c`, `l` and `r` by the specifiers `C`, `L` and `R`.

We store our preamble in the token register `\@temptokena` (those “token registers” are not supported by `expl3`).

```

1216     \@temptokena \exp_after:wN { \l_tmpa_tl }

```

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

```

1217     \@tempswatrue

```

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

```

1218     \@whilesw \if@tempswa \fi { \@tempswafalse \the \NC@list }

```

We put back the preamble in our token list `\l_tmpa_tl`.

```

1219     \tl_gset:NV \l_tmpa_tl \@temptokena

```

We do the replacement of letters `c`, `l` and `r` (these column types can't be redefined by `\newcolumntype` of `array` and that's why we have to do the job “by hand” after expansion of the preamble).

```

1220     \tl_replace_all:Nnn \l_tmpa_tl c { > \@@_Cell: c < \@@_end_Cell: }
1221     \tl_replace_all:Nnn \l_tmpa_tl l { > \@@_Cell: l < \@@_end_Cell: }
1222     \tl_replace_all:Nnn \l_tmpa_tl r { > \@@_Cell: r < \@@_end_Cell: }
1223     \tl_replace_all:Nnn \l_tmpa_tl \@@_true_c: c

```

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

```

1224     \int_compare:nNnTF \l_@@_first_col_int = 0
1225     { \tl_put_left:NV \l_tmpa_tl \c_@@_preamble_first_col_tl }
1226     {
1227         \bool_lazy_all:nT
1228         {
1229             \l_@@_NiceArray_bool
1230             { \bool_not_p:n \l_@@_NiceTabular_bool }
1231             { \bool_not_p:n \l_@@_vlines_bool }
1232             { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
1233         }
1234         { \tl_put_left:Nn \l_tmpa_tl { @ { } } }
1235     }
1236     \int_compare:nNnTF \l_@@_last_col_int > { -1 }
1237     { \tl_put_right:NV \l_tmpa_tl \c_@@_preamble_last_col_tl }
1238     {
1239         \bool_lazy_all:nT
1240         {
1241             \l_@@_NiceArray_bool
1242             { \bool_not_p:n \l_@@_NiceTabular_bool }
1243             { \bool_not_p:n \l_@@_vlines_bool }
1244             { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }

```

³⁸This “expansion” expands all the constructions with `*` and with the new column types defined by `\newcolumntype`.

```

1245     }
1246     { \tl_put_right:Nn \l_tmpa_tl { @ { } } }
1247   }
1248   \tl_put_right:Nn \l_tmpa_tl { > { \@@_error_too_much_cols: } 1 }

```

Now, the preamble is constructed in `\l_tmpa_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).

```

1249   \hbox_set:Nw \l_@@_the_array_box

```

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

```

1250   \bool_if:NT \l_@@_NiceTabular_bool
1251   {
1252     \dim_set_eq:NN \@@_old_arraycolsep_dim \arraycolsep
1253     \dim_set_eq:NN \arraycolsep \tabcolsep
1254   }

```

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

```

1255   \bool_if:NT \l_@@_vlines_bool
1256   {
1257     \dim_add:Nn \arraycolsep { 0.5 \arrayrulewidth }
1258     \skip_horizontal:N 0.5\arrayrulewidth
1259   }
1260   \skip_horizontal:N \l_@@_left_margin_dim
1261   \skip_horizontal:N \l_@@_extra_left_margin_dim
1262   \c_math_toggle_token
1263   \bool_if:NTF \l_@@_light_syntax_bool
1264   { \use:c { @@-light-syntax } }
1265   { \use:c { @@-normal-syntax } }
1266 }
1267 {
1268   \bool_if:NTF \l_@@_light_syntax_bool
1269   { \use:c { end @@-light-syntax } }
1270   { \use:c { end @@-normal-syntax } }
1271   \c_math_toggle_token
1272   \skip_horizontal:N \l_@@_right_margin_dim
1273   \skip_horizontal:N \l_@@_extra_right_margin_dim

```

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

```

1274   \bool_if:NT \l_@@_vlines_bool { \skip_horizontal:N 0.5\arrayrulewidth }
1275   \hbox_set_end:

```

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

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

```

1276   \int_compare:nNnT \l_@@_last_row_int > { -2 }
1277   {
1278     \bool_if:NF \l_@@_last_row_without_value_bool
1279     {

```



```

1280         \int_compare:nNnF \l_@@_last_row_int = \c@iRow
1281         {
1282             \@@_error:n { Wrong~last~row }
1283             \int_gset_eq:NN \l_@@_last_row_int \c@iRow
1284         }
1285     }
1286 }

```

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”.³⁹

```

1287     \int_gset_eq:NN \c@jCol \g_@@_col_total_int
1288     \bool_if:nT \g_@@_last_col_found_bool { \int_gdecr:N \c@jCol }

```

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

```

1289     \int_gset_eq:NN \g_@@_row_total_int \c@iRow
1290     \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. 82).

```

1291     \int_compare:nNnT \l_@@_first_col_int = 0
1292     {
1293         \skip_horizontal:N \arraycolsep
1294         \skip_horizontal:N \g_@@_width_first_col_dim
1295     }

```

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 to take into account the value of `baseline` and we have no delimiter to put. We begin with this case.

```

1296     \bool_if:NTF \l_@@_NiceArray_bool
1297     {

```

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`.

```

1298         \str_if_eq:VnTF \l_@@_baseline_str { b }
1299         {
1300             \pgfpicture
1301             \@@_qpoint:n { row - 1 }
1302             \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1303             \@@_qpoint:n { row - \int_use:N \c@iRow - base }
1304             \dim_gsub:Nn \g_tmpa_dim \pgf@y
1305             \endpgfpicture
1306             \dim_gadd:Nn \g_tmpa_dim \arrayrulewidth
1307             \int_compare:nNnT \l_@@_first_row_int = 0
1308             {
1309                 \dim_gadd:Nn \g_tmpa_dim \g_@@_ht_row_zero_dim
1310                 \dim_gadd:Nn \g_tmpa_dim \g_@@_dp_row_zero_dim
1311             }
1312             \box_move_up:nn \g_tmpa_dim { \@@_use_array_box_with_notes: }
1313         }
1314         {
1315             \str_if_eq:VnTF \l_@@_baseline_str { c }
1316             { \@@_use_array_box_with_notes: }
1317             {

```

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

```

1318             \str_if_eq:VnT \l_@@_baseline_str { t }
1319             { \str_set:Nn \l_@@_baseline_str { 1 } }

```

Now, we convert the value of `\l_@@_baseline_str` (which should represent an integer) to an integer stored in `\l_tmpa_int`.

```

1320             \int_set:Nn \l_tmpa_int \l_@@_baseline_str
1321             \bool_lazy_or:nnT
1322             { \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int }

```

³⁹We remind that the potential “first column” (exterior) has the number 0.

```

1323         { \int_compare:nNn \l_tmpa_int > \g_@@_row_total_int }
1324         {
1325             \@@_error:n { bad-value-for-baseline }
1326             \int_set:Nn \l_tmpa_int 1
1327         }
1328         \pgfpicture
1329         \@@_qpoint:n { row - 1 }
1330         \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1331         \@@_qpoint:n { row - \int_use:N \l_tmpa_int - base }
1332         \dim_gsub:Nn \g_tmpa_dim \pgf@y
1333         \endpgfpicture
1334         \dim_gadd:Nn \g_tmpa_dim \arrayrulewidth
1335         \int_compare:nNnT \l_@@_first_row_int = 0
1336         {
1337             \dim_gadd:Nn \g_tmpa_dim \g_@@_ht_row_zero_dim
1338             \dim_gadd:Nn \g_tmpa_dim \g_@@_dp_row_zero_dim
1339         }
1340         \box_move_up:nn \g_tmpa_dim { \@@_use_array_box_with_notes: }
1341     }
1342 }
1343 }

```

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).

```

1344 {
1345     \int_compare:nNnTF \l_@@_first_row_int = 0
1346     {
1347         \dim_set_eq:NN \l_tmpa_dim \g_@@_dp_row_zero_dim
1348         \dim_add:Nn \l_tmpa_dim \g_@@_ht_row_zero_dim
1349     }
1350     { \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”.⁴⁰

```

1351     \int_compare:nNnTF \l_@@_last_row_int > { -2 }
1352     {
1353         \dim_set_eq:NN \l_tmpb_dim \g_@@_ht_last_row_dim
1354         \dim_add:Nn \l_tmpb_dim \g_@@_dp_last_row_dim
1355     }
1356     { \dim_zero:N \l_tmpb_dim }
1357     \hbox_set:Nn \l_tmpa_box
1358     {
1359         \c_math_toggle_token
1360         \left #1
1361         \vcenter
1362         {

```

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.

```

1363         \skip_vertical:N -\l_tmpa_dim
1364         \skip_vertical:N -\arrayrulewidth
1365         \hbox
1366         {
1367             \bool_if:NTF \l_@@_NiceTabular_bool
1368             { \skip_horizontal:N -\tabcolsep }
1369             { \skip_horizontal:N -\arraycolsep }
1370             \@@_use_array_box_with_notes:
1371             \bool_if:NTF \l_@@_NiceTabular_bool
1372             { \skip_horizontal:N -\tabcolsep }
1373             { \skip_horizontal:N -\arraycolsep }
1374         }

```

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

⁴⁰A 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).

```

1375         \skip_vertical:N -\l_tmpb_dim
1376         \skip_vertical:N \arrayrulewidth
1377     }
1378     \right #2
1379     \c_math_toggle_token
1380 }

```

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.

```

1381     \bool_if:NTF \l_@@_max_delimiter_width_bool
1382     { \@@_put_box_in_flow_bis:nn { #1 } { #2 } }
1383     \@@_put_box_in_flow:
1384 }

```

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. 83).

```

1385     \bool_if:NT \g_@@_last_col_found_bool
1386     {
1387         \skip_horizontal:N \g_@@_width_last_col_dim
1388         \skip_horizontal:N \arraycolsep
1389     }
1390     \@@_after_array:
1391     \egroup
1392     \bool_if:NT \c_@@_footnote_bool { \end { savenotes } }
1393 }

```

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

The command `\@@_put_box_in_flow:` puts the box `\l_tmpa_box` (which contains the array) in the flow. It is used for the environments with delimiters. First, we have to modify the height and the depth to take back into account the potential exterior rows (the total height of the first row has been computed in `\l_tmpa_dim` and the total height of the potential last row in `\l_tmpb_dim`).

```

1394 \cs_new_protected:Npn \@@_put_box_in_flow:
1395 {
1396     \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }
1397     \box_set_dp:Nn \l_tmpa_box { \box_dp:N \l_tmpa_box + \l_tmpb_dim }
1398     \str_if_eq:VnTF \l_@@_baseline_str { c }
1399     { \box_use_drop:N \l_tmpa_box }
1400     \@@_put_box_in_flow_i:
1401 }

```

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).

```

1402 \cs_new_protected:Npn \@@_put_box_in_flow_i:
1403 {
1404     \str_case:VnF \l_@@_baseline_str
1405     {
1406         { t } { \int_set:Nn \l_tmpa_int 1 }
1407         { b } { \int_set_eq:NN \l_tmpa_int \c@iRow }
1408     }
1409     { \int_set:Nn \l_tmpa_int \l_@@_baseline_str }
1410     \bool_if:nT
1411     {
1412         \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int
1413         || \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int
1414     }
1415     {
1416         \@@_error:n { bad~value~for~baseline }
1417         \int_set:Nn \l_tmpa_int 1
1418     }
1419     \pgfpicture
1420     \@@_qpoint:n { row - 1 }
1421     \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1422     \@@_qpoint:n { row - \@@_succ:n \c@iRow }

```

```

1423 \dim_gadd:Nn \g_tmpa_dim \pgf@y
1424 \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).

```

1425 @@_qpoint:n { row - \int_use:N \l_tmpa_int - base }
1426 \dim_gsub:Nn \g_tmpa_dim \pgf@y

```

We take into account the position of the mathematical axis.

```

1427 \dim_gsub:Nn \g_tmpa_dim { \fontdimen22 \textfont2 }

```

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

```

1428 \endpgfpicture
1429 \box_move_up:nn \g_tmpa_dim { \box_use_drop:N \l_tmpa_box }
1430 \box_use_drop:N \l_tmpa_box
1431 }

```

```

1432 \cs_new_protected:Npn @@_use_array_box_with_notes:
1433 {
1434   \int_compare:nNnTF \c@tabularnote = 0
1435   { \box_use_drop:N \l_@@_the_array_box }
1436   {
1437     \begin { minipage } { \box_wd:N \l_@@_the_array_box }
1438     \box_use_drop:N \l_@@_the_array_box
1439     \skip_vertical:N 0.65ex

```

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

```

1440 \group_begin:
1441 \l_@@_notes_code_before_tl

```

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.

```

1442 \bool_if:NTF \l_@@_notes_para_bool
1443 {
1444   \begin { tabularnotes* }
1445   \seq_map_inline:Nn \g_@@_tabularnotes_seq { \item ##1 } \strut
1446   \end { tabularnotes* }

```

The following `\par` is mandatory for the event that the user has put `\footnotesize` (for example) in the `notes/code-before`.

```

1447 \par
1448 }
1449 {
1450   \tabularnotes
1451   \seq_map_inline:Nn \g_@@_tabularnotes_seq { \item ##1 } \strut
1452   \endtabularnotes
1453 }
1454 \unskip
1455 \group_end:
1456 \bool_if:NT \l_@@_notes_bottomrule_bool
1457 {
1458   \bool_if:NTF \c_@@_booktabs_loaded_bool
1459   {

```

The two dimensions `\aboverulesep` et `\heavyrulewidth` are parameters defined by `booktabs`.

```

1460 \skip_vertical:N \aboverulesep
\CT@arc@ is the specification of color defined by colortbl but you use it even if colortbl is not loaded.
1461 { \CT@arc@ \hrule height \heavyrulewidth }
1462 }
1463 { @@_error:n { bottomule~without~booktabs } }
1464 }
1465 \l_@@_notes_code_after_tl
1466 \end { minipage }
1467 \seq_gclear:N \g_@@_tabularnotes_seq
1468 \int_gzero:N \c@tabularnote
1469 }
1470 }

```

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.

```
1471 \cs_new_protected:Npn \@@_put_box_in_flow_bis:nn #1 #2
1472 {
```

We will compute the real width of both delimiters used.

```
1473   \dim_zero_new:N \l_@@_real_left_delim_dim
1474   \dim_zero_new:N \l_@@_real_right_delim_dim
1475   \hbox_set:Nn \l_tmpb_box
1476   {
1477     \c_math_toggle_token
1478     \left #1
1479     \vcenter
1480     {
1481       \vbox_to_ht:nn
1482       { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1483       { }
1484     }
1485     \right .
1486     \c_math_toggle_token
1487   }
1488   \dim_set:Nn \l_@@_real_left_delim_dim
1489   { \box_wd:N \l_tmpb_box - \nulldelimiterspace }
1490   \hbox_set:Nn \l_tmpb_box
1491   {
1492     \c_math_toggle_token
1493     \left .
1494     \vbox_to_ht:nn
1495     { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1496     { }
1497     \right #2
1498     \c_math_toggle_token
1499   }
1500   \dim_set:Nn \l_@@_real_right_delim_dim
1501   { \box_wd:N \l_tmpb_box - \nulldelimiterspace }
```

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

```
1502   \skip_horizontal:N \l_@@_left_delim_dim
1503   \skip_horizontal:N -\l_@@_real_left_delim_dim
1504   \@@_put_box_in_flow:
1505   \skip_horizontal:N \l_@@_right_delim_dim
1506   \skip_horizontal:N -\l_@@_real_right_delim_dim
1507 }
```

The construction of the array in the environment `{NiceArrayWithDelims}` is, in fact, done by the environment `{@@-light-syntax}` or by the environment `{@@-normal-syntax}` (whether the option `light-syntax` is used or not). When the key `light-syntax` is not used, the construction is a standard environment (and, thus, it's possible to use verbatim in the array).

```
1508 \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).

```
1509 {
1510   \peek_meaning_ignore_spaces:NTF \end
1511   { \@@_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`).

```
1512   { \exp_args:NV \@@_array: \l_tmpa_tl }
1513 }
1514 {
```

```

1515 \@@_create_col_nodes:
1516 \endarray
1517 }

```

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

```

1518 \NewDocumentEnvironment { @@-light-syntax } { b }
1519 {

```

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`.

```

1520 \tl_if_empty:nT { #1 } { \@@_fatal:n { empty-environment } }
1521 \tl_map_inline:nn { #1 }
1522 {
1523   \tl_if_eq:nnT { ##1 } { & }
1524   { \@@_fatal:n { ampersand-in-light-syntax } }
1525   \tl_if_eq:nnT { ##1 } { \ }
1526   { \@@_fatal:n { double-backslash-in-light-syntax } }
1527 }

```

Now, you extract the `code-after` or the body of the environment. Maybe, there is no command `\CodeAfter` in the body. That's why you put a marker `\CodeAfter` after `#1`. If there is yet a `\CodeAfter` in `#1`, this second (or third...) `\CodeAfter` will be caught in the value of `\g_@@_code_after_tl`. That doesn't matter because `\CodeAfter` will be set to *no-op* before the execution of `\g_@@_code_after_tl`.

```

1528 \@@_light_syntax_i #1 \CodeAfter \q_stop
1529 }

```

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`.

```

1530 { }
1531 \cs_new_protected:Npn \@@_light_syntax_i #1\CodeAfter #2\q_stop
1532 {
1533   \tl_gput_right:Nn \g_@@_code_after_tl { #2 }

```

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

```

1534 \seq_gclear_new:N \g_@@_rows_seq
1535 \tl_set_rescan:Nno \l_@@_end_of_row_tl { } \l_@@_end_of_row_tl
1536 \exp_args:NNV \seq_gset_split:Nnn \g_@@_rows_seq \l_@@_end_of_row_tl { #1 }

```

If the environment uses the option `last-row` without value (i.e. without saying the number of the rows), we have now the opportunity to know that value. We do it, and so, if the token list `\l_@@_code_for_last_row_tl` is not empty, we will use directly where it should be.

```

1537 \int_compare:nNnT \l_@@_last_row_int = { -1 }
1538 { \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`).

```

1539 \exp_args:NV \@@_array: \l_tmpa_tl

```

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

```

1540 \seq_gpop_left:NN \g_@@_rows_seq \l_tmpa_tl
1541 \exp_args:NV \@@_line_with_light_syntax_i:n \l_tmpa_tl
1542 \seq_map_function:NN \g_@@_rows_seq \@@_line_with_light_syntax:n
1543 \@@_create_col_nodes:
1544 \endarray
1545 }
1546 \cs_new_protected:Npn \@@_line_with_light_syntax:n #1
1547 { \tl_if_empty:nF { #1 } { \ \ \@@_line_with_light_syntax_i:n { #1 } } }
1548 \cs_new_protected:Npn \@@_line_with_light_syntax_i:n #1
1549 {
1550   \seq_gclear_new:N \g_@@_cells_seq
1551   \seq_gset_split:Nnn \g_@@_cells_seq { ~ } { #1 }

```

```

1552 \seq_gpop_left:Nn \g_@@_cells_seq \l_tmpa_tl
1553 \l_tmpa_tl
1554 \seq_map_inline:Nn \g_@@_cells_seq { & ##1 }
1555 }

```

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).

```

1556 \cs_new_protected:Npn \@@_analyze_end:Nn #1 #2
1557 {
1558   \str_if_eq:VnT \g_@@_name_env_str { #2 }
1559   { \@@_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.

```

1560   \end { #2 }
1561 }

```

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

```

1562 \cs_new:Npn \@@_create_col_nodes:
1563 {
1564   \crrc
1565   \int_compare:nNnT \l_@@_first_col_int = 0
1566   {
1567     \omit
1568     \skip_horizontal:N -2\col@sep
1569     \bool_if:NT \l_@@_code_before_bool
1570     { \pgfsys@markposition { \@@_env: - col - 0 } }
1571     \pgfpicture
1572     \pgfrememberpicturerepositiononpagetrue
1573     \pgfcoordinate { \@@_env: - col - 0 } \pgfpintorigin
1574     \str_if_empty:NF \l_@@_name_str
1575     { \pgfnodealias { \l_@@_name_str - col - 0 } { \@@_env: - col - 0 } }
1576     \endpgfpicture
1577     &
1578   }
1579   \omit

```

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

```

1580   \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`).

```

1581   \int_compare:nNnTF \l_@@_first_col_int = 0
1582   {
1583     \bool_if:NT \l_@@_code_before_bool
1584     {
1585       \hbox
1586       {
1587         \skip_horizontal:N -0.5\arrayrulewidth
1588         \pgfsys@markposition { \@@_env: - col - 1 }
1589         \skip_horizontal:N 0.5\arrayrulewidth
1590       }
1591     }
1592     \pgfpicture
1593     \pgfrememberpicturerepositiononpagetrue
1594     \pgfcoordinate { \@@_env: - col - 1 }
1595     { \pgfpint { - 0.5 \arrayrulewidth } \c_zero_dim }
1596     \str_if_empty:NF \l_@@_name_str
1597     { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
1598     \endpgfpicture
1599   }
1600   {

```

```

1601 \bool_if:NT \l_@@_code_before_bool
1602 {
1603   \hbox
1604   {
1605     \skip_horizontal:N 0.5 \arrayrulewidth
1606     \pgfsys@markposition { \@@_env: - col - 1 }
1607     \skip_horizontal:N -0.5\arrayrulewidth
1608   }
1609 }
1610 \pgfpicture
1611 \pgfrememberpicturepositiononpagetrue
1612 \pgfcoordinate { \@@_env: - col - 1 }
1613 { \pgfpoint { 0.5 \arrayrulewidth } \c_zero_dim }
1614 \str_if_empty:NF \l_@@_name_str
1615 { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
1616 \endpgfpicture
1617 }

```

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.

```

1618 \skip_gset:Nn \g_tmpa_skip { 0 pt~plus 1 fill }
1619 \bool_if:NF \l_@@_auto_columns_width_bool
1620 { \dim_compare:nNnT \l_@@_columns_width_dim > \c_zero_dim }
1621 {
1622   \bool_lazy_and:nnTF
1623     \l_@@_auto_columns_width_bool
1624     { \bool_not_p:n \l_@@_block_auto_columns_width_bool }
1625     { \skip_gset_eq:NN \g_tmpa_skip \g_@@_max_cell_width_dim }
1626     { \skip_gset_eq:NN \g_tmpa_skip \l_@@_columns_width_dim }
1627   \skip_gadd:Nn \g_tmpa_skip { 2 \col@sep }
1628 }
1629 \skip_horizontal:N \g_tmpa_skip
1630 \hbox
1631 {
1632   \bool_if:NT \l_@@_code_before_bool
1633   {
1634     \hbox
1635     {
1636       \skip_horizontal:N -0.5\arrayrulewidth
1637       \pgfsys@markposition { \@@_env: - col - 2 }
1638       \skip_horizontal:N 0.5\arrayrulewidth
1639     }
1640   }
1641   \pgfpicture
1642   \pgfrememberpicturepositiononpagetrue
1643   \pgfcoordinate { \@@_env: - col - 2 }
1644   { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
1645   \str_if_empty:NF \l_@@_name_str
1646   { \pgfnodealias { \l_@@_name_str - col - 2 } { \@@_env: - col - 2 } }
1647   \endpgfpicture
1648 }

```

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.

```

1649 \int_gset:Nn \g_tmpa_int 1
1650 \bool_if:NNTF \g_@@_last_col_found_bool
1651 { \prg_replicate:nn { \g_@@_col_total_int - 2 } }
1652 { \prg_replicate:nn { \g_@@_col_total_int - 1 } }
1653 {
1654   &

```



```

1655 \omit
The incrementation of the counter \g_tmpa_int must be done after the \omit of the cell.
1656 \int_gincr:N \g_tmpa_int
1657 \skip_horizontal:N \g_tmpa_skip
1658 \bool_if:NT \l_@@_code_before_bool
1659 {
1660     \hbox
1661     {
1662         \skip_horizontal:N -0.5\arrayrulewidth
1663         \pgfsys@markposition { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1664         \skip_horizontal:N 0.5\arrayrulewidth
1665     }
1666 }

We create the col node on the right of the current column.
1667 \pgfpicture
1668 \pgfrememberpicturepositiononpagetrue
1669 \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1670 { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
1671 \str_if_empty:NF \l_@@_name_str
1672 {
1673     \pgfnodealias
1674     { \l_@@_name_str - col - \@@_succ:n \g_tmpa_int }
1675     { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1676 }
1677 \endpgfpicture
1678 }
1679 \bool_if:NT \g_@@_last_col_found_bool
1680 {
1681     \bool_if:NT \l_@@_code_before_bool
1682     {
1683         \pgfsys@markposition { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
1684     }
1685     \skip_horizontal:N 2\col@sep
1686     \pgfpicture
1687     \pgfrememberpicturepositiononpagetrue
1688     \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
1689     \pgfpintorigin
1690     \str_if_empty:NF \l_@@_name_str
1691     {
1692         \pgfnodealias
1693         { \l_@@_name_str - col - \@@_succ:n \g_@@_col_total_int }
1694         { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
1695     }
1696     \endpgfpicture
1697     \skip_horizontal:N -2\col@sep
1698 }
1699 \cr
1700 }

```

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

```

1701 \tl_const:Nn \c_@@_preamble_first_col_tl
1702 {
1703     >
1704     {
1705         \@@_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.

```

1706     \hbox_set:Nw \l_@@_cell_box
1707     \@@_math_toggle_token:
1708     \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”.

```

1709     \bool_lazy_and:nnT
1710     { \int_compare_p:nNn \c@iRow > 0 }
1711     {
1712         \bool_lazy_or:p:nn
1713         { \int_compare_p:nNn \l_@@_last_row_int < 0 }
1714         { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
1715     }
1716     {
1717         \l_@@_code_for_first_col_tl
1718         \xglobal \colorlet { nicematrix-first-col } { . }
1719     }
1720 }

```

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`.

```

1721     l
1722     <
1723     {
1724         \@@_math_toggle_token:
1725         \hbox_set_end:
1726         \@@_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.

```

1727     \dim_gset:Nn \g_@@_width_first_col_dim
1728     { \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.

```

1729     \hbox_overlap_left:n
1730     {
1731         \dim_compare:nNnTF { \box_wd:N \l_@@_cell_box } > \c_zero_dim
1732         \@@_node_for_the_cell:
1733         { \box_use_drop:N \l_@@_cell_box }
1734         \skip_horizontal:N \l_@@_left_delim_dim
1735         \skip_horizontal:N \l_@@_left_margin_dim
1736         \skip_horizontal:N \l_@@_extra_left_margin_dim
1737     }
1738     \skip_horizontal:N -2\col@sep
1739 }
1740 }

```

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

```

1741 \tl_const:Nn \c_@@_preamble_last_col_tl
1742 {
1743     >
1744     {

```

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

```

1745     \bool_gset_true:N \g_@@_last_col_found_bool
1746     \int_gincr:N \c@jCol
1747     \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.

```

1748     \hbox_set:Nw \l_@@_cell_box
1749     \@@_math_toggle_token:
1750     \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”.

```

1751     \int_compare:nNnT \c@iRow > 0
1752     {
1753         \bool_lazy_or:nnT
1754         { \int_compare_p:nNn \l_@@_last_row_int < 0 }
1755         { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
1756     }

```

```

1757         \l_@@_code_for_last_col_tl
1758         \xglobal \colorlet { nicematrix-last-col } { . }
1759     }
1760 }
1761 }
1762 1
1763 <
1764 {
1765     \@@_math_toggle_token:
1766     \hbox_set_end:
1767     \@@_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.

```

1768     \dim_gset:Nn \g_@@_width_last_col_dim
1769     { \dim_max:nn \g_@@_width_last_col_dim { \box_wd:N \l_@@_cell_box } }
1770     \skip_horizontal:N -2\col@sep

```

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

```

1771     \hbox_overlap_right:n
1772     {
1773         \dim_compare:nNnT { \box_wd:N \l_@@_cell_box } > \c_zero_dim
1774         {
1775             \skip_horizontal:N \l_@@_right_delim_dim
1776             \skip_horizontal:N \l_@@_right_margin_dim
1777             \skip_horizontal:N \l_@@_extra_right_margin_dim
1778             \@@_node_for_the_cell:
1779         }
1780     }
1781 }
1782 }

```

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.

```

1783 \NewDocumentEnvironment { NiceArray } { }
1784 {
1785     \bool_set_true:N \l_@@_NiceArray_bool
1786     \str_if_empty:NT \g_@@_name_env_str
1787     { \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).

```

1788     \NiceArrayWithDelims . .
1789 }
1790 { \endNiceArrayWithDelims }

```

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

```

1791 \NewDocumentEnvironment { pNiceArray } { }
1792 {
1793     \str_if_empty:NT \g_@@_name_env_str
1794     { \str_gset:Nn \g_@@_name_env_str { pNiceArray } }
1795     \@@_test_if_math_mode:
1796     \NiceArrayWithDelims ( )
1797 }
1798 { \endNiceArrayWithDelims }

1799 \NewDocumentEnvironment { bNiceArray } { }
1800 {
1801     \str_if_empty:NT \g_@@_name_env_str
1802     { \str_gset:Nn \g_@@_name_env_str { bNiceArray } }
1803     \@@_test_if_math_mode:
1804     \NiceArrayWithDelims [ ]
1805 }

```

```

1806 { \endNiceArrayWithDelims }
1807 \NewDocumentEnvironment { BNiceArray } { }
1808 {
1809   \str_if_empty:NT \g_@@_name_env_str
1810   { \str_gset:Nn \g_@@_name_env_str { BNiceArray } }
1811   \@@_test_if_math_mode:
1812   \NiceArrayWithDelims {\ }
1813 }
1814 { \endNiceArrayWithDelims }
1815 \NewDocumentEnvironment { vNiceArray } { }
1816 {
1817   \str_if_empty:NT \g_@@_name_env_str
1818   { \str_gset:Nn \g_@@_name_env_str { vNiceArray } }
1819   \@@_test_if_math_mode:
1820   \NiceArrayWithDelims | |
1821 }
1822 { \endNiceArrayWithDelims }
1823 \NewDocumentEnvironment { VNiceArray } { }
1824 {
1825   \str_if_empty:NT \g_@@_name_env_str
1826   { \str_gset:Nn \g_@@_name_env_str { VNiceArray } }
1827   \@@_test_if_math_mode:
1828   \NiceArrayWithDelims \ | \ |
1829 }
1830 { \endNiceArrayWithDelims }

```

The environment `{NiceMatrix}` and its variants

```

1831 \cs_new_protected:Npn \@@_define_env:n #1
1832 {
1833   \NewDocumentEnvironment { #1 NiceMatrix } { ! 0 { } }
1834   {
1835     \str_gset:Nn \g_@@_name_env_str { #1 NiceMatrix }
1836     \tl_set:Nn \l_@@_type_of_col_tl c
1837     \keys_set:nn { NiceMatrix / NiceMatrix } { ##1 }
1838     \exp_args:Nnx \@@_begin_of_NiceMatrix:nn { #1 } \l_@@_type_of_col_tl
1839   }
1840   { \use:c { end #1 NiceArray } }
1841 }
1842 \cs_new_protected:Npn \@@_begin_of_NiceMatrix:nn #1 #2
1843 {
1844   \use:c { #1 NiceArray }
1845   {
1846     *
1847     {
1848       \int_compare:nNnTF \l_@@_last_col_int < 0
1849       \c@MaxMatrixCols
1850       { \@@_pred:n \l_@@_last_col_int }
1851     }
1852     #2
1853   }
1854 }
1855 \@@_define_env:n { }
1856 \@@_define_env:n p
1857 \@@_define_env:n b
1858 \@@_define_env:n B
1859 \@@_define_env:n v
1860 \@@_define_env:n V

```

The environment `{NiceTabular}`

```

1861 \NewDocumentEnvironment { NiceTabular } { 0 { } m ! 0 { } }
1862 {
1863   \str_gset:Nn \g_@@_name_env_str { NiceTabular }
1864   \keys_set:nn { NiceMatrix / NiceTabular } { #1 , #3 }
1865   \bool_set_true:N \l_@@_NiceTabular_bool
1866   \NiceArray { #2 }
1867 }
1868 { \endNiceArray }

```

After the construction of the array

```

1869 \cs_new_protected:Npn \@@_after_array:
1870 {
1871   \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.

```

1872   \bool_if:NT \g_@@_last_col_found_bool
1873   { \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`.

```

1874   \bool_if:NT \l_@@_last_col_without_value_bool
1875   {
1876     \dim_set_eq:NN \l_@@_last_col_int \g_@@_col_total_int
1877     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
1878     \iow_shipout:Nx \@mainaux
1879     {
1880       \cs_gset:cpn { @@_last_col_ \int_use:N \g_@@_env_int }
1881       { \int_use:N \g_@@_col_total_int }
1882     }
1883     \str_if_empty:NF \l_@@_name_str
1884     {
1885       \iow_shipout:Nx \@mainaux
1886       {
1887         \cs_gset:cpn { @@_last_col_ \l_@@_name_str }
1888         { \int_use:N \g_@@_col_total_int }
1889       }
1890     }
1891     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
1892   }

```

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.

```

1893   \bool_if:NT \l_@@_last_row_without_value_bool
1894   {
1895     \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.

```

1896   \bool_if:NF \l_@@_light_syntax_bool
1897   {
1898     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
1899     \iow_shipout:Nx \@mainaux
1900     {
1901       \cs_gset:cpn { @@_last_row_ \int_use:N \g_@@_env_int }
1902       { \int_use:N \g_@@_row_total_int }
1903     }

```

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.

```

1904     \str_if_empty:NF \l_@@_name_str
1905     {
1906         \iow_shipout:Nx \@mainaux
1907         {
1908             \cs_gset:cpn { @@_last_row_ \l_@@_name_str }
1909             { \int_use:N \g_@@_row_total_int }
1910         }
1911     }
1912     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
1913 }
1914 }

```

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

```

1915     \bool_if:NT \l_@@_code_before_bool
1916     {
1917         \iow_now:Nn \@mainaux \ExplSyntaxOn
1918         \iow_now:Nx \@mainaux
1919         { \seq_clear_new:c { @@_size _ \int_use:N \g_@@_env_int _ seq } }
1920         \iow_now:Nx \@mainaux
1921         {
1922             \seq_gset_from_clist:cn { @@_size _ \int_use:N \g_@@_env_int _ seq }
1923             {
1924                 \int_use:N \l_@@_first_row_int ,
1925                 \int_use:N \g_@@_row_total_int ,
1926                 \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.

```

1927         \bool_lazy_and:nnTF
1928         { \int_compare_p:nNn \l_@@_last_col_int > { -2 } }
1929         { \bool_not_p:n \g_@@_last_col_found_bool }
1930         \@@_succ:n
1931         \int_use:N
1932         \g_@@_col_total_int
1933     }
1934 }
1935     \iow_now:Nn \@mainaux \ExplSyntaxOff
1936 }

```

By default, the diagonal lines will be parallelized⁴¹. 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.

```

1937     \bool_if:NT \l_@@_parallelize_diags_bool
1938     {
1939         \int_gzero_new:N \g_@@_ddots_int
1940         \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.

```

1941         \dim_gzero_new:N \g_@@_delta_x_one_dim
1942         \dim_gzero_new:N \g_@@_delta_y_one_dim
1943         \dim_gzero_new:N \g_@@_delta_x_two_dim
1944         \dim_gzero_new:N \g_@@_delta_y_two_dim
1945     }
1946     \bool_if:nTF \l_@@_medium_nodes_bool
1947     {
1948         \bool_if:NTF \l_@@_large_nodes_bool
1949         \@@_create_medium_and_large_nodes:
1950         \@@_create_medium_nodes:
1951     }
1952     { \bool_if:NT \l_@@_large_nodes_bool \@@_create_large_nodes: }

```

⁴¹It's possible to use the option parallelize-diags to disable this parallelization.

```

1953 \int_zero_new:N \l_@@_initial_i_int
1954 \int_zero_new:N \l_@@_initial_j_int
1955 \int_zero_new:N \l_@@_final_i_int
1956 \int_zero_new:N \l_@@_final_j_int
1957 \bool_set_false:N \l_@@_initial_open_bool
1958 \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.

```

1959 \bool_if:NT \l_@@_small_bool
1960 {
1961     \dim_set:Nn \l_@@_radius_dim { 0.37 pt }
1962     \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.

```

1963     \dim_set:Nn \l_@@_xdots_shorten_dim { 0.6 \l_@@_xdots_shorten_dim }
1964 }

```

Now, we actually draw the dotted lines.

```

1965 \@@_draw_dotted_lines:
1966 \bool_if:NTF \l_@@_hvlines_bool
1967 \@@_draw_hvlines:
1968 {
1969     \bool_if:NT \l_@@_hlines_bool \@@_draw_hlines:
1970     \bool_if:NT \l_@@_vlines_bool \@@_draw_vlines:
1971     \bool_if:NT \l_@@_hvlines_except_corners_bool
1972         \@@_draw_hvlines_except_corners:
1973 }

```

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

```

1974 \cs_set_eq:NN \ialign \@@_old_ialign:
1975 \seq_if_empty:NF \g_@@_blocks_seq \@@_draw_blocks:
1976 \g_@@_internal_code_after_tl
1977 \tl_gclear:N \g_@@_internal_code_after_tl
1978 \bool_if:NT \c_@@_tikz_loaded_bool
1979 {
1980     \tikzset
1981     {
1982         every~picture / .style =
1983         {
1984             overlay ,
1985             remember~picture ,
1986             name~prefix = \@@_env: -
1987         }
1988     }
1989 }
1990 \cs_set_eq:NN \line \@@_line

```

When `light-syntax` is used, we insert systematically a `\CodeAfter` in the flow. Thus, it's possible to have two instructions `\CodeAfter` and the second one may be present in `\g_@@_code_after_tl`. That's why we set `\Code-after` to be *no-op* now.

```

1991 \cs_set_eq:NN \CodeAfter \prg_do_nothing:

```

And here's the code-after:

```

1992 \g_@@_code_after_tl
1993 \tl_gclear:N \g_@@_code_after_tl
1994 \group_end:
1995 \str_gclear:N \g_@@_name_env_str
1996 \@@_restore_iRow_jCol:

```

The command `\CT@arc@` contains the instruction of color for the rules of the array⁴². This command is used by `\CT@arc@` but we use it also for compatibility with `colortbl`. But we want also to be able

⁴²e.g. `\color[rgb]{0.5,0.5,0}`)

to use color for the rules of the array when `colortbl` is *not* loaded. That's why we do the following instruction which is in the patch of the end of arrays done by `colortbl`.

```
1997 \cs_gset_eq:NN \CT@arc@ \@@_old_CT@arc@
1998 }
```

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).

```
1999 \AtBeginDocument
2000 {
2001   \cs_new_protected:Npx \@@_draw_dotted_lines:
2002   {
2003     \c_@@_pgfortikzpicture_tl
2004     \@@_draw_dotted_lines_i:
2005     \c_@@_endpgfortikzpicture_tl
2006   }
2007 }
```

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

```
2008 \cs_new_protected:Npn \@@_draw_dotted_lines_i:
2009 {
2010   \pgfrememberpicturepositiononpagetrue
2011   \pgf@relevantforpicturesizefalse
2012   \g_@@_HVdotsfor_lines_tl
2013   \g_@@_Vdots_lines_tl
2014   \g_@@_Ddots_lines_tl
2015   \g_@@_Iddots_lines_tl
2016   \g_@@_Cdots_lines_tl
2017   \g_@@_Ldots_lines_tl
2018 }

2019 \cs_new_protected:Npn \@@_restore_iRow_jCol:
2020 {
2021   \cs_if_exist:NT \theiRow { \int_gset_eq:NN \c@iRow \l_@@_old_iRow_int }
2022   \cs_if_exist:NT \thejCol { \int_gset_eq:NN \c@jCol \l_@@_old_jCol_int }
2023 }
```

We draw the dotted lines

A dotted line will be said *open* in one of its extremities when it stops on the edge of the matrix and *closed* otherwise. In the following matrix, the dotted line is closed on its left extremity and open on its right.

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

The command `\@@_find_extremities_of_line:nnnn` takes four arguments:

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

This command computes:

- `\l_@@_initial_i_int` and `\l_@@_initial_j_int` which are the coordinates of one extremity of the line;

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

```
2024 \cs_new_protected:Npn \l_@@_find_extremities_of_line:nnnn #1 #2 #3 #4
2025 {
```

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

```
2026 \cs_set:cpn { @@ _ dotted _ #1 - #2 } { }
```

Initialization of variables.

```
2027 \int_set:Nn \l_@@_initial_i_int { #1 }
2028 \int_set:Nn \l_@@_initial_j_int { #2 }
2029 \int_set:Nn \l_@@_final_i_int { #1 }
2030 \int_set:Nn \l_@@_final_j_int { #2 }
```

We will do two loops: one when determining the initial cell and the other when determining the final cell. The boolean `\l_@@_stop_loop_bool` will be used to control these loops. In the first loop, we search the “final” extremity of the line.

```
2031 \bool_set_false:N \l_@@_stop_loop_bool
2032 \bool_do_until:Nn \l_@@_stop_loop_bool
2033 {
2034   \int_add:Nn \l_@@_final_i_int { #3 }
2035   \int_add:Nn \l_@@_final_j_int { #4 }
```

We test if we are still in the matrix.

```
2036   \bool_set_false:N \l_@@_final_open_bool
2037   \int_compare:nNnTF \l_@@_final_i_int > \c@iRow
2038   {
2039     \int_compare:nNnTF { #3 } = 1
2040     { \bool_set_true:N \l_@@_final_open_bool }
2041     {
2042       \int_compare:nNnT \l_@@_final_j_int > \c@jCol
2043       { \bool_set_true:N \l_@@_final_open_bool }
2044     }
2045   }
2046   {
2047     \int_compare:nNnTF \l_@@_final_j_int < 1
2048     {
2049       \int_compare:nNnT { #4 } = { -1 }
2050       { \bool_set_true:N \l_@@_final_open_bool }
2051     }
2052     {
2053       \int_compare:nNnT \l_@@_final_j_int > \c@jCol
2054       {
2055         \int_compare:nNnT { #4 } = 1
2056         { \bool_set_true:N \l_@@_final_open_bool }
2057       }
2058     }
2059   }
2060   \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.

```
2061 {
```

We do a step backwards.

```
2062   \int_sub:Nn \l_@@_final_i_int { #3 }
2063   \int_sub:Nn \l_@@_final_j_int { #4 }
2064   \bool_set_true:N \l_@@_stop_loop_bool
2065 }
```

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`.

```

2066     {
2067         \cs_if_exist:cTF
2068         {
2069             @@ _ dotted _
2070             \int_use:N \l_@@_final_i_int -
2071             \int_use:N \l_@@_final_j_int
2072         }
2073         {
2074             \int_sub:Nn \l_@@_final_i_int { #3 }
2075             \int_sub:Nn \l_@@_final_j_int { #4 }
2076             \bool_set_true:N \l_@@_final_open_bool
2077             \bool_set_true:N \l_@@_stop_loop_bool
2078         }
2079         {
2080             \cs_if_exist:cTF
2081             {
2082                 pgf @ sh @ ns @ \@@_env:
2083                 - \int_use:N \l_@@_final_i_int
2084                 - \int_use:N \l_@@_final_j_int
2085             }
2086             { \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.

```

2087     {
2088         \cs_set:cpn
2089         {
2090             @@ _ dotted _
2091             \int_use:N \l_@@_final_i_int -
2092             \int_use:N \l_@@_final_j_int
2093         }
2094         { }
2095     }
2096 }
2097 }
2098 }

```

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

```

2099     \bool_set_false:N \l_@@_stop_loop_bool
2100     \bool_do_until:Nn \l_@@_stop_loop_bool
2101     {
2102         \int_sub:Nn \l_@@_initial_i_int { #3 }
2103         \int_sub:Nn \l_@@_initial_j_int { #4 }
2104         \bool_set_false:N \l_@@_initial_open_bool
2105         \int_compare:nNnTF \l_@@_initial_i_int < 1
2106         {
2107             \int_compare:nNnTF { #3 } = 1
2108             { \bool_set_true:N \l_@@_initial_open_bool }
2109             {
2110                 \int_compare:nNnT \l_@@_initial_j_int = 0
2111                 { \bool_set_true:N \l_@@_initial_open_bool }
2112             }
2113         }
2114         {
2115             \int_compare:nNnTF \l_@@_initial_j_int < 1

```

```

2116     {
2117         \int_compare:nNt { #4 } = 1
2118         { \bool_set_true:N \l_@@_initial_open_bool }
2119     }
2120     {
2121         \int_compare:nNt \l_@@_initial_j_int > \c@jCol
2122         {
2123             \int_compare:nNt { #4 } = { -1 }
2124             { \bool_set_true:N \l_@@_initial_open_bool }
2125         }
2126     }
2127 }
2128 \bool_if:NTF \l_@@_initial_open_bool
2129 {
2130     \int_add:Nn \l_@@_initial_i_int { #3 }
2131     \int_add:Nn \l_@@_initial_j_int { #4 }
2132     \bool_set_true:N \l_@@_stop_loop_bool
2133 }
2134 {
2135     \cs_if_exist:cTF
2136     {
2137         @@ _ dotted _
2138         \int_use:N \l_@@_initial_i_int -
2139         \int_use:N \l_@@_initial_j_int
2140     }
2141     {
2142         \int_add:Nn \l_@@_initial_i_int { #3 }
2143         \int_add:Nn \l_@@_initial_j_int { #4 }
2144         \bool_set_true:N \l_@@_initial_open_bool
2145         \bool_set_true:N \l_@@_stop_loop_bool
2146     }
2147     {
2148         \cs_if_exist:cTF
2149         {
2150             pgf @ sh @ ns @ \@@_env:
2151             - \int_use:N \l_@@_initial_i_int
2152             - \int_use:N \l_@@_initial_j_int
2153         }
2154         { \bool_set_true:N \l_@@_stop_loop_bool }
2155         {
2156             \cs_set:cpn
2157             {
2158                 @@ _ dotted _
2159                 \int_use:N \l_@@_initial_i_int -
2160                 \int_use:N \l_@@_initial_j_int
2161             }
2162             { }
2163         }
2164     }
2165 }
2166 }

```

If the key `hvlines` is used, we remind the rectangle described by all the dotted lines in order to respect the corresponding virtual “block” when drawing the horizontal and vertical rules.

```

2167 \bool_if:NT \l_@@_hvlines_bool
2168 {
2169     \seq_gput_right:Nx \g_@@_pos_of_xdots_seq
2170     {
2171         { \int_use:N \l_@@_initial_i_int }
2172         { \int_use:N \l_@@_initial_j_int }
2173         { \int_use:N \l_@@_final_i_int }
2174         { \int_use:N \l_@@_final_j_int }
2175     }
2176 }

```

```

2177 }

2178 \cs_new_protected:Npn \@@_set_initial_coords:
2179 {
2180   \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2181   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2182 }
2183 \cs_new_protected:Npn \@@_set_final_coords:
2184 {
2185   \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2186   \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2187 }
2188 \cs_new_protected:Npn \@@_set_initial_coords_from_anchor:n #1
2189 {
2190   \pgfpointanchor
2191   {
2192     \@@_env:
2193     - \int_use:N \l_@@_initial_i_int
2194     - \int_use:N \l_@@_initial_j_int
2195   }
2196   { #1 }
2197   \@@_set_initial_coords:
2198 }
2199 \cs_new_protected:Npn \@@_set_final_coords_from_anchor:n #1
2200 {
2201   \pgfpointanchor
2202   {
2203     \@@_env:
2204     - \int_use:N \l_@@_final_i_int
2205     - \int_use:N \l_@@_final_j_int
2206   }
2207   { #1 }
2208   \@@_set_final_coords:
2209 }

```

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.

```

2210 \cs_new_protected:Npn \@@_draw_Ldots:nnn #1 #2 #3
2211 {
2212   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2213   {
2214     \@@_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.

```

2215   \group_begin:
2216   \int_compare:nNnTF { #1 } = 0
2217   { \color { nicematrix-first-row } }
2218   {

```

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.

```

2219       \int_compare:nNnT { #1 } = \l_@@_last_row_int
2220       { \color { nicematrix-last-row } }
2221   }
2222   \keys_set:nn { NiceMatrix / xdots } { #3 }
2223   \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2224   \@@_actually_draw_Ldots:
2225   \group_end:
2226 }
2227 }

```

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.

```

2228 \cs_new_protected:Npn \@@_actually_draw_Ldots:
2229 {
2230   \bool_if:NTF \l_@@_initial_open_bool
2231   {
2232     \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2233     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2234     \dim_add:Nn \l_@@_x_initial_dim \@@_tab_or_array_colsep:
2235     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int - base }
2236     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2237   }
2238   { \@@_set_initial_coords_from_anchor:n { base-east } }
2239   \bool_if:NTF \l_@@_final_open_bool
2240   {
2241     \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
2242     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2243     \dim_sub:Nn \l_@@_x_final_dim \@@_tab_or_array_colsep:
2244     \@@_qpoint:n { row - \int_use:N \l_@@_final_i_int - base }
2245     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2246   }
2247   { \@@_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 (?).

```

2248   \dim_add:Nn \l_@@_y_initial_dim \l_@@_radius_dim
2249   \dim_add:Nn \l_@@_y_final_dim \l_@@_radius_dim
2250   \@@_draw_line:
2251 }

```

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.

```

2252 \cs_new_protected:Npn \@@_draw_Cdots:nnn #1 #2 #3
2253 {
2254   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2255   {
2256     \@@_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.

```

2257   \group_begin:
2258     \int_compare:nNnTF { #1 } = 0
2259     { \color { nicematrix-first-row } }
2260     {

```

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.

```

2261       \int_compare:nNnT { #1 } = \l_@@_last_row_int
2262       { \color { nicematrix-last-row } }
2263     }
2264     \keys_set:nn { NiceMatrix / xdots } { #3 }
2265     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2266     \@@_actually_draw_Cdots:

```

```

2267     \group_end:
2268   }
2269 }

```

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.`

```

2270 \cs_new_protected:Npn \@@_actually_draw_Cdots:
2271 {
2272   \bool_if:NTF \l_@@_initial_open_bool
2273   {
2274     \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2275     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2276     \dim_add:Nn \l_@@_x_initial_dim
2277       { \bool_if:NTF \l_@@_NiceTabular_bool \tabcolsep \arraycolsep }
2278   }
2279   { \@@_set_initial_coords_from_anchor:n { mid~east } }
2280   \bool_if:NTF \l_@@_final_open_bool
2281   {
2282     \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
2283     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2284     \dim_sub:Nn \l_@@_x_final_dim
2285       { \bool_if:NTF \l_@@_NiceTabular_bool \tabcolsep \arraycolsep }
2286   }
2287   { \@@_set_final_coords_from_anchor:n { mid~west } }
2288   \bool_lazy_and:nnTF
2289     \l_@@_initial_open_bool
2290     \l_@@_final_open_bool
2291   {
2292     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2293     \dim_set_eq:NN \l_tmpa_dim \pgf@y
2294     \@@_qpoint:n { row - \@@_succ:n \l_@@_initial_i_int }
2295     \dim_set:Nn \l_@@_y_initial_dim { ( \l_tmpa_dim + \pgf@y ) / 2 }
2296     \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim
2297   }
2298   {
2299     \bool_if:NT \l_@@_initial_open_bool
2300     { \dim_set_eq:NN \l_@@_y_initial_dim \l_@@_y_final_dim }
2301     \bool_if:NT \l_@@_final_open_bool
2302     { \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim }
2303   }
2304   \@@_draw_line:
2305 }

```

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.

```

2306 \cs_new_protected:Npn \@@_draw_Vdots:nnn #1 #2 #3
2307 {
2308   \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2309   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2310   {
2311     \@@_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.

```

2312     \group_begin:
2313     \int_compare:nNnTF { #2 } = 0
2314     { \color { nicematrix-first-col } }
2315     {
2316         \int_compare:nNnT { #2 } = \l_@@_last_col_int
2317         { \color { nicematrix-last-col } }
2318     }
2319     \keys_set:nn { NiceMatrix / xdots } { #3 }
2320     \@@_actually_draw_Vdots:
2321 \group_end:
2322 }
2323 }
```

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`.

```

2324 \cs_new_protected:Npn \@@_actually_draw_Vdots:
2325 {
```

The boolean `\l_tmpa_bool` indicates whether the column is of type 1 or may be considered as if.

```

2326     \bool_set_false:N \l_tmpa_bool
2327     \bool_lazy_or:nnF \l_@@_initial_open_bool \l_@@_final_open_bool
2328     {
2329         \@@_set_initial_coords_from_anchor:n { south-west }
2330         \@@_set_final_coords_from_anchor:n { north-west }
2331         \bool_set:Nn \l_tmpa_bool
2332         { \dim_compare_p:nNn \l_@@_x_initial_dim = \l_@@_x_final_dim }
2333     }
```

Now, we try to determine whether the column is of type c or may be considered as if.

```

2334     \bool_if:NTF \l_@@_initial_open_bool
2335     {
2336         \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2337         \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2338     }
2339     { \@@_set_initial_coords_from_anchor:n { south } }
2340     \bool_if:NTF \l_@@_final_open_bool
2341     {
2342         \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }
2343         \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2344     }
2345     { \@@_set_final_coords_from_anchor:n { north } }
2346     \bool_if:NTF \l_@@_initial_open_bool
2347     {
2348         \bool_if:NTF \l_@@_final_open_bool
2349         {
2350             \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2351             \dim_set_eq:NN \l_tmpa_dim \pgf@x
2352             \@@_qpoint:n { col - \@@_succ:n \l_@@_initial_j_int }
2353             \dim_set:Nn \l_@@_x_initial_dim { ( \pgf@x + \l_tmpa_dim ) / 2 }
2354             \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.

```

2355     \int_compare:nNnT \l_@@_last_col_int > { -2 }
2356     {
2357         \int_compare:nNnT \l_@@_initial_j_int = \g_@@_col_total_int
2358         {
2359             \dim_set_eq:NN \l_tmpa_dim \l_@@_right_margin_dim
2360             \dim_add:Nn \l_tmpa_dim \l_@@_extra_right_margin_dim
2361             \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
2362             \dim_add:Nn \l_@@_x_final_dim \l_tmpa_dim
2363         }
2364     }
2365 }
2366 { \dim_set_eq:NN \l_@@_x_initial_dim \l_@@_x_final_dim }
2367 }
2368 {
2369     \bool_if:NTF \l_@@_final_open_bool
2370     { \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim }
2371     {

```

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.

```

2372     \dim_compare:nNnF \l_@@_x_initial_dim = \l_@@_x_final_dim
2373     {
2374         \dim_set:Nn \l_@@_x_initial_dim
2375         {
2376             \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
2377             \l_@@_x_initial_dim \l_@@_x_final_dim
2378         }
2379         \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim
2380     }
2381 }
2382 }
2383 \@@_draw_line:
2384 }

```

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.

```

2385 \cs_new_protected:Npn \@@_draw_Ddots:nnn #1 #2 #3
2386 {
2387     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2388     {
2389         \@@_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.

```

2390     \group_begin:
2391     \keys_set:nn { NiceMatrix / xdots } { #3 }
2392     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2393     \@@_actually_draw_Ddots:
2394     \group_end:
2395 }
2396 }

```

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`.

```

2397 \cs_new_protected:Npn \@@_actually_draw_Ddots:
2398 {
2399   \bool_if:NTF \l_@@_initial_open_bool
2400   {
2401     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2402     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2403     \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2404     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2405   }
2406   { \@@_set_initial_coords_from_anchor:n { south-east } }
2407   \bool_if:NTF \l_@@_final_open_bool
2408   {
2409     \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }
2410     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2411     \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
2412     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2413   }
2414   { \@@_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.

```

2415   \bool_if:NT \l_@@_parallelize_diags_bool
2416   {
2417     \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).

```

2418     \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.

```

2419     {
2420       \dim_gset:Nn \g_@@_delta_x_one_dim
2421       { \l_@@_x_final_dim - \l_@@_x_initial_dim }
2422       \dim_gset:Nn \g_@@_delta_y_one_dim
2423       { \l_@@_y_final_dim - \l_@@_y_initial_dim }
2424     }

```

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`.

```

2425     {
2426       \dim_set:Nn \l_@@_y_final_dim
2427       {
2428         \l_@@_y_initial_dim +
2429         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2430         \dim_ratio:nn \g_@@_delta_y_one_dim \g_@@_delta_x_one_dim
2431       }
2432     }
2433   }
2434   \@@_draw_line:
2435 }

```

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.

```

2436 \cs_new_protected:Npn \@@_draw_Iddots:nnn #1 #2 #3
2437 {
2438   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2439   {
2440     \@@_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.

```

2441     \group_begin:
2442     \keys_set:nn { NiceMatrix / xdots } { #3 }
2443     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2444     \@@_actually_draw_Iddots:
2445     \group_end:
2446   }
2447 }

```

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`.

```

2448 \cs_new_protected:Npn \@@_actually_draw_Iddots:
2449 {
2450   \bool_if:NTF \l_@@_initial_open_bool
2451   {
2452     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2453     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2454     \@@_qpoint:n { col - \@@_succ:n \l_@@_initial_j_int }
2455     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2456   }
2457   { \@@_set_initial_coords_from_anchor:n { south-west } }
2458   \bool_if:NTF \l_@@_final_open_bool
2459   {
2460     \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }
2461     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2462     \@@_qpoint:n { col - \int_use:N \l_@@_final_j_int }
2463     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2464   }
2465   { \@@_set_final_coords_from_anchor:n { north-east } }
2466   \bool_if:NT \l_@@_parallelize_diags_bool
2467   {
2468     \int_gincr:N \g_@@_iddots_int
2469     \int_compare:nNnTF \g_@@_iddots_int = 1
2470     {
2471       \dim_gset:Nn \g_@@_delta_x_two_dim
2472       { \l_@@_x_final_dim - \l_@@_x_initial_dim }
2473       \dim_gset:Nn \g_@@_delta_y_two_dim
2474       { \l_@@_y_final_dim - \l_@@_y_initial_dim }
2475     }
2476     {
2477       \dim_set:Nn \l_@@_y_final_dim
2478       {
2479         \l_@@_y_initial_dim +
2480         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2481         \dim_ratio:nn \g_@@_delta_y_two_dim \g_@@_delta_x_two_dim
2482       }

```

```

2483     }
2484   }
2485   \@@_draw_line:
2486 }

```

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`

```

2487 \cs_new_protected:Npn \@@_draw_line:
2488 {
2489   \pgfrememberpicturepositiononpagetrue
2490   \pgf@relevantforpicturesizefalse
2491   \tl_if_eq:NNTF \l_@@_xdots_line_style_tl \c_@@_standard_tl
2492     \@@_draw_standard_dotted_line:
2493     \@@_draw_non_standard_dotted_line:
2494 }

```

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.

```

2495 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:
2496 {
2497   \begin { scope }
2498   \exp_args:No \@@_draw_non_standard_dotted_line:n
2499     { \l_@@_xdots_line_style_tl , \l_@@_xdots_color_tl }
2500 }

```

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.

```

2501 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:n #1
2502 {
2503   \draw
2504     [
2505       #1 ,
2506       shorten-> = \l_@@_xdots_shorten_dim ,
2507       shorten~< = \l_@@_xdots_shorten_dim ,
2508     ]
2509     ( \l_@@_x_initial_dim , \l_@@_y_initial_dim )
2510     -- node [ sloped , above ]
2511       { \c_math_toggle_token \scriptstyle \l_@@_xdots_up_tl \c_math_toggle_token }
2512     node [ sloped , below ]
2513       {
2514         \c_math_toggle_token
2515         \scriptstyle \l_@@_xdots_down_tl
2516         \c_math_toggle_token
2517       }
2518     ( \l_@@_x_final_dim , \l_@@_y_final_dim ) ;
2519   \end { scope }
2520 }

```

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

```
2521 \cs_new_protected:Npn \@@_draw_standard_dotted_line:
2522 {
```

First, we put the labels.

```
2523   \bool_lazy_and:nnF
2524   { \tl_if_empty_p:N \l_@@_xdots_up_tl }
2525   { \tl_if_empty_p:N \l_@@_xdots_down_tl }
2526   {
2527     \pgfscope
2528     \pgftransformshift
2529     {
2530       \pgfpointlineattime { 0.5 }
2531       { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
2532       { \pgfpoint \l_@@_x_final_dim \l_@@_y_final_dim }
2533     }
2534     \pgftransformrotate
2535     {
2536       \fp_eval:n
2537       {
2538         atand
2539         (
2540           \l_@@_y_final_dim - \l_@@_y_initial_dim ,
2541           \l_@@_x_final_dim - \l_@@_x_initial_dim
2542         )
2543       }
2544     }
2545     \pgfnode
2546     { rectangle }
2547     { south }
2548     {
2549       \c_math_toggle_token
2550       \scriptstyle \l_@@_xdots_up_tl
2551       \c_math_toggle_token
2552     }
2553     { }
2554     { \pgfusepath { } }
2555     \pgfnode
2556     { rectangle }
2557     { north }
2558     {
2559       \c_math_toggle_token
2560       \scriptstyle \l_@@_xdots_down_tl
2561       \c_math_toggle_token
2562     }
2563     { }
2564     { \pgfusepath { } }
2565     \endpgfscope
2566   }
2567   \pgfrememberpicturepositiononpagetrue
2568   \pgf@relevantforpicturesizefalse
2569   \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.

```
2570   \dim_zero_new:N \l_@@_l_dim
2571   \dim_set:Nn \l_@@_l_dim
2572   {
2573     \fp_to_dim:n
2574     {
2575       sqrt
2576       (
2577         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) ^ 2
```

```

2578      +
2579      ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) ^ 2
2580    )
2581  }
2582 }

```

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.

```

2583 \bool_lazy_or:nnF
2584 { \dim_compare_p:nNn { \dim_abs:n \l_@@_l_dim } > \c_@@_max_l_dim }
2585 { \dim_compare_p:nNn \l_@@_l_dim = \c_zero_dim }
2586 \@@_draw_standard_dotted_line_i:
2587 \group_end:
2588 }
2589 \dim_const:Nn \c_@@_max_l_dim { 50 cm }
2590 \cs_new_protected:Npn \@@_draw_standard_dotted_line_i:
2591 {

```

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

```

2592 \bool_if:NTF \l_@@_initial_open_bool
2593 {
2594   \bool_if:NTF \l_@@_final_open_bool
2595   {
2596     \int_set:Nn \l_tmpa_int
2597     { \dim_ratio:nn \l_@@_l_dim \l_@@_inter_dots_dim }
2598   }
2599   {
2600     \int_set:Nn \l_tmpa_int
2601     {
2602       \dim_ratio:nn
2603       { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
2604       \l_@@_inter_dots_dim
2605     }
2606   }
2607 }
2608 {
2609   \bool_if:NTF \l_@@_final_open_bool
2610   {
2611     \int_set:Nn \l_tmpa_int
2612     {
2613       \dim_ratio:nn
2614       { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
2615       \l_@@_inter_dots_dim
2616     }
2617   }
2618   {
2619     \int_set:Nn \l_tmpa_int
2620     {
2621       \dim_ratio:nn
2622       { \l_@@_l_dim - 2 \l_@@_xdots_shorten_dim }
2623       \l_@@_inter_dots_dim
2624     }
2625   }
2626 }

```

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

```

2627 \dim_set:Nn \l_tmpa_dim
2628 {
2629   ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2630   \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim

```

```

2631 }
2632 \dim_set:Nn \l_tmpb_dim
2633 {
2634   ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
2635   \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
2636 }

```

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`.

```

2637 \int_set:Nn \l_tmpb_int
2638 {
2639   \bool_if:NTF \l_@@_initial_open_bool
2640     { \bool_if:NTF \l_@@_final_open_bool 1 0 }
2641     { \bool_if:NTF \l_@@_final_open_bool 2 1 }
2642 }

```

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.

```

2643 \dim_gadd:Nn \l_@@_x_initial_dim
2644 {
2645   ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2646   \dim_ratio:nn
2647   { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
2648   { 2 \l_@@_l_dim }
2649   * \l_tmpb_int
2650 }
2651 \dim_gadd:Nn \l_@@_y_initial_dim
2652 {
2653   ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
2654   \dim_ratio:nn
2655   { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
2656   { 2 \l_@@_l_dim }
2657   * \l_tmpb_int
2658 }
2659 \pgf@relevantforpicturesizefalse
2660 \int_step_inline:nnn 0 \l_tmpa_int
2661 {
2662   \pgfpathcircle
2663     { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
2664     { \l_@@_radius_dim }
2665   \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
2666   \dim_add:Nn \l_@@_y_initial_dim \l_tmpb_dim
2667 }
2668 \pgfusepathqfill
2669 }

```

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.

```

2670 \AtBeginDocument
2671 {
2672   \tl_set:Nn \l_@@_argspec_tl { 0 { } E { _ ^ } { { } { } } }
2673   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
2674   \exp_args:NNV \NewDocumentCommand \@@_Ldots \l_@@_argspec_tl
2675     {
2676       \int_compare:nNnTF \c@jCol = 0
2677       { \@@_error:nn { in~first~col } \Ldots }
2678       {
2679         \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
2680         { \@@_error:nn { in~last~col } \Ldots }
2681         {
2682           \@@_instruction_of_type:nn { Ldots }
2683           { #1 , down = #2 , up = #3 }
2684         }
2685       }
2686       \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_ldots }
2687       \bool_gset_true:N \g_@@_empty_cell_bool
2688     }

2689   \exp_args:NNV \NewDocumentCommand \@@_Cdots \l_@@_argspec_tl
2690     {
2691       \int_compare:nNnTF \c@jCol = 0
2692       { \@@_error:nn { in~first~col } \Cdots }
2693       {
2694         \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
2695         { \@@_error:nn { in~last~col } \Cdots }
2696         {
2697           \@@_instruction_of_type:nn { Cdots }
2698           { #1 , down = #2 , up = #3 }
2699         }
2700       }
2701       \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_cdots }
2702       \bool_gset_true:N \g_@@_empty_cell_bool
2703     }

2704   \exp_args:NNV \NewDocumentCommand \@@_Vdots \l_@@_argspec_tl
2705     {
2706       \int_compare:nNnTF \c@iRow = 0
2707       { \@@_error:nn { in~first~row } \Vdots }
2708       {
2709         \int_compare:nNnTF \c@iRow = \l_@@_last_row_int
2710         { \@@_error:nn { in~last~row } \Vdots }
2711         {
2712           \@@_instruction_of_type:nn { Vdots }
2713           { #1 , down = #2 , up = #3 }
2714         }
2715       }
2716       \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_vdots }
2717       \bool_gset_true:N \g_@@_empty_cell_bool
2718     }

2719   \exp_args:NNV \NewDocumentCommand \@@_Ddots \l_@@_argspec_tl
2720     {
2721       \int_case:nnF \c@iRow
2722       {
2723         0 { \@@_error:nn { in~first~row } \Ddots }
2724         \l_@@_last_row_int { \@@_error:nn { in~last~row } \Ddots }

```

```

2725     }
2726     {
2727         \int_case:nnF \c@jCol
2728         {
2729             0 { \@@_error:nn { in~first~col } \Ddots }
2730             \l_@@_last_col_int { \@@_error:nn { in~last~col } \Ddots }
2731         }
2732         {
2733             \@@_instruction_of_type:nn { Ddots }
2734             { #1 , down = #2 , up = #3 }
2735         }
2736     }
2737     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_ddots }
2738     \bool_gset_true:N \g_@@_empty_cell_bool
2739 }
2740
2741 \exp_args:NNV \NewDocumentCommand \@@_Iddots \l_@@_argspec_tl
2742 {
2743     \int_case:nnF \c@iRow
2744     {
2745         0 { \@@_error:nn { in~first~row } \Iddots }
2746         \l_@@_last_row_int { \@@_error:nn { in~last~row } \Iddots }
2747     }
2748     {
2749         \int_case:nnF \c@jCol
2750         {
2751             0 { \@@_error:nn { in~first~col } \Iddots }
2752             \l_@@_last_col_int { \@@_error:nn { in~last~col } \Iddots }
2753         }
2754         {
2755             \@@_instruction_of_type:nn { Iddots }
2756             { #1 , down = #2 , up = #3 }
2757         }
2758     }
2759     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_iddots }
2760     \bool_gset_true:N \g_@@_empty_cell_bool
2761 }
2762 }
2763 }

```

End of the \AtBeginDocument.

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

```

2764 \cs_new_protected:Npn \@@_Hspace:
2765 {
2766     \bool_gset_true:N \g_@@_empty_cell_bool
2767     \hspace
2768 }

```

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

```

2769 \cs_set_eq:NN \@@_old_multicolumn \multicolumn
2770 \cs_new:Npn \@@_multicolumn:nnn #1 #2 #3
2771 {
2772     % \begin{macrocode}
2773     % We have to act in expandable way since it will begin by a |\multicolumn|.
2774     % \end{macrocode}
2775     \exp_args:NNe
2776     \@@_old_multicolumn
2777     { #1 }

```


We will have to replace `\tl_lower_case:n` in some times since `\tl_lower_case:n` seems to be deprecated.

```

2778 {
2779   > \@@_Cell:
2780   \bool_if:NT \c_@@_define_L_C_R_bool \tl_lower_case:n
2781   #2
2782   < \@@_end_Cell:
2783 }
2784 { #3 }

```

The `\peek_remove_spaces:n` is mandatory.

```

2785 \peek_remove_spaces:n
2786 {
2787   \int_compare:nNnT #1 > 1
2788   {
2789     \seq_gput_left:Nx \g_@@_multicolumn_cells_seq
2790     { \int_use:N \c@iRow - \int_use:N \c@jCol }
2791     \seq_gput_left:Nn \g_@@_multicolumn_sizes_seq { #1 }
2792     \seq_gput_right:Nx \g_@@_pos_of_blocks_seq
2793     {
2794       { \int_use:N \c@iRow }
2795       { \int_use:N \c@jCol }
2796       { \int_use:N \c@iRow }
2797       { \int_eval:n { \c@jCol + #1 - 1 } }
2798     }
2799   }
2800   \int_gadd:Nn \c@jCol { #1 - 1 }
2801 }
2802 }

```

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`.

```

2803 \cs_new:Npn \@@_Hdotsfor:
2804 {
2805   \multicolumn { 1 } { c } { }
2806   \@@_Hdotsfor_i
2807 }

```

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:`).

```

2808 \AtBeginDocument
2809 {
2810   \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
2811   \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.

```

2812   \exp_args:NNV \NewDocumentCommand \@@_Hdotsfor_i \l_@@_argspec_tl
2813   {
2814     \tl_gput_right:Nx \g_@@_HVdotsfor_lines_tl
2815     {
2816       \@@_Hdotsfor:nnnn
2817       { \int_use:N \c@iRow }
2818       { \int_use:N \c@jCol }
2819       { #2 }
2820       {
2821         #1 , #3 ,
2822         down = \exp_not:n { #4 } , up = \exp_not:n { #5 }
2823       }
2824     }
2825     \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { c } { } }

```

```

2826     }
2827 }

```

Enf of \AtBeginDocument.

```

2828 \cs_new_protected:Npn \@@_Hdotsfor:nnnn #1 #2 #3 #4
2829 {
2830     \bool_set_false:N \l_@@_initial_open_bool
2831     \bool_set_false:N \l_@@_final_open_bool

```

For the row, it's easy.

```

2832     \int_set:Nn \l_@@_initial_i_int { #1 }
2833     \int_set_eq:NN \l_@@_final_i_int \l_@@_initial_i_int

```

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

```

2834     \int_compare:nNnTF #2 = 1
2835     {
2836         \int_set:Nn \l_@@_initial_j_int 1
2837         \bool_set_true:N \l_@@_initial_open_bool
2838     }
2839     {
2840         \cs_if_exist:cTF
2841         {
2842             pgf @ sh @ ns @ \@@_env:
2843             - \int_use:N \l_@@_initial_i_int
2844             - \int_eval:n { #2 - 1 }
2845         }
2846         { \int_set:Nn \l_@@_initial_j_int { #2 - 1 } }
2847         {
2848             \int_set:Nn \l_@@_initial_j_int { #2 }
2849             \bool_set_true:N \l_@@_initial_open_bool
2850         }
2851     }
2852     \int_compare:nNnTF { #2 + #3 - 1 } = \c@jCol
2853     {
2854         \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
2855         \bool_set_true:N \l_@@_final_open_bool
2856     }
2857     {
2858         \cs_if_exist:cTF
2859         {
2860             pgf @ sh @ ns @ \@@_env:
2861             - \int_use:N \l_@@_final_i_int
2862             - \int_eval:n { #2 + #3 }
2863         }
2864         { \int_set:Nn \l_@@_final_j_int { #2 + #3 } }
2865         {
2866             \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
2867             \bool_set_true:N \l_@@_final_open_bool
2868         }
2869     }
2870     \group_begin:
2871     \int_compare:nNnTF { #1 } = 0
2872     { \color { nicematrix-first-row } }
2873     {
2874         \int_compare:nNnT { #1 } = \g_@@_row_total_int
2875         { \color { nicematrix-last-row } }
2876     }
2877     \keys_set:nn { NiceMatrix / xdots } { #4 }
2878     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2879     \@@_actually_draw_Ldots:
2880     \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).

```

2881 \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
2882 { \cs_set:cpn { @@ _ dotted _ #1 - ##1 } { } }
2883 }

2884 \AtBeginDocument
2885 {
2886   \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
2887   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
2888   \exp_args:NNV \NewDocumentCommand \@@_Vdotsfor: \l_@@_argspec_tl
2889   {
2890     \tl_gput_right:Nx \g_@@_HVdotsfor_lines_tl
2891     {
2892       \@@_Vdotsfor:nnnn
2893       { \int_use:N \c@iRow }
2894       { \int_use:N \c@jCol }
2895       { #2 }
2896       {
2897         #1 , #3 ,
2898         down = \exp_not:n { #4 } , up = \exp_not:n { #5 }
2899       }
2900     }
2901   }
2902 }

```

Enf of `\AtBeginDocument`.

```

2903 \cs_new_protected:Npn \@@_Vdotsfor:nnnn #1 #2 #3 #4
2904 {
2905   \bool_set_false:N \l_@@_initial_open_bool
2906   \bool_set_false:N \l_@@_final_open_bool

```

For the column, it’s easy.

```

2907 \int_set:Nn \l_@@_initial_j_int { #2 }
2908 \int_set_eq:NN \l_@@_final_j_int \l_@@_initial_j_int

```

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

```

2909 \int_compare:nNnTF #1 = 1
2910 {
2911   \int_set:Nn \l_@@_initial_i_int 1
2912   \bool_set_true:N \l_@@_initial_open_bool
2913 }
2914 {
2915   \cs_if_exist:cTF
2916   {
2917     pgf @ sh @ ns @ \@@_env:
2918     - \int_eval:n { #1 - 1 }
2919     - \int_use:N \l_@@_initial_j_int
2920   }
2921   { \int_set:Nn \l_@@_initial_i_int { #1 - 1 } }
2922   {
2923     \int_set:Nn \l_@@_initial_i_int { #1 }
2924     \bool_set_true:N \l_@@_initial_open_bool
2925   }
2926 }
2927 \int_compare:nNnTF { #1 + #3 - 1 } = \c@iRow
2928 {
2929   \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }
2930   \bool_set_true:N \l_@@_final_open_bool
2931 }
2932 {
2933   \cs_if_exist:cTF

```

```

2934     {
2935         pgf @ sh @ ns @ \@@_env:
2936         - \int_eval:n { #1 + #3 }
2937         - \int_use:N \l_@@_final_j_int
2938     }
2939     { \int_set:Nn \l_@@_final_i_int { #1 + #3 } }
2940     {
2941         \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }
2942         \bool_set_true:N \l_@@_final_open_bool
2943     }
2944 }
2945 \group_begin:
2946 \int_compare:nNnTF { #2 } = 0
2947 { \color { nicematrix-first-col } }
2948 {
2949     \int_compare:nNnT { #2 } = \g_@@_col_total_int
2950     { \color { nicematrix-last-col } }
2951 }
2952 \keys_set:nn { NiceMatrix / xdots } { #4 }
2953 \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2954 \@@_actually_draw_Vdots:
2955 \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).

```

2956     \int_step_inline:nnn { #1 } { #1 + #3 - 1 }
2957     { \cs_set:cpn { @@ _ dotted _ ##1 - #2 } { } }
2958 }

```

The command `\@@_rotate:` will be linked to `\rotate` in `{NiceArrayWithDelims}`.

The command will exit three levels of groups (only two in `{NiceTabular}` because there is not the group of the math mode to exit) in order to execute the command

“`\box_rotate:Nn \l_@@_cell_box { 90 }`”

just after the construction of the box `\l_@@_cell_box`.

```

2959 \cs_new_protected:Npn \@@_rotate:
2960 {
2961     \bool_if:NTF \l_@@_NiceTabular_bool
2962     { \group_insert_after:N \@@_rotate_ii: }
2963     { \group_insert_after:N \@@_rotate_i: }
2964 }
2965 \cs_new_protected:Npn \@@_rotate_i: { \group_insert_after:N \@@_rotate_ii: }
2966 \cs_new_protected:Npn \@@_rotate_ii: { \group_insert_after:N \@@_rotate_iii: }
2967 \cs_new_protected:Npn \@@_rotate_iii:
2968 {
2969     \box_rotate:Nn \l_@@_cell_box { 90 }

```

If we are in the last row, we want all the boxes composed with the command `\rotate` aligned upwards.

```

2970     \int_compare:nNnT \c@iRow = \l_@@_last_row_int
2971     {
2972         \vbox_set_top:Nn \l_@@_cell_box
2973         {
2974             \vbox_to_zero:n { }

```

0.8 `ex` will be the distance between the principal part of the array and our element (which is composed with `\rotate`).

```

2975         \skip_vertical:n { - \box_ht:N \@arstrutbox + 0.8 ex }
2976         \box_use:N \l_@@_cell_box
2977     }
2978 }
2979 }

```

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).⁴³

```
2980 \cs_new:Npn \@@_double_int_eval:n #1-#2 \q_stop
2981   { \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).

```
2982 \AtBeginDocument
2983 {
2984   \tl_set:Nn \l_@@_argspec_tl { 0 { } m m ! 0 { } E { _ ^ } { { } { } } }
2985   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
2986   \exp_args:NNV \NewDocumentCommand \@@_line \l_@@_argspec_tl
2987     {
2988       \group_begin:
2989       \keys_set:nn { NiceMatrix / xdots } { #1 , #4 , down = #5 , up = #6 }
2990       \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2991       \use:x
2992       {
2993         \@@_line_i:nn
2994         { \@@_double_int_eval:n #2 \q_stop }
2995         { \@@_double_int_eval:n #3 \q_stop }
2996       }
2997       \group_end:
2998     }
2999 }

3000 \cs_new_protected:Npn \@@_line_i:nn #1 #2
3001 {
3002   \bool_set_false:N \l_@@_initial_open_bool
3003   \bool_set_false:N \l_@@_final_open_bool
3004   \bool_if:nTF
3005   {
3006     \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #1 }
3007     ||
3008     \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #2 }
3009   }
3010   {
3011     \@@_error:nnn { unknown~cell~for~line~in~code~after } { #1 } { #2 }
3012   }
3013   { \@@_draw_line_ii:nn { #1 } { #2 } }
3014 }

3015 \AtBeginDocument
3016 {
3017   \cs_new_protected:Npx \@@_draw_line_ii:nn #1 #2
3018   {
```

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:.`

```
3019   \c_@@_pgfortikzpicture_tl
3020   \@@_draw_line_iii:nn { #1 } { #2 }
3021   \c_@@_endpgfortikzpicture_tl
3022 }
3023 }
```

⁴³Indeed, we want that the user may use the command `\line` in `code-after` with LaTeX counters in the arguments — with the command `\value`.

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

```

3024 \cs_new_protected:Npn \@@_draw_line_iii:nn #1 #2
3025 {
3026   \pgfrememberpicturepositiononpagetrue
3027   \pgfpointshapeborder { \@@_env: - #1 } { \@@_qpoint:n { #2 } }
3028   \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
3029   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
3030   \pgfpointshapeborder { \@@_env: - #2 } { \@@_qpoint:n { #1 } }
3031   \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
3032   \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
3033   \@@_draw_line:
3034 }

```

The commands `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, and `\Iddots` don't use this command because they have to do other settings (for example, the diagonal lines must be parallelized).

Commands available in the code-before

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

```

3035 \cs_set_protected:Npn \@@_cut_on_hyphen:w #1-#2\q_stop
3036 {
3037   \tl_set:Nn \l_tmpa_tl { #1 }
3038   \tl_set:Nn \l_tmpb_tl { #2 }
3039 }

```

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

```

3040 \NewDocumentCommand \@@_rowcolor { 0 { } m m }
3041 {
3042   \tl_if_blank:nF { #2 }
3043   {
3044     \pgfpicture
3045     \pgf@relevantforpicturesizefalse
3046     \tl_if_empty:nTF { #1 } \color { \color [ #1 ] } { #2 }

```

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

```

3047     \@@_qpoint:n { col - 1 }
3048     \int_compare:nNnTF \l_@@_first_col_int = 0
3049     { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
3050     { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
3051     \@@_qpoint:n { col - \@@_succ:n \c@jCol }
3052     \dim_set:Nn \l_tmpa_dim { \pgf@x + 0.5 \arrayrulewidth }
3053     \clist_map_inline:nn { #3 }
3054     {
3055       \tl_set:Nn \l_tmpa_tl { ##1 }
3056       \tl_if_in:NnTF \l_tmpa_tl { - }
3057       { \@@_cut_on_hyphen:w ##1 \q_stop }
3058       { \@@_cut_on_hyphen:w ##1 - ##1 \q_stop }
3059       \tl_if_empty:NT \l_tmpa_tl { \tl_set:Nn \l_tmpa_tl { 1 } }
3060       \tl_if_empty:NT \l_tmpb_tl
3061       { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@iRow } }
3062       \int_compare:nNnT \l_tmpb_tl > \c@iRow
3063       { \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`.

```

3064     \@@_qpoint:n { row - \@@_succ:n \l_tmpb_tl }
3065     \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
3066     \@@_qpoint:n { row - \l_tmpa_tl }
3067     \dim_set:Nn \l_tmpd_dim { \pgf@y + 0.5 \arrayrulewidth }
3068     \pgfpathrectanglecorners
3069     { \pgfpoint \l_tmpc_dim \l_tmpd_dim }

```

```

3070         { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3071     }
3072     \pgfusepathqfill
3073     \endpgfpicture
3074 }
3075 }

```

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

```

3076 \NewDocumentCommand \@@_columncolor { 0 { } m m }
3077 {
3078     \tl_if_blank:nF { #2 }
3079     {
3080         \pgfpicture
3081         \pgf@relevantforpicturesizefalse
3082         \tl_if_empty:nTF { #1 } \color { \color [ #1 ] } { #2 }
3083         \@@_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.

```

3084         \dim_set:Nn \l_tmpa_dim { \pgf@y + 0.5 \arrayrulewidth }
3085         \@@_qpoint:n { row - \@@_succ:n \c@iRow }
3086         \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
3087         \clist_map_inline:nn { #3 }
3088         {
3089             \tl_set:Nn \l_tmpa_tl { ##1 }
3090             \tl_if_in:NnTF \l_tmpa_tl { - }
3091             { \@@_cut_on_hyphen:w ##1 \q_stop }
3092             { \@@_cut_on_hyphen:w ##1 - ##1 \q_stop }
3093             \tl_if_empty:NT \l_tmpa_tl { \tl_set:Nn \l_tmpa_tl { 1 } }
3094             \tl_if_empty:NT \l_tmpb_tl
3095             { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
3096             \int_compare:nNnT \l_tmpb_tl > \c@jCol
3097             { \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`.

```

3098         \@@_qpoint:n { col - \l_tmpa_tl }
3099         \int_compare:nNnTF \l_@@_first_col_int = \l_tmpa_tl
3100         { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
3101         { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
3102         \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
3103         \dim_set:Nn \l_tmpd_dim { \pgf@x + 0.5 \arrayrulewidth }
3104         \pgfpathrectanglecorners
3105         { \pgfpoint \l_tmpc_dim \l_tmpa_dim }
3106         { \pgfpoint \l_tmpd_dim \l_tmpb_dim }
3107     }
3108     \pgfusepathqfill
3109     \endpgfpicture
3110 }
3111 }

```

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

```

3112 \NewDocumentCommand \@@_cellcolor { 0 { } m m }
3113 {
3114     \tl_if_blank:nF { #2 }
3115     {
3116         \pgfpicture
3117         \pgf@relevantforpicturesizefalse
3118         \tl_if_empty:nTF { #1 } \color { \color [ #1 ] } { #2 }
3119         \clist_map_inline:nn { #3 }
3120         {
3121             \@@_cut_on_hyphen:w ##1 \q_stop
3122             \@@_qpoint:n { row - \l_tmpa_tl }
3123             \bool_lazy_and:nnT
3124             { \int_compare_p:n { \l_tmpa_tl <= \c@iRow } }

```

```

3125 { \int_compare_p:n { \l_tmpb_tl <= \c@jCol } }
3126 {
3127   \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
3128   \@@_qpoint:n { row - \@@_succ:n \l_tmpa_tl }
3129   \dim_set:Nn \l_tmpa_dim { \pgf@y + 0.5 \arrayrulewidth }
3130   \@@_qpoint:n { col - \l_tmpb_tl }
3131   \int_compare:nNnTF \l_@@_first_col_int = \l_tmpb_tl
3132     { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
3133     { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
3134   \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
3135   \dim_set:Nn \l_tmpd_dim { \pgf@x + 0.5 \arrayrulewidth }
3136   \pgfpathrectanglecorners
3137     { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
3138     { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
3139   }
3140 }
3141 \pgfusepathqfill
3142 \endpgfpicture
3143 }
3144 }

```

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

```

3145 \NewDocumentCommand \@@_rectanglecolor { 0 { } m m m }
3146 {
3147   \tl_if_blank:nF { #2 }
3148   {
3149     \pgfpicture
3150     \pgf@relevantforpicturesizefalse
3151     \tl_if_empty:nTF { #1 } { \color { \color [ #1 ] } { #2 } }
3152     \@@_cut_on_hyphen:w #3 \q_stop
3153     \bool_lazy_and:nnT
3154       { \int_compare_p:n { \l_tmpa_tl <= \c@iRow } }
3155       { \int_compare_p:n { \l_tmpb_tl <= \c@jCol } }
3156     {
3157       \@@_qpoint:n { row - \l_tmpa_tl }
3158       \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
3159       \@@_qpoint:n { col - \l_tmpb_tl }
3160       \int_compare:nNnTF \l_@@_first_col_int = \l_tmpb_tl
3161         { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
3162         { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
3163       \@@_cut_on_hyphen:w #4 \q_stop
3164       \int_compare:nNnT \l_tmpa_tl > \c@iRow
3165         { \tl_set:Nx \l_tmpa_tl { \int_use:N \c@iRow } }
3166       \int_compare:nNnT \l_tmpb_tl > \c@jCol
3167         { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
3168       \@@_qpoint:n { row - \@@_succ:n \l_tmpa_tl }
3169       \dim_set:Nn \l_tmpa_dim { \pgf@y + 0.5 \arrayrulewidth }
3170       \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
3171       \dim_set:Nn \l_tmpd_dim { \pgf@x + 0.5 \arrayrulewidth }
3172       \pgfpathrectanglecorners
3173         { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
3174         { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
3175       \pgfusepathqfill
3176     }
3177   \endpgfpicture
3178 }
3179 }

```

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`.

```

3180 \NewDocumentCommand \@@_rowcolors { 0 { } m m m }

```



```

3181 {
3182   \int_step_inline:nnn { #2 } { \int_use:N \c@iRow }
3183   {
3184     \int_if_odd:nTF { ##1 }
3185       { \@@_rowcolor [ #1 ] { #3 } }
3186       { \@@_rowcolor [ #1 ] { #4 } }
3187     { ##1 }
3188   }
3189 }

3190 \NewDocumentCommand \@@_chessboardcolors { 0 { } m m }
3191 {
3192   \int_step_inline:nn { \int_use:N \c@iRow }
3193   {
3194     \int_step_inline:nn { \int_use:N \c@jCol }
3195     {
3196       \int_if_even:nTF { ####1 + ##1 }
3197         { \@@_cellcolor [ #1 ] { #2 } }
3198         { \@@_cellcolor [ #1 ] { #3 } }
3199       { ##1 - ####1 }
3200     }
3201   }
3202 }

```

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`.

```

3203 \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`.

```

3204 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix:n #1
3205 {
3206   \int_compare:nNnTF \l_@@_first_col_int = 0
3207     { \@@_OnlyMainNiceMatrix_i:n { #1 } }
3208     {
3209       \int_compare:nNnTF \c@jCol = 0
3210       {
3211         \int_compare:nNnF \c@iRow = { -1 }
3212         { \int_compare:nNnF \c@iRow = { \l_@@_last_row_int - 1 } { #1 } }
3213       }
3214       { \@@_OnlyMainNiceMatrix_i:n { #1 } }
3215     }
3216 }

```

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.

```

3217 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix_i:n #1
3218 {
3219   \int_compare:nNnF \c@iRow = 0

```

```

3220     { \int_compare:nNnF \c@iRow = \l_@@_last_row_int { #1 } }
3221 }

```

Remember that `\c@iRow` is not always inferior to `\l_@@_last_row_int` because `\l_@@_last_row_int` may be equal to -2 or -1 (we can't write `\int_compare:nNnT \c@iRow < \l_@@_last_row_int`).

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

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

```

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

```

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

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

Here is the definition of `\@@_vline`:. This definition *must* be protected because you don't want that macro expanded during the construction of the preamble (the tests in `\@@_OnlyMainNiceMatrix:n` must be effective in each row and not once for all when the preamble is constructed). The command `\CT@arc@` is a command of `colortbl` which sets the color of the rules in the array. The package `nicematrix` uses it even if `colortbl` is not loaded.

```

3222 \cs_new_protected:Npn \@@_vline:
3223 { \@@_OnlyMainNiceMatrix:n { { \CT@arc@ \vline } } }

```

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

```

3224 \cs_new_protected:Npn \@@_draw_vlines:
3225 {
3226   \group_begin:

```

The command `\CT@arc@` is a command of `colortbl` which sets the color of the rules in the array. The package `nicematrix` uses it even when `colortbl` is not loaded.

```

3227   \CT@arc@
3228   \pgfpicture
3229   \pgfrememberpicturerepositiononpagetrue
3230   \pgf@relevantforpicturesizefalse
3231   \pgfsetlinewidth \arrayrulewidth
3232   \pgfsetrectcap
3233   \@@_qpoint:n { row - 1 }
3234   \dim_set_eq:NN \l_tmpa_dim \pgf@y
3235   \@@_qpoint:n { row - \@@_succ:n \c@iRow }
3236   \dim_set_eq:NN \l_tmpb_dim \pgf@y

```

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

```

3237   \int_step_inline:nnn
3238     { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
3239     { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@jCol } \c@jCol }
3240     {
3241       \@@_qpoint:n { col - ##1 }
3242       \dim_set_eq:NN \l_tmpc_dim \pgf@x
3243       \pgfpathmoveto { \pgfpoint \l_tmpc_dim \l_tmpa_dim }
3244       \pgfpathlineto { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
3245     }
3246   \pgfusepathqstroke
3247   \endpgfpicture
3248   \group_end:
3249 }

```

The key hvlines

The key hvlines

```
3250 \cs_new_protected:Npn \@@_draw_hvlines:
3251 {
3252   \pgfpicture
3253   \CT@arc@
3254   \pgfrememberpicturepositiononpagetrue
3255   \pgf@relevantforpicturesizefalse
3256   \pgfsetlinewidth \arrayrulewidth
3257   \int_step_inline:nnn
3258     { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
3259     { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@iRow } \c@iRow }
3260     {
3261       \@@_qpoint:n { row - ##1 }
3262       \dim_set_eq:NN \l_tmpa_dim \pgf@y
3263       \pgfpathmoveto { \pgfpoint \pgf@x \pgf@y }
3264       \@@_qpoint:n { col - \@@_succ:n { \c@jCol } }
3265       \dim_set:Nn \l_tmpb_dim { \pgf@x + 0.5 \arrayrulewidth }
3266       \pgfpathlineto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
3267     }
3268   \pgfusepathqstroke
3269   \endpgfpicture
3270 }
```

Since version 4.1, the key `hvlines` is no longer a mere alias for the conjunction of `hlines` and `vlines`. Indeed, with `hvlines`, the vertical and horizontal rules are *not* drawn within the blocks (created by `\Block`) nor within the “virtual blocks” (corresponding to the dotted lines drawn by `\Cdots`, `\Vdots`, etc.).

```
3271 \cs_new_protected:Npn \@@_draw_hvlines:
3272 {
3273   \bool_lazy_and:nnTF
3274     { \seq_if_empty_p:N \g_@@_pos_of_blocks_seq }
3275     { \seq_if_empty_p:N \g_@@_pos_of_xdots_seq }
3276     \@@_draw_hvlines_i:
3277     \@@_draw_hvlines_ii:
3278 }
```

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

```
3279 \cs_new_protected:Npn \@@_draw_hvlines_i:
3280 {
3281   \@@_draw_hlines:
3282   \@@_draw_vlines:
3283 }
```

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

```
3284 \cs_new_protected:Npn \@@_draw_hvlines_ii:
3285 {
3286   \group_begin:
3287   \CT@arc@
3288   \int_step_variable:nnNn
3289     { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
3290     { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@iRow } \c@iRow }
3291     \l_tmpa_tl
3292     {
3293       \int_step_variable:nnNn \c@jCol \l_tmpb_tl
3294       {
```

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.

```

3295         \bool_gset_true:N \g_tmpa_bool
3296         \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
3297         { \@@_test_if_hline_in_block:nnnn ##1 }
3298         \seq_map_inline:Nn \g_@@_pos_of_xdots_seq
3299         { \@@_test_if_hline_in_block:nnnn ##1 }
3300         \bool_if:NT \l_@@_hvlines_except_corners_bool
3301         {
3302             \int_compare:nNnTF \l_tmpa_tl = { \c@iRow + 1 }
3303             {
3304                 \seq_if_in:NxT
3305                 \l_@@_empty_corner_cells_seq
3306                 { \@@_pred:n \l_tmpa_tl - \l_tmpb_tl }
3307                 { \bool_set_false:N \g_tmpa_bool }
3308             }
3309             {
3310                 \seq_if_in:NxT
3311                 \l_@@_empty_corner_cells_seq
3312                 { \l_tmpa_tl - \l_tmpb_tl }
3313                 {
3314                     \int_compare:nNnTF \l_tmpa_tl = 1
3315                     { \bool_set_false:N \g_tmpa_bool }
3316                     {
3317                         \seq_if_in:NxT
3318                         \l_@@_empty_corner_cells_seq
3319                         { \@@_pred:n \l_tmpa_tl - \l_tmpb_tl }
3320                         { \bool_set_false:N \g_tmpa_bool }
3321                     }
3322                 }
3323             }
3324         }
3325         \bool_if:NT \g_tmpa_bool
3326         {
3327             \pgfpicture
3328             \pgfrememberpicturepositiononpagetrue
3329             \pgf@relevantforpicturesizefalse
3330             \pgfsetlinewidth \arrayrulewidth
3331             \pgfsetrectcap
3332             \@@_qpoint:n { row - \l_tmpa_tl }
3333             \dim_set_eq:NN \l_tmpb_dim \pgf@y
3334             \@@_qpoint:n { col - \l_tmpb_tl }
3335             \dim_set_eq:NN \l_tmpa_dim \pgf@x
3336             \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
3337             \dim_set_eq:NN \l_tmpc_dim \pgf@x
3338             \pgfpathmoveto { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3339             \pgfpathlineto { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
3340             \pgfusepathqstroke
3341             \endpgfpicture
3342         }
3343     }
3344 }

```

Now, the vertical rules.

```

3345     \int_step_variable:nNn \c@iRow \l_tmpa_tl
3346     {
3347         \int_step_variable:nnNn
3348         { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
3349         { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@jCol } \c@jCol }
3350         \l_tmpb_tl
3351         {

```

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.

```

3352         \bool_gset_true:N \g_tmpa_bool
3353         \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
3354         { \@@_test_if_vline_in_block:nnnn ##1 }
3355         \seq_map_inline:Nn \g_@@_pos_of_xdots_seq
3356         { \@@_test_if_vline_in_block:nnnn ##1 }
3357         \bool_if:NT \l_@@_hvlines_except_corners_bool
3358         {
3359             \int_compare:nNnTF \l_tmpb_tl = { \@@_succ:n \c@jCol }
3360             {
3361                 \seq_if_in:NxT
3362                 \l_@@_empty_corner_cells_seq
3363                 { \l_tmpa_tl - \@@_pred:n \l_tmpb_tl }
3364                 { \bool_set_false:N \g_tmpa_bool }
3365             }
3366             {
3367                 \seq_if_in:NxT
3368                 \l_@@_empty_corner_cells_seq
3369                 { \l_tmpa_tl - \l_tmpb_tl }
3370                 {
3371                     \int_compare:nNnTF \l_tmpb_tl = 1
3372                     { \bool_set_false:N \g_tmpa_bool }
3373                     {
3374                         \seq_if_in:NxT
3375                         \l_@@_empty_corner_cells_seq
3376                         { \l_tmpa_tl - \@@_pred:n \l_tmpb_tl }
3377                         { \bool_set_false:N \g_tmpa_bool }
3378                     }
3379                 }
3380             }
3381         }
3382         \bool_if:NT \g_tmpa_bool
3383         {
3384             \pgfpicture
3385             \pgfrememberpicturepositiononpagetrue
3386             \pgf@relevantforpicturesizefalse
3387             \pgfsetlinewidth \arrayrulewidth
3388             \pgfsetrectcap
3389             \@@_qpoint:n { row - \l_tmpa_tl }
3390             \dim_set_eq:NN \l_tmpb_dim \pgf@y
3391             \@@_qpoint:n { col - \l_tmpb_tl }
3392             \dim_set_eq:NN \l_tmpa_dim \pgf@x
3393             \@@_qpoint:n { row - \@@_succ:n \l_tmpa_tl }
3394             \dim_set_eq:NN \l_tmpc_dim \pgf@y
3395             \pgfpathmoveto { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3396             \pgfpathlineto { \pgfpoint \l_tmpa_dim \l_tmpc_dim }
3397             \pgfusepathqstroke
3398             \endpgfpicture
3399         }
3400     }
3401 }

```

The group was for the color of the rules.

```

3402     \group_end:
3403     \seq_gclear:N \g_@@_pos_of_xdots_seq
3404 }

```

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.

```

3405 \cs_new_protected:Npn \@@_test_if_hline_in_block:nnnn #1 #2 #3 #4
3406 {

```

```

3407 \bool_lazy_all:nT
3408 {
3409   { \int_compare_p:nNn \l_tmpa_tl > { #1 } }
3410   { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
3411   { \int_compare_p:nNn \l_tmpb_tl > { #2 - 1 } }
3412   { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
3413 }
3414 { \bool_gset_false:N \g_tmpa_bool }
3415 }

```

The same for vertical rules.

```

3416 \cs_new_protected:Npn \@@_test_if_vline_in_block:nnnn #1 #2 #3 #4
3417 {
3418   \bool_lazy_all:nT
3419   {
3420     { \int_compare_p:nNn \l_tmpa_tl > { #1 - 1 } }
3421     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
3422     { \int_compare_p:nNn \l_tmpb_tl > { #2 } }
3423     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
3424   }
3425   { \bool_gset_false:N \g_tmpa_bool }
3426 }

```

The key hvlines-except-corners

```

3427 \cs_new_protected:Npn \@@_draw_hvlines_except_corners:
3428 {

```

The sequence `\l_@@_empty_corner_cells_seq` will be the sequence of all the cells empty (and not in a block) considered in the corners of the array.

```

3429 \seq_clear_new:N \l_@@_empty_corner_cells_seq
3430 \@@_compute_a_corner:nnnnnn 1 1 1 1 \c@iRow \c@jCol
3431 \@@_compute_a_corner:nnnnnn 1 \c@jCol 1 { -1 } \c@iRow 1
3432 \@@_compute_a_corner:nnnnnn \c@iRow 1 { -1 } 1 1 \c@jCol
3433 \@@_compute_a_corner:nnnnnn \c@iRow \c@jCol { -1 } { -1 } 1 1
3434 \@@_draw_hvlines_ii:
3435 }

```

“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 step 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.

```

3436 \cs_new_protected:Npn \@@_compute_a_corner:nnnnnn #1 #2 #3 #4 #5 #6
3437 {

```

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.

```

3438 \bool_set_false:N \l_tmpa_bool
3439 \int_zero_new:N \l_@@_last_empty_row_int
3440 \int_step_inline:nnnn { #1 } { #3 } { #5 }
3441 {
3442   \@@_test_if_cell_in_a_block:nn { ##1 } { \int_eval:n { #2 } }
3443   \bool_if:nTF
3444   {

```

```

3445         \cs_if_exist_p:c
3446         { pgf @ sh @ ns @ \@@_env: - ##1 - \int_eval:n { #2 } }
3447         ||
3448         \l_tmpb_bool
3449     }
3450     { \bool_set_true:N \l_tmpa_bool }
3451     {
3452         \bool_if:NF \l_tmpa_bool
3453         { \int_set:Nn \l_@@_last_empty_row_int { ##1 } }
3454     }
3455 }

```

Now, you determine the last empty cell in the row of number 1.

```

3456     \bool_set_false:N \l_tmpa_bool
3457     \int_zero_new:N \l_@@_last_empty_column_int
3458     \int_step_inline:nnnn { #2 } { #4 } { #6 }
3459     {
3460         \@@_test_if_cell_in_a_block:nn { \int_eval:n { #1 } } { ##1 }
3461         \bool_if:nTF
3462         {
3463             \cs_if_exist_p:c
3464             { pgf @ sh @ ns @ \@@_env: - \int_eval:n { #1 } - ##1 }
3465             || \l_tmpb_bool
3466         }
3467         { \bool_set_true:N \l_tmpa_bool }
3468         {
3469             \bool_if:NF \l_tmpa_bool
3470             { \int_set:Nn \l_@@_last_empty_column_int { ##1 } }
3471         }
3472     }

```

Now, we loop over the rows.

```

3473     \int_step_inline:nnnn { #1 } { #3 } \l_@@_last_empty_row_int
3474     {

```

We treat the row number ##1 with another loop.

```

3475         \bool_set_false:N \l_tmpa_bool
3476         \int_step_inline:nnnn { #2 } { #4 } \l_@@_last_empty_column_int
3477         {
3478             \@@_test_if_cell_in_a_block:nn { ##1 } { #####1 }
3479             \bool_if:nTF
3480             {
3481                 \cs_if_exist_p:c
3482                 { pgf @ sh @ ns @ \@@_env: - ##1 - #####1 }
3483                 || \l_tmpb_bool
3484             }
3485             { \bool_set_true:N \l_tmpa_bool }
3486             {
3487                 \bool_if:NF \l_tmpa_bool
3488                 {
3489                     \int_set:Nn \l_@@_last_empty_column_int { #####1 }
3490                     \seq_put_right:Nn
3491                     \l_@@_empty_corner_cells_seq
3492                     { ##1 - #####1 }
3493                 }
3494             }
3495         }
3496     }
3497 }

```

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`).

```

3498 \cs_new_protected:Npn \@@_test_if_cell_in_a_block:nn #1 #2

```

```

3499 {
3500   \int_set:Nn \l_tmpa_int { #1 }
3501   \int_set:Nn \l_tmpb_int { #2 }
3502   \bool_set_false:N \l_tmpb_bool
3503   \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
3504     { \@@_test_if_cell_in_block:nnnnnnn { \l_tmpa_int } { \l_tmpb_int } ##1 }
3505 }
3506 \cs_new_protected:Npn \@@_test_if_cell_in_block:nnnnnnn #1 #2 #3 #4 #5 #6
3507 {
3508   \bool_lazy_all:nT
3509     {
3510       { \int_compare_p:n { #3 <= #1 } }
3511       { \int_compare_p:n { #1 <= #5 } }
3512       { \int_compare_p:n { #4 <= #2 } }
3513       { \int_compare_p:n { #2 <= #6 } }
3514     }
3515   { \bool_set_true:N \l_tmpb_bool }
3516 }

```

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`.

```

3517 \cs_new:Npn \@@_hdottedline:
3518 {
3519   \noalign { \skip_vertical:N 2\l_@@_radius_dim }
3520   \@@_hdottedline_i:
3521 }

```

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

```

3522 \cs_new_protected:Npn \@@_hdottedline_i:
3523 {

```

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

```

3524   \tl_gput_right:Nx \g_@@_internal_code_after_tl
3525   { \@@_hdottedline:n { \int_use:N \c@iRow } }
3526 }

```

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.

```

3527 \AtBeginDocument
3528 {

```

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

```

3529   \cs_new_protected:Npx \@@_hdottedline:n #1
3530   {
3531     \bool_set_true:N \exp_not:N \l_@@_initial_open_bool
3532     \bool_set_true:N \exp_not:N \l_@@_final_open_bool
3533     \c_@@_pgfortikzpicture_tl
3534     \@@_hdottedline_i:n { #1 }
3535     \c_@@_endpgfortikzpicture_tl
3536   }
3537 }

```


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

```

3538 \cs_new_protected:Npn \@@_hdottedline_i:n #1
3539 {
3540   \pgfrememberpicturepositiononpagetrue
3541   \@@_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).

```

3542   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
3543   \dim_sub:Nn \l_@@_y_initial_dim \l_@@_radius_dim
3544   \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim

```

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

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

```

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

```

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

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

```

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

```

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

```

3545   \@@_qpoint:n { col - 1 }
3546   \dim_set:Nn \l_@@_x_initial_dim
3547   {
3548     \pgf@x +
3549     \bool_if:NTF \l_@@_NiceTabular_bool \tabcolsep \arraycolsep
3550     - \l_@@_left_margin_dim
3551   }
3552   \@@_qpoint:n { col - \@@_succ:n \c@jCol }
3553   \dim_set:Nn \l_@@_x_final_dim
3554   {
3555     \pgf@x -
3556     \bool_if:NTF \l_@@_NiceTabular_bool \tabcolsep \arraycolsep
3557     + \l_@@_right_margin_dim
3558   }

```

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.

```

3559   \tl_set:Nn \l_tmpa_tl { ( }
3560   \tl_if_eq:NnF \l_@@_left_delim_tl \l_tmpa_tl
3561     { \dim_gadd:Nn \l_@@_x_initial_dim { 0.5 \l_@@_inter_dots_dim } }
3562   \tl_set:Nn \l_tmpa_tl { ) }
3563   \tl_if_eq:NnF \l_@@_right_delim_tl \l_tmpa_tl
3564     { \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`.

```

3565   \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
3566   \@@_draw_line:
3567 }

```

Vertical dotted lines

```

3568 \cs_new_protected:Npn \@@_vdottedline:n #1
3569 {
3570     \bool_set_true:N \l_@@_initial_open_bool
3571     \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”.

```

3572     \bool_if:NTF \c_@@_tikz_loaded_bool
3573     {
3574         \tikzpicture
3575         \@@_vdottedline_i:n { #1 }
3576         \endtikzpicture
3577     }
3578     {
3579         \pgfpicture
3580         \@@_vdottedline_i:n { #1 }
3581         \endpgfpicture
3582     }
3583 }

```

```

3584 \cs_new_protected:Npn \@@_vdottedline_i:n #1
3585 {

```

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.

```

3586     \CT@arc@
3587     \pgfrememberpicturepositiononpagetrue
3588     \@@_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).

```

3589     \dim_set:Nn \l_@@_x_initial_dim { \pgf@x - \l_@@_radius_dim }
3590     \dim_set:Nn \l_@@_x_final_dim { \pgf@x - \l_@@_radius_dim }
3591     \@@_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.

```

3592     \dim_set:Nn \l_@@_y_initial_dim { \pgf@y - 0.5 \l_@@_inter_dots_dim }
3593     \@@_qpoint:n { row - \@@_succ:n \c@iRow }
3594     \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`.

```

3595     \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
3596     \@@_draw_line:
3597 }

```

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.

```

3598 \bool_new:N \l_@@_block_auto_columns_width_bool

```

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

```

3599 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
3600 {
3601     auto-columns-width .code:n =
3602     {

```

```

3603     \bool_set_true:N \l_@@_block_auto_columns_width_bool
3604     \dim_gzero_new:N \g_@@_max_cell_width_dim
3605     \bool_set_true:N \l_@@_auto_columns_width_bool
3606   }
3607 }

3608 \NewDocumentEnvironment { NiceMatrixBlock } { ! 0 { } }
3609 {
3610   \int_gincr:N \g_@@_NiceMatrixBlock_int
3611   \dim_zero:N \l_@@_columns_width_dim
3612   \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
3613   \bool_if:NT \l_@@_block_auto_columns_width_bool
3614   {
3615     \cs_if_exist:cT { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
3616     {
3617       \exp_args:Nnc \dim_set:Nn \l_@@_columns_width_dim
3618       { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
3619     }
3620   }
3621 }

```

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).

```

3622 {
3623   \bool_if:NT \l_@@_block_auto_columns_width_bool
3624   {
3625     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
3626     \iow_shipout:Nx \@mainaux
3627     {
3628       \cs_gset:cpn
3629       { @@_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.

```

3630       { \dim_eval:n { \g_@@_max_cell_width_dim + \arrayrulewidth } }
3631     }
3632     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
3633   }
3634 }

```

The extra nodes

First, two variants of the functions \dim_min:nn and \dim_max:nn.

```

3635 \cs_generate_variant:Nn \dim_min:nn { v n }
3636 \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`.

```

3637 \cs_new_protected:Npn \@@_computations_for_medium_nodes:
3638 {
3639   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
3640   {
3641     \dim_zero_new:c { l_@@_row_\@@_i: _min_dim }
3642     \dim_set_eq:cN { l_@@_row_\@@_i: _min_dim } \c_max_dim
3643     \dim_zero_new:c { l_@@_row_\@@_i: _max_dim }
3644     \dim_set:cn { l_@@_row_\@@_i: _max_dim } { - \c_max_dim }
3645   }
3646   \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
3647   {
3648     \dim_zero_new:c { l_@@_column_\@@_j: _min_dim }
3649     \dim_set_eq:cN { l_@@_column_\@@_j: _min_dim } \c_max_dim
3650     \dim_zero_new:c { l_@@_column_\@@_j: _max_dim }
3651     \dim_set:cn { l_@@_column_\@@_j: _max_dim } { - \c_max_dim }
3652   }

```

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

```

3653   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
3654   {
3655     \int_step_variable:nnNn
3656       \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.

```

3657     {
3658       \cs_if_exist:cT
3659       { 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`.

```

3660     {
3661       \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { south-west }
3662       \dim_set:cn { l_@@_row_\@@_i: _min_dim }
3663       { \dim_min:vn { l_@@_row _ \@@_i: _min_dim } \pgf@y }
3664       \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
3665       {
3666         \dim_set:cn { l_@@_column _ \@@_j: _min_dim }
3667         { \dim_min:vn { l_@@_column _ \@@_j: _min_dim } \pgf@x }
3668       }

```

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`.

```

3669       \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { north-east }
3670       \dim_set:cn { l_@@_row _ \@@_i: _ max_dim }
3671       { \dim_max:vn { l_@@_row _ \@@_i: _ max_dim } \pgf@y }
3672       \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
3673       {
3674         \dim_set:cn { l_@@_column _ \@@_j: _ max_dim }
3675         { \dim_max:vn { l_@@_column _ \@@_j: _ max_dim } \pgf@x }
3676       }
3677     }
3678   }
3679 }

```

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

```

3680   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
3681   {
3682     \dim_compare:nNnT
3683     { \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } } = \c_max_dim

```

```

3684     {
3685         \@@_qpoint:n { row - \@@_i: - base }
3686         \dim_set:cn { l_@@_row _ \@@_i: _ max _ dim } \pgf@y
3687         \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim } \pgf@y
3688     }
3689 }
3690 \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
3691 {
3692     \dim_compare:nNnT
3693     { \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } } = \c_max_dim
3694     {
3695         \@@_qpoint:n { col - \@@_j: }
3696         \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim } \pgf@y
3697         \dim_set:cn { l_@@_column _ \@@_j: _ min _ dim } \pgf@y
3698     }
3699 }
3700 }

```

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

```

3701 \cs_new_protected:Npn \@@_create_medium_nodes:
3702 {
3703     \pgfpicture
3704     \pgfrememberpicturepositiononpagetrue
3705     \pgf@relevantforpicturesizefalse
3706     \@@_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”.

```

3707     \tl_set:Nn \l_@@_suffix_tl { -medium }
3708     \@@_create_nodes:
3709     \endpgfpicture
3710 }

```

The command `\@@_create_large_nodes:` must be used when we want to create only the “large nodes” and not the medium ones⁴⁴. 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:`.

```

3711 \cs_new_protected:Npn \@@_create_large_nodes:
3712 {
3713     \pgfpicture
3714     \pgfrememberpicturepositiononpagetrue
3715     \pgf@relevantforpicturesizefalse
3716     \@@_computations_for_medium_nodes:
3717     \@@_computations_for_large_nodes:
3718     \tl_set:Nn \l_@@_suffix_tl { - large }
3719     \@@_create_nodes:
3720     \endpgfpicture
3721 }
3722 \cs_new_protected:Npn \@@_create_medium_and_large_nodes:
3723 {
3724     \pgfpicture
3725     \pgfrememberpicturepositiononpagetrue
3726     \pgf@relevantforpicturesizefalse
3727     \@@_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”.

```

3728     \tl_set:Nn \l_@@_suffix_tl { - medium }

```

⁴⁴If we want to create both, we have to use `\@@_create_medium_and_large_nodes:`

```

3729 \@@_create_nodes:
3730 \@@_computations_for_large_nodes:
3731 \tl_set:Nn \l_@@_suffix_tl { - large }
3732 \@@_create_nodes:
3733 \endpgfpicture
3734 }

```

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.

```

3735 \cs_new_protected:Npn \@@_computations_for_large_nodes:
3736 {
3737   \int_set:Nn \l_@@_first_row_int 1
3738   \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`.

```

3739   \int_step_variable:nNn { \c@iRow - 1 } \@@_i:
3740   {
3741     \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim }
3742     {
3743       (
3744         \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } +
3745         \dim_use:c { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
3746       )
3747       / 2
3748     }
3749     \dim_set_eq:cc { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
3750     { l_@@_row _ \@@_i: _ min _ dim }
3751   }
3752   \int_step_variable:nNn { \c@jCol - 1 } \@@_j:
3753   {
3754     \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim }
3755     {
3756       (
3757         \dim_use:c { l_@@_column _ \@@_j: _ max _ dim } +
3758         \dim_use:c
3759         { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
3760       )
3761       / 2
3762     }
3763     \dim_set_eq:cc { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
3764     { l_@@_column _ \@@_j: _ max _ dim }
3765   }

```

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

```

3766   \dim_sub:cn
3767   { l_@@_column _ 1 _ min _ dim }
3768   \l_@@_left_margin_dim
3769   \dim_add:cn
3770   { l_@@_column _ \int_use:N \c@jCol _ max _ dim }
3771   \l_@@_right_margin_dim
3772 }

```

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).

```

3773 \cs_new_protected:Npn \@@_create_nodes:
3774 {
3775   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:

```

```

3776 {
3777   \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
3778   {

```

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

```

3779     \@@_pgf_rect_node:nnnnn
3780     { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
3781     { \dim_use:c { l_@@_column_ \@@_j: _min_dim } }
3782     { \dim_use:c { l_@@_row_ \@@_i: _min_dim } }
3783     { \dim_use:c { l_@@_column_ \@@_j: _max_dim } }
3784     { \dim_use:c { l_@@_row_ \@@_i: _max_dim } }
3785     \str_if_empty:NF \l_@@_name_str
3786     {
3787       \pgfnodealias
3788       { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
3789       { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
3790     }
3791   }
3792 }

```

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 .

```

3793   \seq_mapthread_function:NNN
3794   \g_@@_multicolumn_cells_seq
3795   \g_@@_multicolumn_sizes_seq
3796   \@@_node_for_multicolumn:nn
3797 }

```

```

3798 \cs_new_protected:Npn \@@_extract_coords_values: #1 - #2 \q_stop
3799 {
3800   \cs_set:Npn \@@_i: { #1 }
3801   \cs_set:Npn \@@_j: { #2 }
3802 }

```

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”).

```

3803 \cs_new_protected:Npn \@@_node_for_multicolumn:nn #1 #2
3804 {
3805   \@@_extract_coords_values: #1 \q_stop
3806   \@@_pgf_rect_node:nnnnn
3807   { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
3808   { \dim_use:c { l_@@_column_ \@@_j: _min_dim } }
3809   { \dim_use:c { l_@@_row_ \@@_i: _min_dim } }
3810   { \dim_use:c { l_@@_column_ \int_eval:n { \@@_j: +#2-1 } _max_dim } }
3811   { \dim_use:c { l_@@_row_ \@@_i: _max_dim } }
3812   \str_if_empty:NF \l_@@_name_str
3813   {
3814     \pgfnodealias
3815     { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
3816     { \int_use:N \g_@@_env_int - \@@_i: - \@@_j: \l_@@_suffix_tl }
3817   }
3818 }

```

The blocks

The code deals with the command `\Block`. This command has no direct link with the environment `{NiceMatrixBlock}`.

The following command will be linked to `\Block` in the environments of `nicematrix`. We define it with `\NewDocumentCommand` of `xparse` because it has an optional argument between `<` and `>` (for TeX instructions put before the math mode of the label)

It's mandatory to use an expandable command (probably because of the first optional argument?).

```
3819 \NewExpandableDocumentCommand \@@_Block: { 0 { } m D < > { } m }
3820 { \@@_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.

```
3821 \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.

```
3822 \cs_new_protected:Npn \@@_Block_ii:nnnnn #1 #2 #3 #4 #5
3823 {
3824   \bool_if:NT \l_@@_NiceTabular_bool
3825   { \tl_if_empty:nF { #4 } { \@@_error:n { angle-option~in~NiceTabular } } }

3826   \tl_set:Nx \l_tmpa_tl
3827   {
3828     { \int_use:N \c@iRow }
3829     { \int_use:N \c@jCol }
3830     { \int_eval:n { \c@iRow + #1 - 1 } }
3831     { \int_eval:n { \c@jCol + #2 - 1 } }
3832   }
```

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

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

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

```
3833 \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}`.

```
3834 \seq_gput_left:Nx \g_@@_blocks_seq
3835 {
3836   \l_tmpa_tl
3837   { #3 }
3838   \exp_not:n { { #4 \@@_math_toggle_token: #5 \@@_math_toggle_token: } }
3839 }
3840 }
```

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?

```
3841 \keys_define:nn { NiceMatrix / Block }
3842 {
3843   tikz .tl_set:N = \l_@@_tikz_tl ,
3844   tikz .value_required:n = true ,
3845 }
```

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

```
3846 \cs_new_protected:Npn \@@_draw_blocks:
3847 { \seq_map_inline:Nn \g_@@_blocks_seq { \@@_Block_iii:nnnnnn ##1 } }

3848 \cs_new_protected:Npn \@@_Block_iii:nnnnnn #1 #2 #3 #4 #5 #6
3849 {
```


The group is for the keys (even if, as of now, there is only one key, called `tikz` and not documented).

```

3850 \group_begin:
3851 \keys_set:nn { NiceMatrix / Block } { #5 }

3852 \cs_set_protected:Npn \diagbox ##1 ##2
3853 {
3854   \tl_gput_right:Nx \g_@@_internal_code_after_tl
3855   {
3856     \@@_actually_diagbox:nnnnnn
3857     { #1 } { #2 } { #3 } { #4 } { ##1 } { ##2 }
3858   }
3859 }

3860 \bool_lazy_or:nnTF
3861 { \int_compare_p:nNn { #3 } > \g_@@_row_total_int }
3862 { \int_compare_p:nNn { #4 } > \c@jCol }
3863 { \msg_error:nnnn { nicematrix } { Block-too-large } { #1 } { #2 } }
3864 {

```

We put the contents of the cell in the box `\l_@@_cell_box` because we want the command `\rotate` used in the content to be able to rotate the box.

```

3865 \hbox_set:Nn \l_@@_cell_box { #6 }

```

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.

```

3866 \pgfpicture
3867 \pgfrememberpicturepositiononpagetrue
3868 \pgf@relevantforpicturesizefalse
3869 \@@_qpoint:n { row - #1 }
3870 \dim_set_eq:NN \l_tmpa_dim \pgf@y
3871 \@@_qpoint:n { col - #2 }
3872 \dim_set_eq:NN \l_tmpb_dim \pgf@x
3873 \@@_qpoint:n { row - \@@_succ:n { #3 } }
3874 \dim_set_eq:NN \l_tmpc_dim \pgf@y
3875 \@@_qpoint:n { col - \@@_succ:n { #4 } }
3876 \dim_set_eq:NN \l_tmpd_dim \pgf@x

```

We construct the node for the block with the name `(#1-#2-block)`.

The function `\@@_pgf_rect_node:nnnnn` takes as arguments the name of the node and the four coordinates of two opposite corner points of the rectangle.

```

3877 \begin { pgfscope }
3878 \exp_args:Nx \pgfset { \l_@@_tikz_tl }

```

```

3879 \@@_pgf_rect_node:nnnnn
3880 { \@@_env: - #1 - #2 - block }
3881 \l_tmpb_dim \l_tmpa_dim \l_tmpd_dim \l_tmpe_dim
3882 \end { pgfscope }

```

We construct the short node.

```

3883 \dim_set_eq:NN \l_tmpb_dim \c_max_dim
3884 \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
3885 {

```

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.

```

3886 \cs_if_exist:cT
3887 { pgf @ sh @ ns @ \@@_env: - ##1 - #2 }
3888 {
3889 \pgfpointanchor { \@@_env: - ##1 - #2 } { west }
3890 \dim_set:Nn \l_tmpb_dim { \dim_min:nn \l_tmpb_dim \pgf@x }
3891 }
3892 }

```

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.

```

3893 \dim_compare:nNnT \l_tmpb_dim = \c_max_dim
3894 {
3895 \@@_qpoint:n { col - #2 }
3896 \dim_set_eq:NN \l_tmpb_dim \pgf@x
3897 }
3898 \dim_set:Nn \l_tmpd_dim { - \c_max_dim }
3899 \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
3900 {
3901 \cs_if_exist:cT
3902 { pgf @ sh @ ns @ \@@_env: - ##1 - #4 }
3903 {
3904 \pgfpointanchor { \@@_env: - ##1 - #4 } { east }
3905 \dim_set:Nn \l_tmpd_dim { \dim_max:nn \l_tmpd_dim \pgf@x }
3906 }
3907 }
3908 \dim_compare:nNnT \l_tmpd_dim = { - \c_max_dim }
3909 {
3910 \@@_qpoint:n { col - \@@_succ:n { #4 } }
3911 \dim_set_eq:NN \l_tmpd_dim \pgf@x
3912 }
3913 \@@_pgf_rect_node:nnnnn
3914 { \@@_env: - #1 - #2 - block - short }
3915 \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 as arguments the name of the node and two PGF points.

```

3916 \bool_if:NT \l_@@_medium_nodes_bool
3917 {
3918 \@@_pgf_rect_node:nnn
3919 { \@@_env: - #1 - #2 - block - medium }
3920 { \pgfpointanchor { \@@_env: - #1 - #2 - medium } { north-west } }
3921 { \pgfpointanchor { \@@_env: - #3 - #4 - medium } { south-east } }
3922 }

```

Now, we will put the label of the block.

```

3923 \int_compare:nNnTF { #1 } = { #3 }
3924 {

```

We take into account the case of a block of one row in the “first row” or the “end row”.

```

3925 \int_compare:nNnTF { #1 } = 0
3926 { \l_@@_code_for_first_row_tl }
3927 {

```

```

3928         \int_compare:nNtT { #1 } = \l_@@_last_row_int
3929         \l_@@_code_for_last_row_tl
3930     }

```

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`.

```

3931     \pgfextracty \l_tmpa_dim { \@@_qpoint:n { row - #1 - base } }

```

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

```

3932     \@@_qpoint:n { #1 - #2 - block - short }

```

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

```

3933     \pgftransformshift { \pgfpoint \pgf@x \l_tmpa_dim }
3934     \pgfnode { rectangle } { base }
3935     { \box_use_drop:N \l_@@_cell_box } { } { }
3936 }

```

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

```

3937     {
3938         \pgftransformshift { \@@_qpoint:n { #1 - #2 - block - short } }
3939         \pgfnode { rectangle } { center }
3940         { \box_use_drop:N \l_@@_cell_box } { } { }
3941     }
3942     \endpgfpicture
3943 }
3944 \group_end:
3945 }

```

How to draw the dotted lines transparently

```

3946 \cs_set_protected:Npn \@@_renew_matrix:
3947 {
3948     \RenewDocumentEnvironment { pmatrix } { } {
3949         { \pNiceMatrix }
3950         { \endpNiceMatrix }
3951     }
3952     \RenewDocumentEnvironment { vmatrix } { } {
3953         { \vNiceMatrix }
3954         { \endvNiceMatrix }
3955     }
3956     \RenewDocumentEnvironment { Vmatrix } { } {
3957         { \VNiceMatrix }
3958         { \endVNiceMatrix }
3959     }
3960     \RenewDocumentEnvironment { bmatrix } { } {
3961         { \bNiceMatrix }
3962         { \endbNiceMatrix }
3963     }
3964     \RenewDocumentEnvironment { Bmatrix } { } {
3965         { \BNiceMatrix }
3966         { \endBNiceMatrix }
3967     }
3968 }

```

Automatic arrays

```

3964 \cs_new_protected:Npn \@@_set_size:n #1-#2 \q_stop
3965 {
3966     \int_set:Nn \l_@@_nb_rows_int { #1 }
3967     \int_set:Nn \l_@@_nb_cols_int { #2 }
3968 }
3969 \NewDocumentCommand \AutoNiceMatrixWithDelims { m m O { } m O { } m ! O { } }
3970 {
3971     \int_zero_new:N \l_@@_nb_rows_int

```

```

3972 \int_zero_new:N \l_@@_nb_cols_int
3973 \@@_set_size:n #4 \q_stop
3974 \begin { NiceArrayWithDelims } { #1 } { #2 }
3975 { * { \l_@@_nb_cols_int } { c } } [ #3 , #5 , #7 ]
3976 \int_compare:nNnT \l_@@_first_row_int = 0
3977 {
3978 \int_compare:nNnT \l_@@_first_col_int = 0 { & }
3979 \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
3980 \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
3981 }
3982 \prg_replicate:nn \l_@@_nb_rows_int
3983 {
3984 \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).

```

3985 \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { { } #6 & } #6
3986 \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
3987 }
3988 \int_compare:nNnT \l_@@_last_row_int > { -2 }
3989 {
3990 \int_compare:nNnT \l_@@_first_col_int = 0 { & }
3991 \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
3992 \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
3993 }
3994 \end { NiceArrayWithDelims }
3995 }
3996 \cs_set_protected:Npn \@@_define_com:nnn #1 #2 #3
3997 {
3998 \cs_set_protected:cpn { #1 AutoNiceMatrix }
3999 {
4000 \str_gset:Nx \g_@@_name_env_str { #1 AutoNiceMatrix }
4001 \AutoNiceMatrixWithDelims { #2 } { #3 }
4002 }
4003 }
4004 \@@_define_com:nnn p ( )
4005 \@@_define_com:nnn b [ ]
4006 \@@_define_com:nnn v | |
4007 \@@_define_com:nnn V \ | \ |
4008 \@@_define_com:nnn B \{ \}

```

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

```

4009 \NewDocumentCommand \AutoNiceMatrix { O { } m O { } m ! O { } }
4010 {
4011 \group_begin:
4012 \bool_set_true:N \l_@@_NiceArray_bool
4013 \AutoNiceMatrixWithDelims . . { #2 } { #4 } [ #1 , #3 , #5 ]
4014 \group_end:
4015 }

```

The redefinition of the command \dotfill

```

4016 \cs_set_eq:NN \@@_dotfill \dotfill
4017 \cs_new_protected:Npn \@@_dotfill:
4018 {

```

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}).

```

4019 \@@_dotfill
4020 \bool_if:NT \l_@@_NiceTabular_bool
4021 { \group_insert_after:N \@@_dotfill_ii: }
4022 { \group_insert_after:N \@@_dotfill_i: }

```

```

4023 }
4024 \cs_new_protected:Npn \@@_dotfill_ii: { \group_insert_after:N \@@_dotfill_iii: }
4025 \cs_new_protected:Npn \@@_dotfill_iii: { \group_insert_after:N \@@_dotfill_iiii: }

```

Now, if the box is not empty (unfortunately, we can't actually test whether the box is empty and that's why we only consider its width), we insert `\@@_dotfill` (which is the saved version of `\dotfill`) in the cell of the array, and it will extend, since it is no longer in `\l_@@_cell_box`.

```

4026 \cs_new_protected:Npn \@@_dotfill_iiii:
4027 { \dim_compare:nNnT { \box_wd:N \l_@@_cell_box } = \c_zero_dim \@@_dotfill }

```

The command `\diagbox`

The command `\diagbox` will be linked to `\diagbox:nn` in the environments of `nicematrix`.

```

4028 \cs_new_protected:Npn \@@_diagbox:nn #1 #2
4029 {
4030   \tl_gput_right:Nx \g_@@_internal_code_after_tl
4031   {
4032     \@@_actually_diagbox:nnnnnn
4033     { \int_use:N \c@iRow }
4034     { \int_use:N \c@jCol }
4035     { \int_use:N \c@iRow }
4036     { \int_use:N \c@jCol }
4037     { #1 }
4038     { #2 }
4039   }

```

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 `hvlines-except-corners`.

```

4040   \seq_gput_right:Nx \g_@@_pos_of_blocks_seq
4041   {
4042     { \int_use:N \c@iRow }
4043     { \int_use:N \c@jCol }
4044     { \int_use:N \c@iRow }
4045     { \int_use:N \c@jCol }
4046   }
4047 }

```

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.

```

4048 \cs_new_protected:Npn \@@_actually_diagbox:nnnnnn #1 #2 #3 #4 #5 #6
4049 {
4050   \pgfpicture
4051   \pgf@relevantforpicturesizefalse
4052   \pgfrememberpicturepositiononpagetrue
4053   \@@_qpoint:n { row - #1 }
4054   \dim_set_eq:NN \l_tmpa_dim \pgf@y
4055   \@@_qpoint:n { col - #2 }
4056   \dim_set_eq:NN \l_tmpb_dim \pgf@x
4057   \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
4058   \@@_qpoint:n { row - \@@_succ:n { #3 } }
4059   \dim_set_eq:NN \l_tmpc_dim \pgf@y
4060   \@@_qpoint:n { col - \@@_succ:n { #4 } }
4061   \dim_set_eq:NN \l_tmpd_dim \pgf@x
4062   \pgfpathlineto { \pgfpoint \l_tmpd_dim \l_tmpc_dim }
4063   {

```

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.

```

4064   \CT@arc@
4065   \pgfsetroundcap
4066   \pgfusepathqstroke

```

```

4067 }
4068 \pgfset { inner~sep = 1 pt }
4069 \pgfscope
4070 \pgftransformshift { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
4071 \pgfnode { rectangle } { south~west }
4072 { \@@_math_toggle_token: #5 \@@_math_toggle_token: } { } { }
4073 \endpgfscope
4074 \pgftransformshift { \pgfpoint \l_tmpd_dim \l_tmpe_dim }
4075 \pgfnode { rectangle } { north~east }
4076 { \@@_math_toggle_token: #6 \@@_math_toggle_token: } { } { }
4077 \endpgfpicture
4078 }

```

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. 78.

The command `\CodeAfter` catches everything until the end of the current environment (of `nicematrix`). First, we go until the next command `\end`.

```

4079 \cs_new_protected:Npn \@@_CodeAfter:n #1 \end
4080 {
4081   \tl_gput_right:Nn \g_@@_code_after_tl { #1 }
4082   \@@_CodeAfter_i:n
4083 }

```

We catch the argument of the command `\end` (in `#1`).

```

4084 \cs_new_protected:Npn \@@_CodeAfter_i:n #1
4085 {

```

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.

```

4086   \str_if_eq:eeTF \@@_currenvir { #1 }
4087   { \end { #1 } }

```

If this is not the `\end` we are looking for, we put those tokens in `\g_@@_code_after_tl` and we go on searching for the next command `\end` with a recursive call to the command `\@@_CodeAfter:n`.

```

4088   {
4089     \tl_gput_right:Nn \g_@@_code_after_tl { \end { #1 } }
4090     \@@_CodeAfter:n
4091   }
4092 }

```

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.

```

4093 \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.

```

4094 \bool_new:N \c_@@_footnote_bool

```

```

4095 \@@_msg_new:nnn { Unknown~option~for~package }
4096 {
4097   The~option~'\l_keys_key_tl'~is~unknown. \\
4098   If~you~go~on,~it~will~be~ignored. \\
4099   For~a~list~of~the~available~options,~type~H~<return>.
4100 }
4101 {
4102   The~available~options~are~(in~alphabetic~order):~
4103   define-L-C-R,~
4104   footnote,~
4105   footnotehyper,~
4106   renew-dots,~
4107   renew-matrix~and~
4108   transparent.
4109 }
4110 \keys_define:nn { NiceMatrix / Package }
4111 {
4112   define-L-C-R .bool_set:N = \c_@@_define_L_C_R_bool ,
4113   define-L-C-R .default:n = true ,
4114   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
4115   renew-dots .value_forbidden:n = true ,
4116   renew-matrix .code:n = \@@_renew_matrix: ,
4117   renew-matrix .value_forbidden:n = true ,
4118   transparent .meta:n = { renew-dots , renew-matrix } ,
4119   transparent .value_forbidden:n = true,
4120   footnote .bool_set:N = \c_@@_footnote_bool ,
4121   footnotehyper .bool_set:N = \c_@@_footnotehyper_bool ,
4122   unknown .code:n = \@@_error:n { Unknown~option~for~package }
4123 }
4124 \ProcessKeysOptions { NiceMatrix / Package }

4125 \@@_msg_new:nn { footnote~with~footnotehyper~package }
4126 {
4127   You~can't~use~the~option~'footnote'~because~the~package~
4128   footnotehyper~has~already~been~loaded.~
4129   If~you~want,~you~can~use~the~option~'footnotehyper'~and~the~footnotes~
4130   within~the~environments~of~nicematrix~will~be~extracted~with~the~tools~
4131   of~the~package~footnotehyper.\\
4132   If~you~go~on,~the~package~footnote~won't~be~loaded.
4133 }
4134 \@@_msg_new:nn { footnotehyper~with~footnote~package }
4135 {
4136   You~can't~use~the~option~'footnotehyper'~because~the~package~
4137   footnote~has~already~been~loaded.~
4138   If~you~want,~you~can~use~the~option~'footnote'~and~the~footnotes~
4139   within~the~environments~of~nicematrix~will~be~extracted~with~the~tools~
4140   of~the~package~footnote.\\
4141   If~you~go~on,~the~package~footnotehyper~won't~be~loaded.
4142 }

4143 \bool_if:NT \c_@@_footnote_bool
4144 {
4145   \@ifclassloaded { beamer }
4146   { \msg_info:nn { nicematrix } { Option~incompatible~with~Beamer } }
4147   {
4148     \@ifpackageloaded { footnotehyper }
4149     { \@@_error:n { footnote~with~footnotehyper~package } }
4150     { \usepackage { footnote } }
4151   }
4152 }

```

```

4153 \bool_if:NT \c_@@_footnotehyper_bool
4154 {
4155   \@ifclassloaded { beamer }
4156   { \@@_info:n { Option-incompatible-with-Beamer } }
4157   {
4158     \@ifpackageloaded { footnote }
4159     { \@@_error:n { footnotehyper-with-footnote-package } }
4160     { \usepackage { footnotehyper } }
4161   }
4162   \bool_set_true:N \c_@@_footnote_bool
4163 }

```

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 creates a sequence of strings (`str`) from a `clist`. Be careful: we use `\seq_set_map:Nnn` and the name of that function will maybe change to `\eq_set_map_x:Nnn`.

```

4164 \cs_new_protected:Npn \@@_set_seq_of_str_from_clist:Nn #1 #2
4165 {
4166   \seq_set_from_clist:Nn #1 { #2 }
4167   \seq_set_map:Nnn #1 #1 { \tl_to_str:n { ##1 } }
4168 }
4169 \@@_set_seq_of_str_from_clist:Nn \c_@@_types_of_matrix_seq
4170 {
4171   NiceMatrix ,
4172   pNiceMatrix , bNiceMatrix , vNiceMatrix, BNiceMatrix, VNiceMatrix
4173 }

```

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`.

```

4174 \cs_new_protected:Npn \@@_error_too_much_cols:
4175 {
4176   \seq_if_in:NVTF \c_@@_types_of_matrix_seq \g_@@_name_env_str
4177   {
4178     \int_compare:nNnTF \l_@@_last_col_int = { -2 }
4179     { \@@_fatal:n { too-much-cols-for-matrix } }
4180     {
4181       \bool_if:NF \l_@@_last_col_without_value_bool
4182       { \@@_fatal:n { too-much-cols-for-matrix-with-last-col } }
4183     }
4184   }
4185   { \@@_fatal:n { too-much-cols-for-array } }
4186 }

```

The following command must *not* be protected since it's used in an error message.

```

4187 \cs_new:Npn \@@_message_hdotsfor:
4188 {
4189   \tl_if_empty:VF \g_@@_HVdotsfor_lines_tl
4190   { ~Maybe-your-use-of~\token_to_str:N \Hdotsfor\ is-incorrect.}
4191 }
4192 \@@_msg_new:nn { too-much-cols-for-matrix-with-last-col }
4193 {
4194   You-try-to-use-more-columns-than-allowed-by-your~
4195   \@@_full_name_env:.\@@_message_hdotsfor:\ The-maximal-number-of~
4196   columns-is~\int_eval:n { \l_@@_last_col_int - 1 }~(plus-the-potential~
4197   exterior-ones).~This-error-is-fatal.
4198 }

```



```

4199 \@@_msg_new:nn { too-much-cols-for-matrix }
4200 {
4201   You-try-to-use-more-columns-than-allowed-by-your-
4202   \@@_full_name_env:.\@@_message_hdotsfor:\ Recall-that-the-maximal-
4203   number-of-columns-for-a-matrix-is-fixed-by-the-LaTeX-counter-
4204   'MaxMatrixCols'.~Its-actual-value-is~\int_use:N \c@MaxMatrixCols.~
4205   This-error-is-fatal.
4206 }

```

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`.

```

4207 \@@_msg_new:nn { too-much-cols-for-array }
4208 {
4209   You-try-to-use-more-columns-than-allowed-by-your-
4210   \@@_full_name_env:.\@@_message_hdotsfor:\ The-maximal-number-of-columns-is-
4211   \int_eval:n { \c@jCol - 1 }~(plus-the-potential-exterior-ones).~
4212   This-error-is-fatal.
4213 }
4214 \@@_msg_new:nn { in-first-col }
4215 {
4216   You-can't-use-the-command~#1 in-the-first-column-(number~0)-of-the-array.\\
4217   If-you-go-on,~this-command-will-be-ignored.
4218 }
4219 \@@_msg_new:nn { in-last-col }
4220 {
4221   You-can't-use-the-command~#1 in-the-last-column-(exterior)-of-the-array.\\
4222   If-you-go-on,~this-command-will-be-ignored.
4223 }
4224 \@@_msg_new:nn { in-first-row }
4225 {
4226   You-can't-use-the-command~#1 in-the-first-row-(number~0)-of-the-array.\\
4227   If-you-go-on,~this-command-will-be-ignored.
4228 }
4229 \@@_msg_new:nn { in-last-row }
4230 {
4231   You-can't-use-the-command~#1 in-the-last-row-(exterior)-of-the-array.\\
4232   If-you-go-on,~this-command-will-be-ignored.
4233 }
4234 \@@_msg_new:nn { option-S-without-siunitx }
4235 {
4236   You-can't-use-the-option-'S'-in-your-environment-\@@_full_name_env:
4237   because-you-have-not-loaded-siunitx.\\
4238   If-you-go-on,~this-option-will-be-ignored.
4239 }
4240 \@@_msg_new:nn { bad-option-for-line-style }
4241 {
4242   Since-you-haven't-loaded-Tikz,~the-only-value-you-can-give-to-'line-style'~
4243   is-'standard'.~If-you-go-on,~this-option-will-be-ignored.
4244 }
4245 \@@_msg_new:nn { Unknown-option-for-xdots }
4246 {
4247   As-for-now-there-is-only-three-options-available-here:~'color',~'line-style'~
4248   and~'shorten'~(and-you-try-to-use~'\l_keys_key_tl').~If-you-go-on,~
4249   this-option-will-be-ignored.
4250 }
4251 \@@_msg_new:nn { ampersand-in-light-syntax }
4252 {
4253   You-can't-use-an-ampersand~(\token_to_str &)~to-separate-columns-because
4254   ~you-have-used-the-option-'light-syntax'.~This-error-is-fatal.
4255 }

```

```

4256 \@@_msg_new:nn { double-backslash-in-light-syntax }
4257 {
4258   You~can't~use~\token_to_str:N \~to~separate~rows~because~you~have~used~
4259   the~option~'light-syntax'.~You~must~use~the~character~'\l_@@_end_of_row_tl'~
4260   (set~by~the~option~'end-of-row').~This~error~is~fatal.
4261 }
4262 \@@_msg_new:nn { standard-cline-in-document }
4263 {
4264   The~key~'standard-cline'~is~available~only~in~the~preamble.\\
4265   If~you~go~on~this~command~will~be~ignored.
4266 }
4267 \@@_msg_new:nn { bad-value-for-baseline }
4268 {
4269   The~value~given~to~'baseline'~(\int_use:N \l_tmpa_int)~is~not~
4270   valid.~The~value~must~be~between~\int_use:N \l_@@_first_row_int\ and~
4271   \int_use:N \g_@@_row_total_int\ or~equal~to~'t',~'c'~or~'b'.\\
4272   If~you~go~on,~a~value~of~1~will~be~used.
4273 }
4274 \@@_msg_new:nn { empty-environment }
4275 { Your~\@@_full_name_env:\ is~empty.~This~error~is~fatal. }
4276 \@@_msg_new:nn { unknown-cell-for-line-in-code-after }
4277 {
4278   Your~command~\token_to_str:N\line\{#1\}\{#2\}~in~the~'code-after'~
4279   can't~be~executed~because~a~cell~doesn't~exist.\\
4280   If~you~go~on~this~command~will~be~ignored.
4281 }
4282 \@@_msg_new:nn { last-col-non-empty-for-NiceArray }
4283 {
4284   In~the~\@@_full_name_env:,~you~must~use~the~option~
4285   'last-col'~without~value.\\
4286   However,~you~can~go~on~for~this~time~
4287   (the~value~'\l_keys_value_tl'~will~be~ignored).
4288 }
4289 \@@_msg_new:nn { last-col-non-empty-for-NiceMatrixOptions }
4290 {
4291   In~\NiceMatrixoptions,~you~must~use~the~option~
4292   'last-col'~without~value.\\
4293   However,~you~can~go~on~for~this~time~
4294   (the~value~'\l_keys_value_tl'~will~be~ignored).
4295 }
4296 \@@_msg_new:nn { Block-too-large }
4297 {
4298   You~try~to~draw~a~block~in~the~cell~#1-#2~of~your~matrix~but~the~matrix~is~
4299   too~small~for~that~block. \\
4300 }
4301 \@@_msg_new:nn { angle-option-in-NiceTabular }
4302 {
4303   You~should~not~the~option~between~angle~brackets~(<~and~>)~for~a~command~
4304   \token_to_str:N \Block\ in~\{NiceTabular\}.~However,~you~can~go~on.
4305 }
4306 \@@_msg_new:nn { tabularnote-forbidden }
4307 {
4308   You~can't~use~the~command~\token_to_str:N\tabularnote\
4309   ~in~a~\@@_full_name_env:.~This~command~is~available~only~in~
4310   \{NiceTabular\},~\{NiceArray\}~and~\{NiceMatrix\}. \\
4311   If~you~go~on,~this~command~will~be~ignored.
4312 }
4313 \@@_msg_new:nn { bottomule-without-booktabs }
4314 {

```

```

4315   You~can't~use~the~option~'tabular/bottomrule'~because~you~haven't~
4316   loaded~'booktabs'.\\
4317   If~you~go~on,~this~option~will~be~ignored.
4318 }
4319 \@@_msg_new:nn { enumitem~not~loaded }
4320 {
4321   You~can't~use~the~command~\token_to_str:N\tabularnote\
4322   ~because~you~haven't~loaded~'enumitem'.\\
4323   If~you~go~on,~this~command~will~be~ignored.
4324 }
4325 \@@_msg_new:nn { Wrong~last~row }
4326 {
4327   You~have~used~'last-row=\int_use:N \l_@@_last_row_int'~but~your~
4328   \@@_full_name_env:\ seems~to~have~\int_use:N \c@iRow \ rows.~
4329   If~you~go~on,~the~value~of~\int_use:N \c@iRow \ will~be~used~for~
4330   last~row.~You~can~avoid~this~problem~by~using~'last-row'~
4331   without~value~(more~compilations~might~be~necessary).
4332 }
4333 \@@_msg_new:nn { Yet~in~env }
4334 { Environments~of~nicematrix~can't~be~nested.\\ This~error~is~fatal. }
4335 \@@_msg_new:nn { Outside~math~mode }
4336 {
4337   The~\@@_full_name_env:\ can~be~used~only~in~math~mode~
4338   (and~not~in~\token_to_str:N \vcenter).\\
4339   This~error~is~fatal.
4340 }
4341 \@@_msg_new:nn { Bad~value~for~letter~for~dotted~lines }
4342 {
4343   The~value~of~key~'\l_keys_key_tl'~must~be~of~length~1.\\
4344   If~you~go~on,~it~will~be~ignored.
4345 }
4346 \@@_msg_new:nnn { Unknown~key~for~notes }
4347 {
4348   The~key~'\l_keys_key_tl'~is~unknown.\\
4349   If~you~go~on,~it~will~be~ignored. \\
4350   For~a~list~of~the~available~keys~about~notes,~type~H~<return>.
4351 }
4352 {
4353   The~available~options~are~(in~alphabetic~order):~
4354   bottomrule,~
4355   code~after,~
4356   code~before,~
4357   enumitem~keys,~
4358   enumitem~keys~para,~
4359   para,~
4360   label~in~list,~
4361   label~in~tabular~and~
4362   style.
4363 }
4364 \@@_msg_new:nnn { Unknown~key~for~NiceMatrixOptions }
4365 {
4366   The~key~'\l_keys_key_tl'~is~unknown~for~the~command~
4367   \token_to_str:N \NiceMatrixOptions. \\
4368   If~you~go~on,~it~will~be~ignored. \\
4369   For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
4370 }
4371 {
4372   The~available~options~are~(in~alphabetic~order):~
4373   allow~duplicate~names,~
4374   code~for~first~col,~
4375   cell~space~bottom~limit,~

```

```

4376   cell-space-top-limit,~
4377   code-for-first-row,~
4378   code-for-last-col,~
4379   code-for-last-row,~
4380   create-extra-nodes,~
4381   create-medium-nodes,~
4382   create-large-nodes,~
4383   end-of-row,~
4384   first-col,~
4385   first-row,~
4386   hlines,~
4387   hvlines,~
4388   hvlines-except-corners,~
4389   last-col,~
4390   last-row,~
4391   left-margin,~
4392   letter-for-dotted-lines,~
4393   light-syntax,~
4394   notes~(several subkeys),~
4395   nullify-dots,~
4396   renew-dots,~
4397   renew-matrix,~
4398   right-margin,~
4399   small,~
4400   transparent,~
4401   vlines,~
4402   xdots/color,~
4403   xdots/shorten~and~
4404   xdots/line-style.
4405 }

4406 \@@_msg_new:nnn { Unknown~option~for~NiceArray }
4407 {
4408   The~option~'\l_keys_key_tl'~is~unknown~for~the~environment~
4409   \{NiceArray\}. \\
4410   If~you~go~on,~it~will~be~ignored. \\
4411   For~a~list~of~the~*principal*~available~options,~type~H~<return>.
4412 }
4413 {
4414   The~available~options~are~(in~alphabetic~order):~
4415   b,~
4416   baseline,~
4417   c,~
4418   cell-space-bottom-limit,~
4419   cell-space-top-limit,~
4420   code-after,~
4421   code-for-first-col,~
4422   code-for-first-row,~
4423   code-for-last-col,~
4424   code-for-last-row,~
4425   columns-width,~
4426   create-extra-nodes,~
4427   create-medium-nodes,~
4428   create-large-nodes,~
4429   extra-left-margin,~
4430   extra-right-margin,~
4431   first-col,~
4432   first-row,~
4433   hlines,~
4434   hvlines,~
4435   last-col,~
4436   last-row,~
4437   left-margin,~
4438   light-syntax,~

```

```

4439     name,~
4440     notes/bottomrule,~
4441     notes/para,~
4442     nullify-dots,~
4443     renew-dots,~
4444     right-margin,~
4445     rules/color,~
4446     rules/width,~
4447     small,~
4448     t,~
4449     vl原因,~
4450     xdots/color,~
4451     xdots/shorten~and~
4452     xdots/line-style.
4453 }

```

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).

```

4454 \@@_msg_new:nnn { Unknown~option~for~NiceMatrix }
4455 {
4456     The~option~'\l_keys_key_tl'~is~unknown~for~the~
4457     \@@_full_name_env:. \\\
4458     If~you~go~on,~it~will~be~ignored. \\\
4459     For~a~list~of~the~*principal*~available~options,~type~H~<return>.
4460 }
4461 {
4462     The~available~options~are~(in~alphabetic~order):~
4463     b,~
4464     baseline,~
4465     c,~
4466     cell-space-bottom-limit,~
4467     cell-space-top-limit,~
4468     code-after,~
4469     code-for-first-col,~
4470     code-for-first-row,~
4471     code-for-last-col,~
4472     code-for-last-row,~
4473     columns-width,~
4474     create-extra-nodes,~
4475     create-medium-nodes,~
4476     create-large-nodes,~
4477     extra-left-margin,~
4478     extra-right-margin,~
4479     first-col,~
4480     first-row,~
4481     hlines,~
4482     hvlines,~
4483     l~(=L),~
4484     last-col,~
4485     last-row,~
4486     left-margin,~
4487     light-syntax,~
4488     name,~
4489     nullify-dots,~
4490     r~(=R),~
4491     renew-dots,~
4492     right-margin,~
4493     rules/color,~
4494     rules/width,~
4495     S,~
4496     small,~
4497     t,~
4498     vl原因,~

```

```

4499     xdots/color,~
4500     xdots/shorten~and~
4501     xdots/line-style.
4502 }

4503 \@@_msg_new:nnn { Unknown~option~for~NiceTabular }
4504 {
4505     The~option~'\l_keys_key_tl'~is~unknown~for~the~environment~
4506     \{NiceTabular\}.  \\\
4507     If~you~go~on,~it~will~be~ignored.  \\\
4508     For~a~list~of~the~*principal*~available~options,~type~H~<return>.
4509 }
4510 {
4511     The~available~options~are~(in~alphabetic~order):~
4512     b,~
4513     baseline,~
4514     c,~
4515     cell-space-bottom-limit,~
4516     cell-space-top-limit,~
4517     code-after,~
4518     code-for-first-col,~
4519     code-for-first-row,~
4520     code-for-last-col,~
4521     code-for-last-row,~
4522     columns-width,~
4523     create-extra-nodes,~
4524     create-medium-nodes,~
4525     create-large-nodes,~
4526     extra-left-margin,~
4527     extra-right-margin,~
4528     first-col,~
4529     first-row,~
4530     hlines,~
4531     hvlines,~
4532     last-col,~
4533     last-row,~
4534     left-margin,~
4535     light-syntax,~
4536     name,~
4537     notes/bottomrule,~
4538     notes/para,~
4539     nullify-dots,~
4540     renew-dots,~
4541     right-margin,~
4542     rules/color,~
4543     rules/width,~
4544     t,~
4545     vlines,~
4546     xdots/color,~
4547     xdots/shorten~and~
4548     xdots/line-style.
4549 }

4550 \@@_msg_new:nnn { Duplicate~name }
4551 {
4552     The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
4553     the~same~environment~name~twice.~You~can~go~on,~but,~
4554     maybe,~you~will~have~incorrect~results~especially~
4555     if~you~use~'columns-width=auto'.~If~you~don't~want~to~see~this~
4556     message~again,~use~the~option~'allow-duplicate-names'.  \\\
4557     For~a~list~of~the~names~already~used,~type~H~<return>.  \\\
4558 }
4559 {
4560     The~names~already~defined~in~this~document~are:~
4561     \seq_use:Nnnn \g_@@_names_seq { ,~ } { ,~ } { ~and~ }.

```

```

4562 }
4563 \@@_msg_new:nn { Option~auto~for~columns~width }
4564 {
4565     You~can't~give~the~value~'auto'~to~the~option~'columns~width'~here.~
4566     If~you~go~on,~the~option~will~be~ignored.
4567 }

```

17 History

Changes between versions 1.0 and 1.1

The dotted lines are no longer drawn with Tikz nodes but with Tikz circles (for efficiency).
Modification of the code which is now twice faster.

Changes between versions 1.1 and 1.2

New environment `{NiceArray}` with column types L, C and R.

Changes between version 1.2 and 1.3

New environment `{pNiceArrayC}` and its variants.

Correction of a bug in the definition of `{BNiceMatrix}`, `{vNiceMatrix}` and `{VNiceMatrix}` (in fact, it was a typo).

Options are now available locally in `{pNiceMatrix}` and its variants.

The names of the options are changed. The old names were names in “camel style”.

Changes between version 1.3 and 1.4

The column types `w` and `W` can now be used in the environments `{NiceArray}`, `{pNiceArrayC}` and its variants with the same meaning as in the package `array`.

New option `columns-width` to fix the same width for all the columns of the array.

Changes between version 1.4 and 2.0

The versions 1.0 to 1.4 of `nicematrix` were focused on the continuous dotted lines whereas the version 2.0 of `nicematrix` provides different features to improve the typesetting of mathematical matrices.

Changes between version 2.0 and 2.1

New implementation of the environment `{pNiceArrayRC}`. With this new implementation, there is no restriction on the width of the columns.

The package `nicematrix` no longer loads `mathtools` but only `amsmath`.

Creation of “medium nodes” and “large nodes”.

Changes between version 2.1 and 2.1.1

Small corrections: for example, the option `code-for-first-row` is now available in the command `\NiceMatrixOptions`.

Following a discussion on TeX StackExchange⁴⁵, Tikz externalization is now deactivated in the environments of the package `nicematrix`.⁴⁶

Changes between version 2.1.2 and 2.1.3

When searching the end of a dotted line from a command like `\Cdots` issued in the “main matrix” (not in the exterior column), the cells in the exterior column are considered as outside the matrix. That means that it’s possible to do the following matrix with only a `\Cdots` command (and a single `\Vdots`).

$$\left(\begin{array}{ccc} & C_j & \\ 0 & \vdots & 0 \\ & a \cdots \cdots & \\ 0 & & 0 \end{array} \right) L_i$$

Changes between version 2.1.3 and 2.1.4

Replacement of some options `0 { }` in commands and environments defined with `xparse` by `! 0 { }` (because a recent version of `xparse` introduced the specifier `!` and modified the default behaviour of the last optional arguments).

See www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end

Changes between version 2.1.4 and 2.1.5

Compatibility with the classes `revtex4-1` and `revtex4-2`.

Option `allow-duplicate-names`.

Changes between version 2.1.5 and 2.2

Possibility to draw horizontal dotted lines to separate rows with the command `\hdottedline` (similar to the classical command `\hline` and the command `\hdashline` of `arydshln`).

Possibility to draw vertical dotted lines to separate columns with the specifier “:” in the preamble (similar to the classical specifier “|” and the specifier “:” of `arydshln`).

Changes between version 2.2 and 2.2.1

Improvement of the vertical dotted lines drawn by the specifier “:” in the preamble.

Modification of the position of the dotted lines drawn by `\hdottedline`.

Changes between version 2.2.1 and 2.3

Compatibility with the column type `S` of `siunitx`.

Option `hlines`.

Changes between version 2.3 and 3.0

Modification of `\Hdotsfor`. Now `\Hdotsfor` erases the `\vlines` (of “|”) as `\hdotsfor` does.

Composition of exterior rows and columns on the four sides of the matrix (and not only on two sides) with the options `first-row`, `last-row`, `first-col` and `last-col`.

⁴⁵cf. tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package

⁴⁶Before 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 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 StackExchange⁴⁷, optimization of Tikz externalization is disabled in the environments of `nicematrix` when the class `standalone` or the package `standalone` is used.

Changes between version 3.4 and 3.5

Correction on a bug on the two previous versions where the `code-after` was not executed.

Changes between version 3.5 and 3.6

LaTeX counters `iRow` and `jCol` available in the cells of the array.

Addition of `\normalbaselines` before the construction of the array: in environments like `{align}` of `amsmath` the value of `\baselineskip` is changed and if the options `first-row` and `last-row` were used in an environment of `nicematrix`, the position of the delimiters was wrong.

A warning is written in the `.log` file if an obsolete environment is used.

There is no longer artificial errors `Duplicate-name` in the environments of `amsmath`.

Changes between version 3.6 and 3.7

The four “corners” of the matrix are correctly protected against the four codes: `code-for-first-col`, `code-for-last-col`, `code-for-first-row` and `code-for-last-row`.

New command `\pAutoNiceMatrix` and its variants (suggestion of Christophe Bal).

⁴⁷cf. tex.stackexchange.com/questions/510841/nicematrix-and-tikz-external-optimize

Changes between version 3.7 and 3.8

New programming for the command `\Block` when the block has only one row. With this programming, the vertical rules drawn by the specifier “|” at the end of the block is actually drawn. In previous versions, they were not because the block of one row was constructed with `\multicolumn`. An error is raised when an obsolete environment is used.

Changes between version 3.8 and 3.9

New commands `\NiceMatrixLastEnv` and `\OnlyMainNiceMatrix`.
New options `create-medium-nodes` and `create-large-nodes`.

Changes between version 3.9 and 3.10

New option `light-syntax` (and `end-of-row`).
New option `dotted-lines-margin` for fine tuning of the dotted lines.

Changes between versions 3.10 and 3.11

Correction of a bug linked to `first-row` and `last-row`.

Changes between versions 3.11 and 3.12

Command `\rotate` in the cells of the array.
Options `vlines`, `hlines` and `hvlines`.
Option `baseline` pour `{NiceArray}` (not for the other environments).
The name of the Tikz nodes created by the command `\Block` has changed: when the command has been issued in the cell $i-j$, the name is $i-j$ -block and, if the creation of the “medium nodes” is required, a node $i-j$ -block-medium is created.
If the user try to use more columns than allowed by its environment, an error is raised by `nicematrix` (instead of a low-level error).
The package must be loaded with the option `obsolete-environments` if we want to use the deprecated environments.

Changes between versions 3.12 and 3.13

The behaviour of the command `\rotate` is improved when used in the “last row”.
The option `dotted-lines-margin` has been renamed in `xdots/shorten` and the options `xdots/color` and `xdots/line-style` have been added for a complete customization of the dotted lines.
In the environments without preamble (`{NiceMatrix}`, `{pNiceMatrix}`, etc.), it’s possible to use the options `l` (=L) or `r` (=R) to specify the type of the columns.
The starred versions of the commands `\Cdots`, `\Ldots`, `\Vdots`, `\Ddots` and `\Iddots` are deprecated since the version 3.1 of `nicematrix`. Now, one should load `nicematrix` with the option `starred-commands` to avoid an error at the compilation.
The code of `nicematrix` no longer uses Tikz but only PGF. By default, Tikz is *not* loaded by `nicematrix`.

Changes between versions 3.13 and 3.14

Correction of a bug (question 60761504 on [stackoverflow](https://stackoverflow.com)).
Better error messages when the user uses `&` or `\\` when `light-syntax` is in force.

Changes between versions 3.14 and 3.15

It's possible to put labels on the dotted lines drawn by `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots`, `\Hdotsfor` and the command `\line` in the code-after with the tokens `_` and `^`.

The option `baseline` is now available in all the environments of `nicematrix`. Before, it was available only in `{NiceArray}`.

New 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 `hylvline` don't draw rules in the blocks (commands `\Block`) and in the virtual blocks corresponding to the dotted lines.

Changes between versions 4.1 and 4.2

It's now possible to write `\begin{pNiceMatrix}a&b\\c&d\end{pNiceMatrix}^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 `!\{qqquad}` is used in the preamble of the array.

It's now possible to use the command `\Block` in the "last row".

Changes between versions 4.3 and 4.4

New key `hvlines-except-corners`.

Changes between versions 4.4 and 5.0

Use of the standard column types `l`, `c` and `r` instead of `L`, `C` and `R`.

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

Command `\tabularnote`

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:		704, 894, 996, 1011, 1220, 1221, 1222, 2779
<code>\@@_Block:</code>	967, 3819	<code>\@@_CodeAfter:n</code> 971, 4079, 4090
<code>\@@_Block_i</code>	3820, 3821	<code>\@@_CodeAfter_i:n</code> 4082, 4084
<code>\@@_Block_ii:nnnnn</code>	3821, 3822	<code>\@@_Ddots</code> 960, 978, 2719
<code>\@@_Block_iii:nnnnnn</code>	3847, 3848	<code>\g_@@_Ddots_lines_tl</code> 1049, 2014
<code>\@@_Cdots</code>	958, 976, 2689	<code>\g_@@_HVdotsfor_lines_tl</code> 1051, 2012, 2814, 2890, 4189
<code>\g_@@_Cdots_lines_tl</code>	1046, 2016	<code>\@@_Hdotsfor:</code> 964, 981, 2803
<code>\@@_Cell:</code>	180,	<code>\@@_Hdotsfor:nnnn</code> 2816, 2828

\@@_Hdotsfor_i	2806, 2812	\l_@@_code_before_tl	499, 1147
\@@_Hspace:	963, 2764	\l_@@_code_for_first_col_tl	455, 1717
\@@_Iddots	961, 979, 2741	\l_@@_code_for_first_row_tl	459, 720, 3926
\g_@@_Iddots_lines_tl	1050, 2015	\l_@@_code_for_last_col_tl	457, 1757
\@@_Ldots	957, 975, 980, 2674	\l_@@_code_for_last_row_tl	461, 727, 3929
\g_@@_Ldots_lines_tl	1047, 2017	\g_@@_col_total_int	709, 987, 1287, 1651, 1652, 1683, 1688, 1693, 1694, 1747, 1873, 1876, 1881, 1888, 1932, 2357, 2949, 3646, 3656, 3690, 3777
\l_@@_NiceArray_bool	213, 294, 1091, 1202, 1229, 1241, 1296, 1785, 3238, 3239, 3258, 3259, 3289, 3290, 3348, 3349, 4012	\c_@@_colortbl_loaded_bool	81, 85, 923
\g_@@_NiceMatrixBlock_int	209, 3610, 3615, 3618, 3629	\@@_columncolor	1144, 3076
\l_@@_NiceTabular_bool	134, 142, 214, 711, 921, 1073, 1146, 1148, 1230, 1242, 1250, 1367, 1371, 1865, 2277, 2285, 2961, 3549, 3556, 3824, 4020	\l_@@_columns_width_dim	210, 511, 631, 1620, 1626, 3611, 3617
\@@_OnlyMainNiceMatrix:n	969, 3204, 3223	\g_@@_com_or_env_str	224, 227
\@@_OnlyMainNiceMatrix_i:n	3207, 3214, 3217	\@@_computations_for_large_nodes:	3717, 3730, 3735
\@@_Vdots	959, 977, 2704	\@@_computations_for_medium_nodes:	3637, 3706, 3716, 3727
\g_@@_Vdots_lines_tl	1048, 2013	\@@_compute_a_corner:nnnnnn	3430, 3431, 3432, 3433, 3436
\@@_Vdotsfor:	965, 2888	\@@_create_col_nodes:	1515, 1543, 1562
\@@_Vdotsfor:nnnn	2892, 2903	\@@_create_large_nodes:	1952, 3711
\@@_actually_diagbox:nnnnnn	3856, 4032, 4048	\@@_create_medium_and_large_nodes:	1949, 3722
\@@_actually_draw_Cdots:	2266, 2270	\@@_create_medium_nodes:	1950, 3701
\@@_actually_draw_Ddots:	2393, 2397	\@@_create_nodes:	3708, 3719, 3729, 3732, 3773
\@@_actually_draw_Iddots:	2444, 2448	\@@_create_row_node:	837, 869, 905
\@@_actually_draw_Ldots:	2224, 2228, 2879	\@@_cut_on_hyphen:w	3035, 3057, 3058, 3091, 3092, 3121, 3152, 3163
\@@_actually_draw_Vdots:	2320, 2324, 2954	\g_@@_ddots_int	1939, 2417, 2418
\@@_adapt_S_column:	150, 165, 1072	\@@_define_L_C_R:	190, 1215
\@@_after_array:	1390, 1869	\c_@@_define_L_C_R_bool	189, 1215, 2780, 4112
\@@_analyze_end:Nn	1511, 1556	\@@_define_columntype:nn	889, 989, 990, 991
\l_@@_argspec_tl	2672, 2673, 2674, 2689, 2704, 2719, 2741, 2810, 2811, 2812, 2886, 2887, 2888, 2984, 2985, 2986	\@@_define_com:nnn	3996, 4004, 4005, 4006, 4007, 4008
\@@_array:	823, 1512, 1539	\@@_define_env:n	1831, 1855, 1856, 1857, 1858, 1859, 1860
\l_@@_auto_columns_width_bool	399, 510, 1619, 1623, 3605	\g_@@_delta_x_one_dim	1941, 2420, 2430
\l_@@_baseline_str	387, 388, 503, 504, 505, 506, 834, 1298, 1315, 1318, 1319, 1320, 1398, 1404, 1409	\g_@@_delta_x_two_dim	1943, 2471, 2481
\@@_begin_of_NiceMatrix:nn	1838, 1842	\g_@@_delta_y_one_dim	1942, 2422, 2430
\@@_begin_of_row:	708, 732, 1705	\g_@@_delta_y_two_dim	1944, 2473, 2481
\l_@@_block_auto_columns_width_bool	1086, 1624, 3598, 3603, 3613, 3623	\@@_diagbox:nn	972, 4028
\g_@@_blocks_seq	248, 1089, 1975, 3834, 3847	\@@_dotfill	4016, 4019, 4027
\c_@@_booktabs_loaded_bool	23, 29, 904, 1458	\@@_dotfill:	970, 4017
\l_@@_cell_box	710, 756, 758, 764, 772, 773, 774, 775, 777, 780, 782, 784, 801, 906, 995, 1004, 1010, 1020, 1157, 1159, 1706, 1728, 1731, 1733, 1748, 1769, 1773, 2969, 2972, 2976, 3865, 3935, 3940, 4027	\@@_dotfill_i:	4022, 4024
\l_@@_cell_space_bottom_limit_dim	376, 443, 775	\@@_dotfill_ii:	4021, 4024, 4025
\l_@@_cell_space_top_limit_dim	375, 441, 773	\@@_dotfill_iii:	4025, 4026
\@@_cellcolor	1140, 3112, 3197, 3198	\@@_double_int_eval:n	2980, 2994, 2995
\g_@@_cells_seq	1550, 1551, 1552, 1554	\g_@@_dp_ante_last_row_dim	735, 939
\@@_chessboardcolors	1145, 3190	\g_@@_dp_last_row_dim	735, 736, 942, 943, 1158, 1159, 1354
\@@_cline	117, 956	\g_@@_dp_row_zero_dim	755, 756, 933, 934, 1310, 1338, 1347
\@@_cline_i:nn	118, 119, 127, 130	\@@_draw_Cdots:nnn	2252
\@@_cline_i:w	119, 120	\@@_draw_Ddots:nnn	2385
\g_@@_code_after_tl	231, 523, 1533, 1992, 1993, 4081, 4089	\@@_draw_Iddots:nnn	2436
\l_@@_code_before_bool	237, 500, 841, 1097, 1569, 1583, 1601, 1632, 1658, 1681, 1915	\@@_draw_Ldots:nnn	2210
		\@@_draw_Vdots:nnn	2306
		\@@_draw_blocks:	1975, 3846
		\@@_draw_dotted_lines:	1965, 2001
		\@@_draw_dotted_lines_i:	2004, 2008
		\@@_draw_hlines:	1969, 3250, 3281
		\@@_draw_hvlines:	1967, 3271

```

\@@_draw_hvlines_except_corners: 1972, 3427
\@@_draw_hvlines_i: ..... 3276, 3279
\@@_draw_hvlines_ii: ..... 3277, 3284, 3434
\@@_draw_line: ..... 2250,
2304, 2383, 2434, 2485, 2487, 3033, 3566, 3596
\@@_draw_line_ii:nn ..... 3013, 3017
\@@_draw_line_iii:nn ..... 3020, 3024
\@@_draw_non_standard_dotted_line: ..
..... 2493, 2495
\@@_draw_non_standard_dotted_line:n ..
..... 2498, 2501
\@@_draw_standard_dotted_line: . 2492, 2521
\@@_draw_standard_dotted_line_i: 2586, 2590
\@@_draw_vlines: ..... 1970, 3224, 3282
\g_@@_empty_cell_bool ..... 244,
779, 786, 2687, 2702, 2717, 2739, 2761, 2766
\l_@@_empty_corner_cells_seq .....
3305, 3311, 3318, 3362, 3368, 3375, 3429, 3491
\@@_end_Cell: ..... 182,
768, 899, 1000, 1015, 1220, 1221, 1222, 2782
\l_@@_end_of_row_tl .....
..... 407, 408, 449, 1535, 1536, 4259
\c_@@_endpgfortikzpicture_tl .....
..... 38, 42, 2005, 3021, 3535
\c_@@_enumitem_loaded_bool .....
..... 24, 32, 271, 550, 555, 566, 571
\@@_env: .....
.. 204, 208, 741, 747, 802, 808, 846, 852,
858, 1111, 1112, 1118, 1119, 1126, 1127,
1137, 1570, 1573, 1575, 1588, 1594, 1597,
1606, 1612, 1615, 1637, 1643, 1646, 1663,
1669, 1675, 1683, 1688, 1694, 1986, 2082,
2150, 2192, 2203, 2842, 2860, 2917, 2935,
3006, 3008, 3027, 3030, 3446, 3464, 3482,
3659, 3661, 3669, 3780, 3789, 3807, 3880,
3887, 3889, 3902, 3904, 3914, 3919, 3920, 3921
\g_@@_env_int .. 203, 204, 206, 1085, 1099,
1103, 1106, 1115, 1116, 1123, 1124, 1167,
1170, 1185, 1188, 1880, 1901, 1919, 1922, 3816
\@@_error:n ..... 12,
274, 295, 419, 428, 576, 619, 630, 639,
644, 664, 667, 674, 680, 686, 691, 700, 702,
1282, 1325, 1416, 1463, 3825, 4122, 4149, 4159
\@@_error:nn ..... 13,
518, 2677, 2680, 2692, 2695, 2707, 2710,
2723, 2724, 2729, 2730, 2745, 2746, 2751, 2752
\@@_error:nnn ..... 14, 3011
\@@_error_too_much_cols: ..... 1248, 4174
\@@_everycr: ..... 863, 928, 931
\@@_everycr_i: ..... 863, 864
\l_@@_exterior_arraycolsep_bool .....
..... 389, 627, 1232, 1244
\l_@@_extra_left_margin_dim .....
..... 405, 485, 1261, 1736
\l_@@_extra_right_margin_dim .....
..... 406, 486, 1273, 1777, 2360
\@@_extract_coords_values: .... 3798, 3805
\@@_fatal:n ..... 15, 218,
1076, 1520, 1524, 1526, 1559, 4179, 4182, 4185
\l_@@_final_i_int .....
..... 1955, 2029, 2034, 2037, 2062,
2070, 2074, 2083, 2091, 2173, 2204, 2244,
2342, 2409, 2460, 2833, 2861, 2929, 2939, 2941
\l_@@_final_j_int .....
1956, 2030, 2035, 2042, 2047, 2053, 2063,
2071, 2075, 2084, 2092, 2174, 2205, 2241,
2282, 2411, 2462, 2854, 2864, 2866, 2908, 2937
\l_@@_final_open_bool ..... 1958, 2036,
2040, 2043, 2050, 2056, 2060, 2076, 2239,
2280, 2290, 2301, 2327, 2340, 2348, 2369,
2407, 2458, 2594, 2609, 2640, 2641, 2831,
2855, 2867, 2906, 2930, 2942, 3003, 3532, 3571
\@@_find_extremities_of_line:nnnn ...
..... 2024, 2214, 2256, 2311, 2389, 2440
\l_@@_first_col_int ..... 109, 118,
253, 254, 451, 684, 708, 1224, 1291, 1565,
1581, 1926, 3048, 3099, 3131, 3160, 3206,
3646, 3656, 3690, 3738, 3777, 3978, 3984, 3990
\l_@@_first_row_int .....
..... 251, 252, 452, 688, 985,
1307, 1322, 1335, 1345, 1412, 1924, 3639,
3653, 3680, 3737, 3775, 3884, 3899, 3976, 4270
\@@_foldcase: ..... 1003, 1019, 1053
\c_@@_footnote_bool .....
..... 1065, 1392, 4094, 4120, 4143, 4162
\c_@@_footnotehyper_bool . 4093, 4121, 4153
\@@_full_name_env: .... 225, 4195, 4202,
4210, 4236, 4275, 4284, 4309, 4328, 4337, 4457
\@@_hdottedline: ..... 962, 3517
\@@_hdottedline:n ..... 3525, 3529
\@@_hdottedline_i: ..... 3520, 3522
\@@_hdottedline_i:n ..... 3534, 3538
\l_@@_hlines_bool .. 392, 463, 469, 870, 1969
\g_@@_ht_last_row_dim .....
..... 737, 940, 941, 1156, 1157, 1353
\g_@@_ht_row_one_dim .... 763, 764, 937, 938
\g_@@_ht_row_zero_dim .....
..... 757, 758, 935, 936, 1309, 1337, 1348
\l_@@_hvlines_bool ... 394, 467, 1966, 2167
\l_@@_hvlines_except_corners_bool ...
..... 395, 493, 1971, 3300, 3357
\@@_i: ..... 3639, 3641,
3642, 3643, 3644, 3653, 3659, 3661, 3662,
3663, 3664, 3669, 3670, 3671, 3672, 3680,
3683, 3685, 3686, 3687, 3739, 3741, 3744,
3745, 3749, 3750, 3775, 3780, 3782, 3784,
3788, 3789, 3800, 3807, 3809, 3811, 3815, 3816
\g_@@_iddots_int ..... 1940, 2468, 2469
\l_@@_in_env_bool .... 212, 294, 1076, 1077
\c_@@_in_preamble_bool . 20, 21, 22, 546, 562
\@@_info:n ..... 4156
\l_@@_initial_i_int ..... 1953,
2027, 2102, 2105, 2130, 2138, 2142, 2151,
2159, 2171, 2193, 2235, 2292, 2294, 2336,
2401, 2452, 2832, 2833, 2843, 2911, 2921, 2923
\l_@@_initial_j_int .....
..... 1954, 2028, 2103, 2110,
2115, 2121, 2131, 2139, 2143, 2152, 2160,
2172, 2194, 2232, 2274, 2350, 2352, 2357,
2403, 2454, 2836, 2846, 2848, 2907, 2908, 2919
\l_@@_initial_open_bool .....
..... 1957, 2104, 2108, 2111, 2118, 2124,
2128, 2144, 2230, 2272, 2289, 2299, 2327,
2334, 2346, 2399, 2450, 2592, 2639, 2830,
2837, 2849, 2905, 2912, 2924, 3002, 3531, 3570

```

\@@_instruction_of_type:nn	\@@_msg_new:nn
..... 812, 2682, 2697, 2712, 2733, 2755	16, 4125,
\l_@@_inter_dots_dim	4134, 4192, 4199, 4207, 4214, 4219, 4224,
. 377, 378, 1962, 2597, 2604, 2615, 2623,	4229, 4234, 4240, 4245, 4251, 4256, 4262,
2630, 2635, 2647, 2655, 3561, 3564, 3592, 3594	4267, 4274, 4276, 4282, 4289, 4296, 4301,
\g_@@_internal_code_after_tl	4306, 4313, 4319, 4325, 4333, 4335, 4341, 4563
.... 232, 1037, 1976, 1977, 3524, 3854, 4030	\@@_msg_new:nnn
\@@_j:	17, 4095, 4346, 4364, 4406, 4454, 4503, 4550
3646, 3648,	\@@_msg_redirect_name:nn
3649, 3650, 3651, 3656, 3659, 3661, 3664,	18, 633
3666, 3667, 3669, 3672, 3674, 3675, 3690,	\@@_multicolumn:nnn
3693, 3695, 3696, 3697, 3752, 3754, 3757,	966, 2770
3759, 3763, 3764, 3777, 3780, 3781, 3783,	\g_@@_multicolumn_cells_seq
3788, 3789, 3801, 3807, 3808, 3810, 3815, 3816 983, 2789, 3664, 3672, 3794
\l_@@_l_dim	\g_@@_multicolumn_sizes_seq
.... 2570, 2571, 2584, 2585, 2597, 2603,	984, 2791, 3795
2614, 2622, 2630, 2635, 2647, 2648, 2655, 2656	\g_@@_name_env_str
\l_@@_large_nodes_bool 402, 476, 1948, 1952	223,
\g_@@_last_col_found_bool	228, 229, 1070, 1071, 1558, 1786, 1787,
261,	1793, 1794, 1801, 1802, 1809, 1810, 1817,
1045, 1288, 1385, 1650, 1679, 1745, 1872, 1929	1818, 1825, 1826, 1835, 1863, 1995, 4000, 4176
\l_@@_last_col_int	\l_@@_name_str
.... 259, 260, 620, 655, 657, 675, 687,	400, 520, 743, 746,
701, 1109, 1181, 1187, 1194, 1236, 1848,	804, 807, 854, 857, 1165, 1174, 1177, 1183,
1850, 1873, 1876, 1928, 2316, 2355, 2679,	1192, 1195, 1574, 1575, 1596, 1597, 1614,
2694, 2730, 2752, 3980, 3986, 3992, 4178, 4196	1615, 1645, 1646, 1671, 1674, 1690, 1693,
\l_@@_last_col_without_value_bool ...	1883, 1887, 1904, 1908, 3785, 3788, 3812, 3815
..... 258, 654, 1874, 4181	\g_@@_names_seq
\l_@@_last_empty_column_int	211, 517, 519, 4561
..... 3457, 3470, 3476, 3489	\l_@@_nb_cols_int
\l_@@_last_empty_row_int . 3439, 3453, 3473 3967, 3972, 3975, 3979, 3985, 3991
\l_@@_last_row_int 255, 256, 453, 725, 874,	\l_@@_nb_rows_int
1031, 1107, 1152, 1162, 1169, 1176, 1276,	3966, 3971, 3982
1280, 1283, 1290, 1351, 1537, 1538, 1713,	\@@_newcolumnntype
1714, 1754, 1755, 1895, 2219, 2261, 2709,	880, 891, 992, 1007
2724, 2746, 2970, 3212, 3220, 3928, 3988, 4327	\@@_node_for_multicolumn:nn
\l_@@_last_row_without_value_bool ...	3796, 3803
..... 257, 1164, 1278, 1893	\@@_node_for_the_cell: 783, 788, 1732, 1778
\g_@@_last_vdotted_col_int	\@@_node_position: .. 1118, 1120, 1126, 1128
..... 1034, 1036, 1042, 1044	\l_@@_notes_above_space_dim
\l_@@_left_delim_dim	396, 397
..... 1200, 1204, 1209, 1502, 1734	\l_@@_notes_bottomrule_bool
\l_@@_left_delim_tl	534, 678, 697, 1456
1067, 3560	\l_@@_notes_code_after_tl
\l_@@_left_margin_dim	532, 1465
..... 403, 479, 1260, 1735, 3550, 3768	\l_@@_notes_code_before_tl
\l_@@_letter_for_dotted_lines_str ...	530, 1441
..... 638, 646, 647, 1024, 1025	\@@_notes_label_in_list:n 267, 282, 290, 542
\l_@@_light_syntax_bool	\@@_notes_label_in_tabular:n . 266, 303, 539
..... 386, 447, 1263, 1268, 1896	\l_@@_notes_para_bool .. 528, 676, 695, 1442
\@@_light_syntax_i	\@@_notes_style:n
1528, 1531 265, 268, 282, 290, 306, 311, 536
\@@_line	\l_@@_nullify_dots_bool
1990, 2986 398, 474, 2686, 2701, 2716, 2738, 2760
\@@_line_i:nn	\l_@@_number_of_notes_int 264, 297, 307, 317
2993, 3000	\@@_old_CT@arc@
\@@_line_with_light_syntax:n ... 1542, 1546	1078, 1997
\@@_line_with_light_syntax_i:n	\@@_old_arraycolsep_dim 245, 922, 1252
..... 1541, 1547, 1548	\@@_old_cdots
\@@_math_toggle_token:	948, 2701
133,	\@@_old_ddots
770, 1707, 1724, 1749, 1765, 3838, 4072, 4076	950, 2738
\g_@@_max_cell_width_dim	\l_@@_old_iRow_int
..... 776, 777, 1087, 1625, 3604, 3630	233, 908, 2021
\l_@@_max_delimiter_width_bool	\@@_old_ialign:
..... 410, 446, 1381	836, 944, 1974
\c_@@_max_l_dim	\@@_old_iddots
2584, 2589	951, 2760
\l_@@_medium_nodes_bool 401, 475, 1946, 3916	\l_@@_old_jCol_int
\@@_message_hdotsfor: 4187, 4195, 4202, 4210	234, 911, 2022
	\@@_old_ldots
	947, 2686
	\@@_old_multicolumn
	2769, 2776
	\@@_old_pgful@check@rerun
	74, 78
	\@@_old_vdots
	949, 2716
	\l_@@_parallelize_diags_bool
 390, 391, 471, 1937, 2415, 2466
	\@@_pgf_rect_node:nnn
	350, 3918
	\@@_pgf_rect_node:nnnnn
 325, 3779, 3806, 3879, 3913
	\c_@@_pgfortikzpicture_tl
 37, 41, 2003, 3019, 3533
	\@@_picture_position: 1112, 1120, 1128

<code>\g_@@_pos_of_blocks_seq</code>	249,	<code>\c_@@_siunitx_loaded_bool</code>	143, 147, 152, 170, 662
1090, 2792, 3274, 3296, 3353, 3503, 3833, 4040		<code>\l_@@_small_bool</code>	621,
<code>\g_@@_pos_of_xdots_seq</code>	250, 2169, 3275, 3298, 3355, 3403	665, 671, 689, 714, 914, 1708, 1750, 1959	
<code>@@_pre_array:</code>	902, 1199	<code>@@_standard_cline</code>	106, 955
<code>\c_@@_preamble_first_col_tl</code>	1225, 1701	<code>@@_standard_cline:w</code>	106, 107
<code>\c_@@_preamble_last_col_tl</code>	1237, 1741	<code>\l_@@_standard_cline_bool</code> ...	374, 439, 954
<code>@@_pred:n</code>	110, 132, 1850, 3306, 3319, 3363, 3376	<code>\c_@@_standard_tl</code> 384, 385, 2491, 3565, 3595	
<code>@@_put_box_in_flow:</code>	1383, 1394, 1504	<code>\l_@@_stop_loop_bool</code>	2031, 2032,
<code>@@_put_box_in_flow_bis:nn</code>	1382, 1471	2064, 2077, 2086, 2099, 2100, 2132, 2145, 2154	
<code>@@_put_box_in_flow_i:</code>	1400, 1402	<code>@@_succ:n</code>	127,
<code>@@_qpoint:n</code>	207, 1301, 1303, 1329, 1331, 1420, 1422,	131, 846, 852, 1422, 1663, 1669, 1674,	
1425, 2232, 2235, 2241, 2244, 2274, 2282,		1675, 1683, 1688, 1693, 1694, 1930, 2241,	
2292, 2294, 2336, 2342, 2350, 2352, 2401,		2282, 2294, 2342, 2352, 2409, 2411, 2454,	
2403, 2409, 2411, 2452, 2454, 2460, 2462,		2460, 3051, 3064, 3085, 3102, 3128, 3134,	
3027, 3030, 3047, 3051, 3064, 3066, 3083,		3168, 3170, 3235, 3239, 3259, 3264, 3290,	
3085, 3098, 3102, 3122, 3128, 3130, 3134,		3336, 3349, 3359, 3393, 3552, 3593, 3745,	
3157, 3159, 3168, 3170, 3233, 3235, 3241,		3749, 3759, 3763, 3873, 3875, 3910, 4058, 4060	
3261, 3264, 3332, 3334, 3336, 3389, 3391,		<code>\l_@@_suffix_tl</code>	3707, 3718,
3393, 3541, 3545, 3552, 3588, 3591, 3593,		3728, 3731, 3780, 3788, 3789, 3807, 3815, 3816	
3685, 3695, 3869, 3871, 3873, 3875, 3895,		<code>@@_tab_or_array_colsep:</code> ..	141, 2234, 2243
3910, 3931, 3932, 3938, 4053, 4055, 4058, 4060		<code>\c_@@_table_collect_begin_tl</code> .	160, 162, 180
<code>\l_@@_radius_dim</code>	381, 382, 1032,	<code>\c_@@_table_print_tl</code>	163, 164, 182
1961, 2248, 2249, 2664, 3519, 3543, 3589, 3590		<code>\g_@@_tabularnotes_seq</code>	263, 298, 1445, 1451, 1467
<code>\l_@@_real_left_delim_dim</code> 1473, 1488, 1503		<code>@@_test_if_cell_in_a_block:nn</code>	3442, 3460, 3478, 3498
<code>\l_@@_real_right_delim_dim</code> 1474, 1500, 1506		<code>@@_test_if_cell_in_block:nnnnnnn</code> ...	3504, 3506
<code>@@_rectanglecolor</code>	1141, 3145	<code>@@_test_if_hline_in_block:nnnn</code>	3297, 3299, 3405
<code>@@_renew_NC@rewrite@S:</code>	171, 173, 1043	<code>@@_test_if_math_mode:</code>	215, 1075, 1795, 1803, 1811, 1819, 1827
<code>\l_@@_renew_dots_bool</code>	472, 973, 4114	<code>@@_test_if_vline_in_block:nnnn</code>	3354, 3356, 3416
<code>@@_renew_matrix:</code>	623, 3946, 4116	<code>\l_@@_the_array_box</code>	1213, 1249, 1435, 1437, 1438
<code>@@_restore_iRow_jCol:</code>	1996, 2019	<code>\c_@@_tikz_loaded_bool</code>	25, 36, 1132, 1978, 3572
<code>\c_@@_revtex_bool</code>	45, 47, 50, 825	<code>\l_@@_tikz_tl</code>	3843, 3878
<code>\l_@@_right_delim_dim</code>	1201, 1205, 1211, 1505, 1775	<code>@@_true:c:</code>	181, 998, 1013, 1223
<code>\l_@@_right_delim_tl</code>	1068, 3563	<code>\l_@@_type_of_col_tl</code>	658, 659, 660, 661, 663, 1836, 1838
<code>\l_@@_right_margin_dim</code>	404, 481, 1272, 1776, 2359, 3557, 3771	<code>\c_@@_types_of_matrix_seq</code>	4169, 4176
<code>@@_rotate:</code>	968, 2959	<code>@@_update_for_first_and_last_row:</code> ..	751, 778, 1154, 1726, 1767
<code>@@_rotate_i:</code>	2963, 2965	<code>@@_use_array_box_with_notes:</code>	1312, 1316, 1340, 1370, 1432
<code>@@_rotate_ii:</code>	2962, 2965, 2966	<code>@@_vdottedline:n</code>	1038, 3568
<code>@@_rotate_iii:</code>	2966, 2967	<code>@@_vdottedline_i:n</code>	3575, 3580, 3584
<code>\g_@@_row_of_col_done_bool</code>	236, 867, 1069, 1580	<code>@@_vline:</code>	1088, 3222
<code>\g_@@_row_total_int</code>	986, 1289,	<code>\l_@@_vlines_bool</code>	393, 464, 468, 1231, 1243, 1255, 1274, 1970
1323, 1413, 1895, 1902, 1909, 1925, 2874,		<code>\g_@@_width_first_col_dim</code>	247, 1294, 1727, 1728
3639, 3653, 3680, 3775, 3861, 3884, 3899, 4271		<code>\g_@@_width_last_col_dim</code>	246, 1387, 1768, 1769
<code>@@_rowcolor</code>	1142, 3040, 3185, 3186	<code>\l_@@_x_final_dim</code>	240,
<code>@@_rowcolors</code>	1143, 3180	2185, 2242, 2243, 2283, 2284, 2332, 2354,	
<code>\g_@@_rows_seq</code> .	1534, 1536, 1538, 1540, 1542	2362, 2366, 2370, 2372, 2377, 2379, 2412,	
<code>\l_@@_rules_color_tl</code> ..	235, 432, 1095, 1096	2421, 2429, 2463, 2472, 2480, 2518, 2532,	
<code>@@_set_CT@arc@:</code>	135, 1096	2541, 2577, 2629, 2645, 3031, 3553, 3564, 3590	
<code>@@_set_CT@arc@_i:</code>	136, 137		
<code>@@_set_CT@arc@_ii:</code>	136, 139		
<code>@@_set_final_coords:</code>	2183, 2208		
<code>@@_set_final_coords_from_anchor:n</code> ..	2199, 2247, 2287, 2330, 2345, 2414, 2465		
<code>@@_set_initial_coords:</code>	2178, 2197		
<code>@@_set_initial_coords_from_anchor:n</code> ..	2188, 2238, 2279, 2329, 2339, 2406, 2457		
<code>@@_set_seq_of_str_from_clist:Nn</code> 4164, 4169			
<code>@@_set_size:n</code>	3964, 3973		

<code>\l_@@_x_initial_dim</code>	<code>\bigskip</code> 198
. 238, 2180, 2233, 2234, 2275, 2276,	<code>\Block</code> 967, 4304
2332, 2353, 2354, 2361, 2366, 2370, 2372,	<code>\BNiceMatrix</code> 3961
2374, 2377, 2379, 2404, 2421, 2429, 2455,	<code>\bNiceMatrix</code> 3958
2472, 2480, 2509, 2531, 2541, 2577, 2629,	bool commands:
2643, 2645, 2663, 2665, 3028, 3546, 3561, 3589	<code>\bool_do_until:Nn</code> 2032, 2100
<code>\l_@@_xdots_color_tl</code> 409, 422, 2223,	<code>\bool_gset_false:N</code> 786, 1045, 1069, 3414, 3425
2265, 2308, 2392, 2443, 2499, 2878, 2953, 2990	<code>\bool_gset_true:N</code> 1580, 1745,
<code>\l_@@_xdots_down_tl</code> . . . 426, 2515, 2525, 2560	2687, 2702, 2717, 2739, 2761, 2766, 3295, 3352
<code>\l_@@_xdots_line_style_tl</code>	<code>\bool_if:NTF</code> 134, 152, 550, 555, 566,
. 383, 385, 418, 2491, 2499, 3565, 3595	571, 711, 714, 841, 867, 870, 904, 914, 921,
<code>\l_@@_xdots_shorten_dim</code> 379,	973, 1065, 1076, 1086, 1097, 1132, 1146,
380, 424, 1963, 2506, 2507, 2603, 2614, 2622	1148, 1215, 1250, 1255, 1274, 1278, 1385,
<code>\l_@@_xdots_up_tl</code> 427, 2511, 2524, 2550	1392, 1456, 1569, 1583, 1601, 1619, 1632,
<code>\l_@@_y_final_dim</code> 241,	1658, 1679, 1681, 1708, 1750, 1872, 1874,
2186, 2245, 2249, 2296, 2300, 2302, 2343,	1893, 1896, 1915, 1937, 1952, 1959, 1969,
2410, 2423, 2426, 2461, 2474, 2477, 2518,	1970, 1971, 1978, 2167, 2299, 2301, 2415,
2532, 2540, 2579, 2634, 2653, 3032, 3544, 3594	2466, 2686, 2701, 2716, 2738, 2760, 2780,
<code>\l_@@_y_initial_dim</code>	3300, 3325, 3357, 3382, 3452, 3469, 3487,
239, 2181, 2236, 2248, 2295, 2296, 2300,	3613, 3623, 3824, 3916, 4020, 4143, 4153, 4181
2302, 2337, 2402, 2423, 2428, 2453, 2474,	<code>\bool_if:nTF</code> 170, 271,
2479, 2509, 2531, 2540, 2579, 2634, 2651,	294, 1288, 1410, 1946, 3004, 3443, 3461, 3479
2653, 2663, 2666, 3029, 3542, 3543, 3544, 3592	<code>\bool_lazy_all:nTF</code>
<code>\</code> 1525, 1547, 3980, 1227, 1239, 3407, 3418, 3508
3986, 3992, 4097, 4098, 4131, 4140, 4216,	<code>\bool_lazy_and:nnTF</code>
4221, 4226, 4231, 4237, 4258, 4264, 4271,	1622, 1709, 1927, 2288, 2523, 3123, 3153, 3273
4279, 4285, 4292, 4299, 4310, 4316, 4322,	<code>\bool_lazy_or:nnTF</code>
4334, 4338, 4343, 4348, 4349, 4367, 4368, 415, 1321, 1753, 2327, 2583, 3860
4409, 4410, 4457, 4458, 4506, 4507, 4556, 4557	<code>\bool_lazy_or_p:nn</code> 1712
<code>\{</code> . . 229, 1812, 4008, 4278, 4304, 4310, 4409, 4506	<code>\bool_not_p:n</code>
<code>\}</code> . . 229, 1812, 4008, 4278, 4304, 4310, 4409, 4506	1230, 1231, 1232, 1242, 1243, 1244, 1624, 1929
<code>\ </code> 1828, 4007	<code>\bool_set:Nn</code> 2331
	<code>\g_tmpa_bool</code> 3295, 3307, 3315, 3320,
	3325, 3352, 3364, 3372, 3377, 3382, 3414, 3425
<code>_</code> 4190, 4195, 4202, 4210, 4270,	<code>\l_tmpb_bool</code> . . . 3448, 3465, 3483, 3502, 3515
4271, 4275, 4304, 4308, 4321, 4328, 4329, 4337	box commands:
	<code>\box_clear_new:N</code> 906, 1213
A	<code>\box_dp:N</code> 736,
<code>\aboverulesep</code> 1460	756, 775, 934, 943, 1159, 1397, 1482, 1495
<code>\addtocounter</code> 315	<code>\box_ht:N</code> 737, 758, 764, 773,
<code>\alph</code> 265	936, 938, 941, 1157, 1396, 1482, 1495, 2975
<code>\array</code> 833	<code>\box_move_up:nn</code> . 57, 59, 61, 1312, 1340, 1429
<code>\arraycolsep</code> 142, 480, 482, 484,	<code>\box_rotate:Nn</code> 2969
917, 922, 1204, 1205, 1252, 1253, 1257,	<code>\box_set_dp:Nn</code> 774, 1397
1293, 1369, 1373, 1388, 2277, 2285, 3549, 3556	<code>\box_set_ht:Nn</code> 772, 1396
<code>\arrayrulecolor</code> 88	<code>\box_use:N</code> 318, 896, 899, 2976
<code>\arrayrulewidth</code> 112, 115, 125,	<code>\box_use_drop:N</code> 780, 784, 801, 1004, 1020,
434, 742, 845, 847, 853, 875, 1257, 1258,	1399, 1429, 1430, 1435, 1438, 1733, 3935, 3940
1274, 1306, 1334, 1364, 1376, 1587, 1589,	<code>\box_wd:N</code> 319, 777, 782, 1209, 1211,
1595, 1605, 1607, 1613, 1636, 1638, 1644,	1437, 1489, 1501, 1728, 1731, 1769, 1773, 4027
1662, 1664, 1670, 3049, 3050, 3052, 3065,	<code>\l_tmpa_box</code>
3067, 3084, 3086, 3100, 3101, 3103, 3127,	. . 301, 318, 319, 1208, 1209, 1210, 1211,
3129, 3132, 3133, 3135, 3158, 3161, 3162,	1357, 1396, 1397, 1399, 1429, 1430, 1482, 1495
3169, 3171, 3231, 3256, 3265, 3330, 3387, 3630	<code>\l_tmpb_box</code> 1475, 1489, 1490, 1501
<code>\arraystretch</code> 916	
<code>\AtBeginDocument</code>	C
. 22, 26, 66, 82, 144, 168, 269, 548,	<code>\Cdots</code> 958, 2692, 2695
564, 1999, 2670, 2808, 2884, 2982, 3015, 3527	<code>\cdots</code> 948, 976
<code>\AutoNiceMatrix</code> 4009	<code>\cellcolor</code> 1140
<code>\AutoNiceMatrixWithDelims</code> . . . 3969, 4001, 4013	<code>\chessboardcolors</code> 1145
	<code>\cline</code> 126, 955, 956
B	clist commands:
<code>\baselineskip</code> 91	<code>\clist_map_inline:nn</code> 3053, 3087, 3119
<code>\bgroup</code> 1066	<code>\CodeAfter</code> 971, 1528, 1531, 1991

`\hrule` 98, 112, 125, 875, 1461
`\hskip` 97
`\Hspace` 963
`\hspace` 2767
`\hss` 1017

I

`\ialign` 836, 919, 944, 1974
`\iddots` 961, 2745, 2746, 2751, 2752
`\iddots` 52, 951, 979
 if commands:
 `\if_mode_math:` 217
`\ifnum` 96
`\ifstandalone` 1082
 int commands:
 `\int_case:nNTF` 2721, 2727, 2743, 2749
 `\int_compare:nNnTF` 109, 110, 122,
 707, 708, 718, 725, 761, 872, 874, 1029,
 1031, 1034, 1107, 1109, 1152, 1162, 1181,
 1276, 1280, 1290, 1291, 1307, 1335, 1537,
 1565, 1751, 2042, 2049, 2053, 2055, 2110,
 2117, 2121, 2123, 2219, 2261, 2316, 2355,
 2357, 2787, 2874, 2949, 2970, 3062, 3096,
 3164, 3166, 3211, 3212, 3219, 3220, 3928,
 3976, 3978, 3980, 3984, 3986, 3988, 3990, 3992
 `\int_compare_p:n`
 3124, 3125, 3154, 3155, 3510, 3511, 3512, 3513
 `\int_gadd:Nn` 2800
 `\int_gincr:N`
 706, 734, 1085, 1656, 1746, 2417, 2468, 3610
 `\int_if_even:nTF` 3196
 `\int_incr:N` 297
 `\int_step_inline:nn` 3192, 3194
 `\int_step_inline:nnnn` 3440, 3458, 3473, 3476
 iow commands:
 `\iow_now:Nn` 69, 1917, 1918, 1920, 1935
 `\iow_shipout:Nn` 1877, 1878, 1885,
 1891, 1898, 1899, 1906, 1912, 3625, 3626, 3632
`\item` 1445, 1451

K

`\kern` 62
 keys commands:
 `\keys_define:nn` . 411, 430, 437, 491, 526,
 578, 616, 650, 669, 682, 693, 3599, 3841, 4110
 `\l_keys_key_tl`
 4097, 4248, 4343, 4348, 4366, 4408, 4456, 4505
 `\keys_set:nn` 445, 642,
 649, 1092, 1093, 1837, 1864, 2222, 2264,
 2319, 2391, 2442, 2877, 2952, 2989, 3612, 3851
 `\l_keys_value_tl` 4287, 4294, 4552

L

`\lasthline` 953
`\Ldots` 957, 2677, 2680
`\ldots` 947, 975
`\leaders` 112, 125
`\left` 1360, 1478, 1493
 legacy commands:
 `\legacy_if:nTF` 514
`\line` 1990, 4278

M

`\makebox` 1003, 1019

`\mathinner` 54
 mode commands:
 `\mode_leave_vertical:` 896, 1074
 msg commands:
 `\msg_error:nn` 12
 `\msg_error:nnn` 13
 `\msg_error:nnnn` 14, 3863
 `\msg_fatal:nn` 15
 `\msg_info:nn` 4146
 `\msg_new:nnn` 16
 `\msg_new:nnnn` 17
 `\msg_redirect_name:nnn` 19
`\multicolumn` 966, 2769, 2773, 2805, 2825
`\multispan` 110, 111, 123, 124
`\myfiledate` 6
`\myfileversion` 7

N

`\newcolumntype` 192, 193, 194, 1025
`\newcounter` 262
`\NewDocumentCommand` ... 273, 292, 648, 2674,
 2689, 2704, 2719, 2741, 2812, 2888, 2986,
 3040, 3076, 3112, 3145, 3180, 3190, 3969, 4009
`\NewDocumentEnvironment`
 1063, 1508, 1518, 1783,
 1791, 1799, 1807, 1815, 1823, 1833, 1861, 3608
`\NewExpandableDocumentCommand` 205, 3819
`\newlist` 277, 284
`\NiceArray` 1866
`\NiceArrayWithDelims`
 1788, 1796, 1804, 1812, 1820, 1828
`\NiceMatrixLastEnv` 205
`\NiceMatrixOptions` 648, 4367
`\NiceMatrixoptions` 4291
`\noalign` 91, 96, 115, 863, 927, 3519
`\nobreak` 287
`\normalbaselines` 913
`\nulldelimiterspace` 1489, 1501
`\numexpr` 131, 132

O

`\omit` 109, 1567, 1579, 1655
`\OnlyMainNiceMatrix` 969, 3203

P

`\par` 1447
 peek commands:
 `\peek_meaning:NnTF` 136, 299, 885
 `\peek_meaning_ignore_spaces:NnTF` 1510
 `\peek_meaning_remove_ignore_spaces:NnTF` 126
 `\peek_remove_spaces:n` 2785
`\pgfextracty` 3931
`\pgfgetlastxy` 361
`\pgfpathcircle` 2662
`\pgfpathlineto` 3244, 3266, 3339, 3396, 4062
`\pgfpathmoveto` 3243, 3263, 3338, 3395, 4057
`\pgfpathrectanglecorners` 3068, 3104, 3136, 3172
`\pgfpointhead` 359
`\pgfpointheadanchor` 208,
 2190, 2201, 3661, 3669, 3889, 3904, 3920, 3921
`\pgfpointdiff` 360, 1120, 1128
`\pgfpointlineatime` 2530
`\pgfpointorigin` 1573, 1689
`\pgfpointscale` 359
`\pgfpointshapeborder` 3027, 3030

<code>\vNiceMatrix</code>	3952	<code>\vtop</code>	843
<code>\vrule</code>	98		
<code>\vskip</code>	97		
		X	
		<code>\xglobal</code>	721, 728, 1718, 1758

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	3
5	The rules	5
5.1	The thickness and the color of the rules	5
5.2	A remark about <code>\cline</code>	5
5.3	The keys <code>hlines</code> and <code>vlines</code>	5
5.4	The key <code>hlines</code>	6
5.5	The key <code>hlines-except-corners</code>	6
5.6	The command <code>\diagbox</code>	7
5.7	Dotted rules	7
6	The color of the rows and columns	8
7	The width of the columns	10
8	The exterior rows and columns	11
9	The continuous dotted lines	13
9.1	The option <code>nullify-dots</code>	14
9.2	The command <code>\Hdotsfor</code>	14
9.3	How to generate the continuous dotted lines transparently	15
9.4	The labels of the dotted lines	15
9.5	Customization of the dotted lines	16
9.6	The dotted lines and the key <code>hlines</code>	17
10	The code-after	17
11	The notes in the tabulars	17
11.1	The footnotes	17
11.2	The notes of tabular	18
11.3	Customisation of the tabular notes	19
11.4	Utilisation of <code>{NiceTabular}</code> with <code>threeparttable</code>	21
12	Other features	21
12.1	Use of the column type <code>S</code> of <code>siunitx</code>	21
12.2	Alignment option in <code>{NiceMatrix}</code>	22
12.3	The command <code>\rotate</code>	22
12.4	The option <code>small</code>	22
12.5	The counters <code>iRow</code> and <code>jCol</code>	23
12.6	The option <code>light-syntax</code>	24
12.7	The environment <code>{NiceArrayWithDelims}</code>	24

13	Utilisation of Tikz with nicematrix	24
13.1	The nodes corresponding to the contents of the cells	24
13.2	The “medium nodes” and the “large nodes”	25
13.3	The “row-nodes” and the “col-nodes”	26
14	Technical remarks	27
14.1	Definition of new column types	27
14.2	Diagonal lines	28
14.3	The “empty” cells	28
14.4	The option exterior-arraycolsep	29
14.5	Incompatibilities	29
15	Examples	29
15.1	Notes in the tabulars	29
15.2	Dotted lines	30
15.3	Dotted lines which are no longer dotted	32
15.4	Width of the columns	32
15.5	How to highlight cells of the matrix	33
15.6	Direct use of the Tikz nodes	36
16	Implementation	37
17	History	143
	Index	147