

The package `nicematrix`*

F. Pantigny
fpantigny@wanadoo.fr

November 8, 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`, `array`, `amsmath`, `pgfcore` and the module `shapes` of PGF (`tikz`, which is a layer over PGF is *not* loaded). The final user only has to load the package with `\usepackage{nicematrix}`.

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

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

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

Important

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

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

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

*This document corresponds to the version 5.6 of `nicematrix`, at the date of 2020/11/08.

1 The environments of this package

The package `nicematrix` defines the following new environments.

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

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

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

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

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

Important

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

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

2 The vertical space between the rows

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

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

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

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

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

¹One should remark that these parameters apply also to the columns of type `S` of `siunitx` whereas the package `cellspace` is not able to act on such columns of type `S`.

3 The vertical position of the arrays

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

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

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

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

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

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

1. an item

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

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

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

1. an item

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

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

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

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

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

4 The blocks

4.1 General case

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

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

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

New 5.6 If this argument is empty, its default value is 1-1. If the number of rows is not specified, the block extends until the last row (idem for the columns).

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

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

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

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

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.²

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

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

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

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

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

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

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

²This argument between angular brackets may also be used to insert a command of font such as `\bfseries` when the command `\\` is used in the content of the block.

```
\begin{NiceTabular}{cwc{2cm}wc{3cm}c}
rose      & tulipe & marguerite & dahlia \\
violette  & \Block{2-2}{\LARGE\color{blue}De très jolies fleurs} & & souci \\
pervenche & & & lys \\
arum      & iris & jacinthe & muguet
\end{NiceTabular}
```

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

4.2 The mono-column blocks

New 5.4 The mono-column blocks have a special behaviour.

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

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

4.3 The mono-row blocks

New 5.6 For the mono-row blocks, the natural height and depth are taken into account for the height and depth of the current row (as does a standard `\multicolumn` of LaTeX).

4.4 The mono-cell blocks

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

There are two reasons to use a mono-column block:

- It's possible to use the command `\` in a (mono-cell) block.
- It's possible to use the option of horizontal alignment of the block in derogation of the type of column given in the preamble of the array.

```

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

```

Write	year of birth
Hugo	1802
Balzac	1799

We recall that if the first mandatory argument of `\Block` is left blank, the block is mono-cell.³

4.5 A small remark

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

```

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

```

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

5 The rules

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

5.1 Some differences with the classical environments

5.1.1 The vertical rules

In the environments of `nicematrix`, the vertical rules specified by `|` in the preambles of the environments are never broken, even by an incomplete row or by a double horizontal rule specified by `\hline\hline` (there is no need to use `hhline`).⁴

³One may consider that the default value of the first mandatory argument of `\Block` is 1-1

⁴This is the behaviour since the version 5.1 of `nicematrix`. Prior to that version, the behaviour was the standard behaviour of `array`.

```

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

```

First	Second
Peter	
Mary	George

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

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

```

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

```

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

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

```

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

```

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

```

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

```

5.1.2 The command `\cline`

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

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

```

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

```

A	B	C	D
A	B	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	B	C	D

5.2 The thickness and the color of the rules

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

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

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

```

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

```

rose	tulipe	lys
arum	iris	violette
muguet	dahlia	souci

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

5.3 The keys `hlines` and `vlines`

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

```

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

```

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

5.4 The key `hvlines`

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

```

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

```

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

5.5 The key `hvlines-except-corners`

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

```

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

```


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

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

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

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

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

5.6 The command `\diagbox`

The command `\diagbox` (inspired by the package `diagbox`), allows, when it is used in a cell, to slash that cell diagonally downwards.⁵

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

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

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

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}
```

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

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

⁵The author of this document considers that type of construction as graphically poor.

```

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

```

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

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

Remark : In the package `array` (on which the package `nicematrix` relies), horizontal and vertical rules make the array larger or wider by a quantity equal to the width of the rule⁶. In `nicematrix`, the dotted lines drawn by `\hdottedline` and “:” do likewise.

6 The color of the rows and columns

6.1 Use of `colortbl`

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

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

However, there is two drawbacks:

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

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

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

6.2 The tools of `nicematrix` in the code-before

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

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

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

⁶In fact, this is true only for `\hline` and “|” but not for `\cline`: cf p. 7

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

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

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

a	b	c
e	f	g
h	i	j

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

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

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

a	b	c
e	f	g
h	i	j

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

```
\begin{NiceArray}{l1l}[hvlines, code-before = \rowcolor{red!15}{1,3-5,8-}]
a_1 & b_1 & c_1 \\
a_2 & b_2 & c_2 \\
a_3 & b_3 & c_3 \\
a_4 & b_4 & c_4 \\
a_5 & b_5 & c_5 \\
a_6 & b_6 & c_6 \\
a_7 & b_7 & c_7 \\
a_8 & b_8 & c_8 \\
a_9 & b_9 & c_9 \\
a_{10} & b_{10} & c_{10} \\
\end{NiceArray}
```

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

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

- The command `\columncolor` takes its name from the command `\columncolor` of `colortbl`. Its syntax is similar to the syntax of `\rowcolor`.
- The command `\rowcolors` (with a s) takes its name from the command `\rowcolors` of `xcolor`⁷. 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.

⁷The command `\rowcolors` of `xcolor` is available when `xcolor` is loaded with the option `table`.

```

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

```

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

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

```

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

```

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

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

```

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

```

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

We have used the key **r** which aligns all the columns rightwards (cf. p. 26).

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

```

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

```

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

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

6.3 Color tools with the syntax of colortbl

It's possible to access the preceding tools with a syntax close to the syntax of `colortbl`. For that, one must use the key `colortbl-like` in the current environment.⁸

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

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

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

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

Each instruction `\cellcolor`, `\rowcolor` or `\columncolor` will generate an instruction `fill` (coded `f`) in the resulting PDF. In cases of juxtaposed colored rectangles, one may have a thin white color line in some PDF viewers⁹ (between the two first columns in the above example). In you want to avoid this problem, you should use the tools in the `code-before`. That's what we do with the following code.

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

⁸As of now, this key is not available in `\NiceMatrixOptions`.

⁹For example SumatraPDF, which uses MuPDF of Artifex Software, or PDF.js used by Firefox.

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

7 The width of the columns

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

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

Paris	New York	Madrid
Berlin	London	Roma
Rio	Tokyo	Oslo

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

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

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

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

It's possible to give the special value `auto` to the option `columns-width`: all the columns of the array will have a width equal to the widest cell of the array.¹⁰

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

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

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

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

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

But it's also possible to fix a zone where all the matrices will have their columns of the same width, equal to the widest cell of all the matrices. This construction uses the environment `{NiceMatrixBlock}` with the option `auto-columns-width`¹¹. The environment `{NiceMatrixBlock}` has no direct link with the command `\Block` presented previously in this document (cf. p. 4).

¹⁰The 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).

¹¹At this time, this is the only usage of the environment `{NiceMatrixBlock}` but it may have other usages in the future.

```

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

```

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

```

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

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

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. 28) 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}}
$\begin{pNiceArray}{cc|cc}[first-row,last-row=5,first-col,last-col,nullify-dots]
    & C_1 & & \Cdots & & C_4 & & \\
L_1 & & a_{11} & & a_{12} & & a_{13} & & a_{14} & & L_1 \\
\vdots & & a_{21} & & a_{22} & & a_{23} & & a_{24} & & \vdots \\
\hline
    & & a_{31} & & a_{32} & & a_{33} & & a_{34} & & \\
L_4 & & a_{41} & & a_{42} & & a_{43} & & a_{44} & & L_4 \\
    & & C_1 & & \Cdots & & C_4 & & \\
\end{pNiceArray}$
```

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

Remarks

- As shown in the previous example, the horizontal and vertical rules doesn't extend in the exterior rows and columns.
However, if one wishes to define new specifiers for columns in order to draw vertical rules (for example thicker than the standard rules), he should consider the command `\OnlyMainNiceMatrix` described on page 32.
- A specification of color present in `code-for-first-row` also applies to a dotted line draw in this exterior “first row” (excepted if a value has been given to `xdots/color`). Idem for the other exterior rows and columns.
- Logically, the potential option `columns-width` (described p. 14) 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).

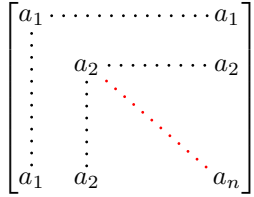
9 The continuous dotted lines

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`.¹²

¹²The 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`.

Each of them must be used alone in the cell of the array and it draws a dotted line between the first non-empty cells¹³ 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`.¹⁴

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



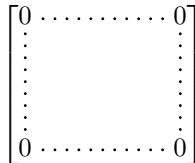
In order to represent the null matrix, one can use the following codage:

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



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

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



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

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

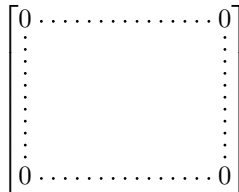
```
\begin{bNiceMatrix}
0      & \Cdots &      & 0      & \\
\Vdots &        &      & \Vdots & \\
          &        &      & \Vdots & \\
0      &        & \Cdots & 0      & \\
\end{bNiceMatrix}
```



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



¹³The precise definition of a “non-empty cell” is given below (cf. p. 33).

¹⁴It's also possible to change the color of all theses dotted lines with the option `xdots/color` (`xdots` to remind that it works for `\Cdots`, `\Ldots`, `\Vdots`, etc.): cf. p. 20.

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. 14

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.

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

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

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).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

9.3 How to generate the continuous dotted lines transparently

Imagine you have a document with a great number of mathematical matrices with ellipsis. You may wish to use the dotted lines of `nicematrix` without having to modify the code of each matrix. It's possible with the keys. `renew-dots` and `renew-matrix`.¹⁶

- The option `renew-dots`

With this option, the commands `\ldots`, `\cdots`, `\vdots`, `\ddots`, `\iddots`¹² and `\hdotsfor` are redefined within the environments provided by `nicematrix` and behave like `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots` and `\Hdotsfor`; the command `\dots` (“automatic dots” of `amsmath`) is also redefined to behave like `\Ldots`.

- The option `renew-matrix`

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

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

¹⁶The options `renew-dots`, `renew-matrix` can be fixed with the command `\NiceMatrixOptions` like the other options. However, they can also be fixed as options of the command `\usepackage`. There is also a key `transparent` which is an alias for the conjunction of `renew-dots` and `renew-matrix` but it must be considered as obsolete.

nicematrix.

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

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

9.4 The labels of the dotted lines

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

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

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

9.5 Customization of the dotted lines

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

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

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

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

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

The option `xdots/color`

The option `xdots/color` fixes the color of the dotted line. However, one should remark that the dotted lines drawn in the exterior rows and columns have a special treatment: cf. p. 15.

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

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.

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

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

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

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

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

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

9.6 The dotted lines and the rules

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

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

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

10 The code-after

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

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

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

¹⁸There is also a key `code-before` described p. 10.

cf. p. 38). **New 5.5** Before the version 5.5, it was necessary, in some circumstances, to put the keyword `\omit` before `\CodeAfter`. Since version 5.5, one must never put `\omit`.

11 The notes in the tabulars

11.1 The footnotes

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

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

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

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

11.2 The notes of tabular

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

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

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

^a Achard is an old family of the Poitou.

^b The name Lefebvre is an alteration of the name Lefebure.

- If you have several successive commands `\tabularnote{...}` *with no space at all between them*, the labels of the corresponding notes are composed together, separated by commas (this is similar to the option `multiple` of `footmisc` for the footnotes).
- If a command `\tabularnote{...}` is exactly at the end of a cell (with no space at all after), the label of the note is composed in an overlapping position (towards the right). This structure may provide a better alignment of the cells of a given column.
- If the key `notes/para` is used, the notes are composed at the end of the array in a single paragraph (as with the key `para` of `threeparttable`).
- **New 5.4** There is a key `tabularnote` which provides a way to insert some text in the zone of the notes before the numbered tabular notes.
- If the package `booktabs` has been loaded (before or after `nicematrix`), the key `notes/bottomrule` draws a `\bottomrule` of `booktabs` *after* the notes.
- The command `\tabularnote` may be used *before* the environment of `nicematrix`. Thus, it's possible to use it on the title inserted by `\caption` in an environment `{table}` of LaTeX.
- It's possible to create a reference to a tabular note created by `\tabularnote` (with the usual command `\label` used after the `\tabularnote`).

For an illustration of some of those remarks, see table 1, p. 24. This table has been composed with the following code.

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

11.3 Customisation of the tabular notes

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

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

Table 1: Use of `\tabularnotea`

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

Some text before the notes.

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

- `notes/enumitem-keys-para`
- `notes/code-before`

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

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

We detail these keys.

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

Initial value: `false`

That key is also available within a given environment.

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

Initial value: `false`

That key is also available within a given environment.

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

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

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

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

Initial value: `#1`

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

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

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

Initial value: `#1`

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

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

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

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

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

This initial value contains the specification `align = left` which requires a composition of the label leftwards in the box affected to that label. With that tuning, the notes are composed flush left, which is pleasant when composing tabulars in the spirit of `booktabs` (see for example the table 1, p. 24).

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

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

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

Initial value: `empty`

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

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

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

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

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

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

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

12 Other features

12.1 Use of the column type S of siunitx

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

```


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


```

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

12.2 Alignment option in `{NiceMatrix}`

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

```


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


```

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}

```

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

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

```

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

```

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

12.4 The option `small`

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

```

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

```

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

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 column¹⁹. Of course, the user must not change the value of these counters which are used internally by `nicematrix`.

In the `code-before` (cf. p. 10) and in the `code-after` (cf. p. 21), `iRow` represents the total number of rows (excepted the potential exterior rows) and `jCol` represents the total number of columns (excepted the potential exterior columns).

```

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

```

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

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

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

¹⁹We 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.

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

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

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

12.6 The option `light-syntax`

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

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

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

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

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

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.²⁰

12.7 The environment `{NiceArrayWithDelims}`

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

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

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

13 Use of Tikz with `nicematrix`

13.1 The nodes corresponding to the contents of the cells

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

²⁰The 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.

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

New 5.6 However, it’s possible to impose the creation of a node with the command `\NotEmpty`. ²¹

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

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

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

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

```


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


```

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

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

```


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


```

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

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

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

In fact, the package `nicematrix` can create “extra nodes”: the “medium nodes” and the “large nodes”. The first ones are created with the option `create-medium-nodes` and the second ones with the option `create-large-nodes`. ²²

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

²¹One should note that, with that command, the cell is considered as non-empty, which has consequences for the continuous dotted lines (cf. p. 16).

²²There is also an option `create-extra-nodes` which is an alias for the conjunction of `create-medium-nodes` and `create-large-nodes`.

The names of the “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.²³

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

The “large nodes” of the first column and last column may appear too small for some usage. That’s why it’s possible to use the options `left-margin` and `right-margin` to add space on both sides of the array and also space in the “large nodes” of the first column and last column. In the following example, we have used the options `left-margin` and `right-margin`.²⁴

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

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.

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

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

Here is an array composed with the following code:

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

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

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

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

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

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

²³There is no “large nodes” created in the exterior rows and columns (for these rows and columns, cf. p. 15).

²⁴The options `left-margin` and `right-margin` take dimensions as values but, if no value is given, the default value is used, which is `\arraycolsep` (by default: 5 pt). There is also an option `margin` to fix both `left-margin` and `right-margin` to the same value.

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

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

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

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

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

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

14 API for the developpers

The package `nicematrix` provides two variables which are internal but public²⁵:

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

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

²⁵According to the LaTeX3 conventions, each variable with name beginning with `\g_nicematrix` ou `\l_nicematrix` is public and each variable with name beginning with `\g__nicematrix` or `\l__nicematrix` is private.

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

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

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

\NewDocumentCommand \hatchcell { ! 0 { black } }
{
  \tl_gput_right:Nx \g_nicematrix_code_before_tl
  {
    \__pantigny_hatchcell:nnn
    { \int_use:c { c@iRow } }
    { \int_use:c { c@jCol } }
    { #1 }
  }
}
\ExplSyntaxOff
```

Here is an example of use:

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

Tokyo	Paris	London
Lima	Oslo	Miami
Los Angeles	Madrid	Roma

15 Technical remarks

15.1 Definition of new column types

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

For example, one may wish to define a new column type `?` in order to draw a (black) heavy rule of width 1 pt. The following definition will do the job²⁶:

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

The heavy vertical rule won't extend in the exterior rows.²⁷

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

²⁷Of course, such rule is defined by the classical technics of `nicematrix` and, for this reason, won't cross the double rules of `\hline\hline`.


```

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

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

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

15.2 Diagonal lines

By default, all the diagonal lines²⁸ of a same array are “parallelized”. That means that the first diagonal line is drawn and, then, the other lines are drawn parallel to the first one (by rotation around the left-most extremity of the line). That’s why the position of the instructions `\Ddots` in the array can have a marked effect on the final result.

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

Example with parallelization (default):

```

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

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

```

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

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

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

The same example without parallelization:

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

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

15.3 The “empty” cells

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

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

```

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

²⁸We speak of the lines created by `\Ddots` and not the lines created by a command `\line` in `code-after`.

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

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

15.4 The option `exterior-arraycolsep`

The environment `{array}` inserts an horizontal space equal to `\arraycolsep` before and after each column. In particular, there is a space equal to `\arraycolsep` before and after the array. This feature of the environment `{array}` was probably not a good idea²⁹. The environment `{matrix}` of `amsmath` and its variants (`{pmatrix}`, `{vmatrix}`, etc.) of `amsmath` prefer to delete these spaces with explicit instructions `\hskip -\arraycolsep`³⁰. The package `nicematrix` does the same in all its environments, `{NiceArray}` included. However, if the user wants the environment `{NiceArray}` behaving by default like the environment `{array}` of `array` (for example, when adapting an existing document) it's possible to control this behaviour with the option `exterior-arraycolsep`, set by the command `\NiceMatrixOptions`. With this option, exterior spaces of length `\arraycolsep` will be inserted in the environments `{NiceArray}` (the other environments of `nicematrix` are not affected).

15.5 Incompatibilities

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

16 Examples

16.1 Notes in the tabulars

The tools provided by `nicematrix` for the composition of the tabular notes have been presented in the section 11 p. 22.

Let's consider that we wish to number the notes of a tabular with stars.³¹

First, we write a command `\stars` similar the well-known commands `\arabic`, `\alph`, `\Alph`, etc. which produces a number of stars equal to its argument³²

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

²⁹In 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).*

³⁰And 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

³¹Of course, it's realistic only when there is very few notes in the tabular.

³²In fact: the value of its argument.

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

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

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

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

*Achard is an old family of the Poitou.

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

16.2 Dotted lines

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

```
\begin{pNiceMatrix}[xdots/shorten=0.6em]
0 & 1 & 0 & & & 0 \\
\vdots & & & \ddots & & \vdots \\
& & & \ddots & & \\
& & & \ddots & & 0 \\
0 & 0 & & & & 1 \\
1 & 0 & & \cdots & & 0 \\
\end{pNiceMatrix}
```

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

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 & \\
 $\end{pNiceMatrix}$ 

```

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

An example with `\multicolumn`:

```

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

```

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

An example with `\Hdotsfor`:

```

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

```

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

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 & a_1 & b_q & & \Vdots \\
& & \Vdots & & \Ddots & \\
& & a_p & & & b_q \\
\end{vNiceArray}

```

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

An example for a linear system:

```

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

```

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

16.3 Dotted lines which are no longer dotted

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

```

\NiceMatrixOptions
{nullify-dots,code-for-first-col = \color{blue},code-for-first-col=\color{blue}}
 $\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 0 \\ 0 & 1 & 0 & \cdots & 0 & L_2 \leftarrow L_2 - L_1 \\ 0 & 0 & 1 & \ddots & \vdots & L_3 \leftarrow L_3 - L_1 \\ \vdots & & & \ddots & \vdots & \vdots \\ 0 & \cdots & \cdots & 0 & 1 & L_n \leftarrow L_n - L_1 \end{pmatrix}$ 

```

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

16.4 Width of the columns

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

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

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

16.5 How to highlight cells of the matrix

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

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

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

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

```

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

```

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

We should remark that the rules we have drawn are drawn *after* the construction of the array and thus, they don't spread the cells of the array. We recall that, on the other side, the command `\hline`, the specifier “|” and the options `hlines`, `vlines` and `hvlines` spread the cells.³³

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

```

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

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

```

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

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`.

³³For the command `\cline`, see the remark p. 7.

```

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

```

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

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

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

```

We obtain the following matrix.

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

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

```

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

```

We obtain the following matrix.

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

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

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

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

16.6 Direct use of the Tikz nodes

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

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

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

\NiceMatrixOptions{nullify-dots}
```

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

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

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

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

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

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

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

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

\end{NiceMatrixBlock}

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

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

17 Implementation

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

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

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

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

Declaration of the package and packages loaded

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

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

<@@=nicematrix>

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

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

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

```

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

```

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

We load some packages.

```

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

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

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

```

Technical definitions

```

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

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

```

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.

```

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

```

```

44   }
45 }

```

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.

```

46 \bool_new:N \c_@@_revtex_bool
47 \ifclassloaded { revtex4-1 }
48   { \bool_set_true:N \c_@@_revtex_bool }
49   { }
50 \ifclassloaded { revtex4-2 }
51   { \bool_set_true:N \c_@@_revtex_bool }
52   { }

```

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

```

53 \ProvideDocumentCommand \iddots { }
54 {
55   \mathinner
56   {
57     \tex_mkern:D 1 mu
58     \box_move_up:nn { 1 pt } { \hbox:n { . } }
59     \tex_mkern:D 2 mu
60     \box_move_up:nn { 4 pt } { \hbox:n { . } }
61     \tex_mkern:D 2 mu
62     \box_move_up:nn { 7 pt }
63     { \vbox:n { \kern 7 pt \hbox:n { . } } }
64     \tex_mkern:D 1 mu
65   }
66 }

```

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.

```

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

```

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

```

76   \cs_set_protected:Npn \pgfutil@check@rerun ##1 ##2
77   {
78     \str_if_eq:eeF { nm- } { \tl_range:nnn { ##1 } 1 3 }
79     { \@@_old_pgful@check@rerun { ##1 } { ##2 } }
80   }
81 }

```

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

```

82 \bool_new:N \c_@@_colortbl_loaded_bool
83 \AtBeginDocument
84 {
85   \ifpackageloaded { colortbl }
86     { \bool_set_true:N \c_@@_colortbl_loaded_bool }
87     { }

```

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

```

88     \cs_set_protected:Npn \CT@arc@ { }
89     \cs_set:Npn \arrayrulecolor #1 # { \CT@arc@ { #1 } }
90     \cs_set:Npn \CT@arc@ #1 #2
91     {
92         \dim_compare:nNnT \baselineskip = \c_zero_dim \noalign
93         { \cs_gset:Npn \CT@arc@ { \color #1 { #2 } } }
94     }

```

Idem for `\CT@drs@`.

```

95     \cs_set_protected:Npn \CT@drsc@ { }
96     \cs_set:Npn \doublerulesepcolor #1 # { \CT@drs@ { #1 } }
97     \cs_set:Npn \CT@drs@ #1 #2
98     {
99         \dim_compare:nNnT \baselineskip = \c_zero_dim \noalign
100        { \cs_gset:Npn \CT@drsc@ { \color #1 { #2 } } }
101    }
102    \cs_set:Npn \hline
103    {
104        \noalign { \ifnum 0 = ` } \fi
105        \cs_set_eq:NN \hskip \vskip
106        \cs_set_eq:NN \vrule \hrule
107        \cs_set_eq:NN \@width \@height
108        { \CT@arc@ \vline }
109        \futurelet \reserved@a
110        \@xhline
111    }
112 }
113 }

```

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.

```

114 \cs_set:Npn \@@_standard_cline #1 { \@@_standard_cline:w #1 \q_stop }
115 \cs_set:Npn \@@_standard_cline:w #1-#2 \q_stop
116 {
117     \int_compare:nNnT \l_@@_first_col_int = 0 { \omit & }
118     \int_compare:nNnT { #1 } > 1 { \multispan { \@@_pred:n { #1 } } & }
119     \multispan { \int_eval:n { #2 - #1 + 1 } }
120     {
121         \CT@arc@
122         \leaders \hrule \@height \arrayrulewidth \hfill

```

The following `\skip_horizontal:N \c_zero_dim` is to prevent a potential `\unskip` to delete the `\leaders`³⁴

```

123     \skip_horizontal:N \c_zero_dim
124 }

```

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`.

```

125     \everycr { }
126     \cr
127     \noalign { \skip_vertical:N -\arrayrulewidth }
128 }

```

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

```

129 \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`.

```

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

```

³⁴See question 99041 on TeX StackExchange.

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 .

```

131 \cs_set:Npn \@@_cline_i:nn #1 #2 { \@@_cline_i:w #1-#2 \q_stop }
132 \cs_set:Npn \@@_cline_i:w #1-#2-#3 \q_stop
133 {

```

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).

```

134   \int_compare:nNnT { #1 } < { #2 }
135     { \multispan { \int_eval:n { #2 - #1 } } & }
136   \multispan { \int_eval:n { #3 - #2 + 1 } }
137   {
138     \CT@arc@
139     \leaders \hrule \@height \arrayrulewidth \hfill
140     \skip_horizontal:N \c_zero_dim
141   }

```

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

```

142   \peek_meaning_remove_ignore_spaces:NTF \cline
143     { & \@@_cline_i:en { \@@_succ:n { #3 } } }
144     { \everycr { } \cr }
145   }
146 \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.

```

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

```

The following command is a small shortcut.

```

149 \cs_new:Npn \@@_math_toggle_token:
150   { \bool_if:NF \l_@@_NiceTabular_bool \c_math_toggle_token }

151 \cs_new_protected:Npn \@@_set_CT@arc@:
152   { \peek_meaning:NTF [ \@@_set_CT@arc@_i: \@@_set_CT@arc@_ii: }
153 \cs_new_protected:Npn \@@_set_CT@arc@_i: [ #1 ] #2 \q_stop
154   { \cs_set:Npn \CT@arc@ { \color [ #1 ] { #2 } } }
155 \cs_new_protected:Npn \@@_set_CT@arc@_ii: #1 \q_stop
156   { \cs_set:Npn \CT@arc@ { \color { #1 } } }

```

The column S of siunitx

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

```

157 \bool_new:N \c_@@_siunitx_loaded_bool
158 \AtBeginDocument
159 {
160   \ifpackageloaded { siunitx }
161     { \bool_set_true:N \c_@@_siunitx_loaded_bool }
162     { }
163 }

```

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:`.

```
164 \cs_set_protected:Npn \@@_adapt_S_column:
165 {
166   \bool_if:NT \c_@@_siunitx_loaded_bool
167   {
168     \group_begin:
169     \@temptokena = { }
```

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

```
170   \cs_set_eq:NN \NC@find \prg_do_nothing:
171   \NC@rewrite@S { }
```

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

```
172   \tl_gset:NV \g_tmpa_tl \@temptokena
173   \group_end:
174   \tl_new:N \c_@@_table_collect_begin_tl
175   \tl_set:Nx \l_tmpa_tl { \tl_item:Nn \g_tmpa_tl 2 }
176   \tl_gset:Nx \c_@@_table_collect_begin_tl { \tl_item:Nn \l_tmpa_tl 1 }
177   \tl_new:N \c_@@_table_print_tl
178   \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).

```
179   \cs_gset_eq:NN \@@_adapt_S_column: \prg_do_nothing:
180 }
181 }
```

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

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

```

192         {
193             \tex_the:D \@temptokena
194             > { \c_@@_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.
195             \@@_true_c:
196             < { \c_@@_table_print_tl \@@_end_Cell: }
197         }
198     \NC@find
199 }
200 }
201 }
202 }

```

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

```

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

```

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

```

204 \cs_new_protected:Npn \@@_provide_pgfsyspdfmark:
205 {
206     \iow_now:Nn \@mainaux
207     {
208         \ExplSyntaxOn
209         \cs_if_free:NT \pgfsyspdfmark
210         { \cs_set_eq:NN \pgfsyspdfmark \@gobblethree }
211         \ExplSyntaxOff
212     }
213     \cs_gset_eq:NN \@@_provide_pgfsyspdfmark: \prg_do_nothing:
214 }

```

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.

```

215 \bool_new:N \c_@@_define_L_C_R_bool
216 \cs_new_protected:Npn \@@_define_L_C_R:
217 {
218     \newcolumntype L l
219     \newcolumntype C c
220     \newcolumntype R r
221 }

```

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

```

222 \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).

```

223 \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.

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

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

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

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

```
228 \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).

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

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

```
230 \tl_new:N \l_@@_cell_type_tl
231 \tl_set:Nn \l_@@_cell_type_tl { c }
```

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

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

Idem pour the blocks mono-row.

```
233 \dim_new:N \g_@@_blocks_ht_dim
234 \dim_new:N \g_@@_blocks_dp_dim
```

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

```
235 \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.

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

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

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

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

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

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

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

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

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

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

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

```
242 \cs_new_protected:Npn \@@_test_if_math_mode:
243 {
244   \if_mode_math: \else:
245     \@@_fatal:n { Outside-math-mode }
246   \fi:
247 }
```

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

```
248 \colorlet { nicematrix-last-col } { . }
249 \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*).

```
250 \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*.

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

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

```
252 \cs_new:Npn \@@_full_name_env:
253 {
254   \str_if_eq:VnTF \g_@@_com_or_env_str { command }
255     { command \space \c_backslash_str \g_@@_name_env_str }
256     { environment \space \{ \g_@@_name_env_str \} }
257 }
```

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

```
258 \tl_new:N \g_nicematrix_code_after_tl
```

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

```
259 \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.

```
260 \int_new:N \l_@@_old_iRow_int
261 \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.

```
262 \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).

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

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

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

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

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

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

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

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

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

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

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

```
270 \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`).

```
271 \dim_zero_new:N \l_tmpc_dim
```

```
272 \dim_zero_new:N \l_tmpd_dim
```

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

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

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

```
274 \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”.

```
275 \dim_new:N \g_@@_width_last_col_dim
```

```
276 \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.

```
277 \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}`.

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

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

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

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

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

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

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

Used for the color of the blocks.

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

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

```
282 \tl_new:N \l_@@_pos_of_block_tl
283 \tl_set:Nn \l_@@_pos_of_block_tl { c }
```

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

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

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

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

Variables for the exterior rows and columns

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

• First row

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

```
286 \int_new:N \l_@@_first_row_int
287 \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.

```
288 \int_new:N \l_@@_first_col_int
289 \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).

```
290 \int_new:N \l_@@_last_row_int
291 \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”.³⁵

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

Idem for `\l_@@_last_col_without_value_bool`

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

• Last column

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

```
294 \int_new:N \l_@@_last_col_int
```

```
295 \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.

```
296 \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.

```
297 \newcounter { tablarnote }
```

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

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

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

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

³⁵We can’t use `\l_@@_last_row_int` for this usage because, if `nicematrix` has read its value from the `aux` file, the value of the counter won’t be `-1` any longer.

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

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

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

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

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

```
302 \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`.

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

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

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

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

```
305 \AtBeginDocument
306 {
307   \bool_if:nTF { ! \c_@@_enumitem_loaded_bool }
308   {
309     \NewDocumentCommand \tabularnote { m }
310     { \@@_error:n { enumitem-not-loaded } }
311   }
312 }
```

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.

```
313 \newlist { tabularnotes } { enumerate } { 1 }
314 \setlist [ tabularnotes ]
315 {
316   topsep = 0pt ,
317   noitemsep ,
318   leftmargin = * ,
319   align = left ,
320   labelsep = 0pt ,
321   label =
322     \@@_notes_label_in_list:n { \@@_notes_style:n { tabularnotesi } } ,
323 }
324 \newlist { tabularnotes* } { enumerate* } { 1 }
325 \setlist [ tabularnotes* ]
326 {
327   afterlabel = \nobreak ,
328   itemjoin = \quad ,
329   label =
330     \@@_notes_label_in_list:n { \@@_notes_style:n { tabularnotes*i } }
331 }
```

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.³⁶

```

332 \NewDocumentCommand \tabularnote { m }
333 {
334   \bool_if:nTF { ! \l_@@_NiceArray_bool && \l_@@_in_env_bool }
335     { \@@_error:n { tabularnote~forbidden } }
336     {

```

`\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).

```

337   \int_incr:N \l_@@_number_of_notes_int

```

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

```

338   \seq_gput_right:Nn \g_@@_tabularnotes_seq { #1 }
339   \peek_meaning:NF \tabularnote
340   {

```

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.

```

341   \hbox_set:Nn \l_tmpa_box
342   {

```

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

```

343   \@@_notes_label_in_tabular:n
344   {
345     \stepcounter { tabularnote }
346     \@@_notes_style:n { tabularnote }
347     \prg_replicate:nn { \l_@@_number_of_notes_int - 1 }
348     {
349       ,
350       \stepcounter { tabularnote }
351       \@@_notes_style:n { tabularnote }
352     }
353   }
354 }

```

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.

```

355   \addtocounter { tabularnote } { -1 }
356   \refstepcounter { tabularnote }
357   \int_zero:N \l_@@_number_of_notes_int
358   \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).

```

359   \skip_horizontal:n { \box_wd:N \l_tmpa_box }
360   }
361 }
362 }
363 }
364 }

```

³⁶We should try to find a solution to that problem.

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.

```

365 \cs_new_protected:Npn \@@_pgf_rect_node:nnnnn #1 #2 #3 #4 #5
366 {
367   \begin { pgfscope }
368   \pgfset
369   {
370     outer~sep = \c_zero_dim ,
371     inner~sep = \c_zero_dim ,
372     minimum~size = \c_zero_dim
373   }
374   \pgftransformshift { \pgfpoint { 0.5 * ( #2 + #4 ) } { 0.5 * ( #3 + #5 ) } }
375   \pgfnode
376   { rectangle }
377   { center }
378   {
379     \vbox_to_ht:nn
380     { \dim_abs:n { #5 - #3 } }
381     {
382       \vfill
383       \hbox_to_wd:nn { \dim_abs:n { #4 - #2 } } { }
384     }
385   }
386   { #1 }
387   { }
388   \end { pgfscope }
389 }
```

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.

```

390 \cs_new_protected:Npn \@@_pgf_rect_node:nnn #1 #2 #3
391 {
392   \begin { pgfscope }
393   \pgfset
394   {
395     outer~sep = \c_zero_dim ,
396     inner~sep = \c_zero_dim ,
397     minimum~size = \c_zero_dim
398   }
399   \pgftransformshift { \pgfpoint scale { 0.5 } { \pgfpointadd { #2 } { #3 } } }
400   \pgfpointdiff { #3 } { #2 }
401   \pgfgetlastxy \l_tmpa_dim \l_tmpb_dim
402   \pgfnode
403   { rectangle }
404   { center }
405   {
406     \vbox_to_ht:nn
407     { \dim_abs:n \l_tmpb_dim }
408     { \vfill \hbox_to_wd:nn { \dim_abs:n \l_tmpa_dim } { } }
409   }
410   { #1 }
411   { }
412   \end { pgfscope }
413 }
```


The options

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

```
414 \bool_new:N \l_@@_colortbl_like_bool
```

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

```
415 \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`).

```
416 \dim_new:N \l_@@_cell_space_top_limit_dim
417 \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.

```
418 \dim_new:N \l_@@_inter_dots_dim
419 \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).

```
420 \dim_new:N \l_@@_xdots_shorten_dim
421 \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.

```
422 \dim_new:N \l_@@_radius_dim
423 \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.

```
424 \tl_new:N \l_@@_xdots_line_style_tl
425 \tl_const:Nn \c_@@_standard_tl { standard }
426 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
```

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

```
427 \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).

```
428 \str_new:N \l_@@_baseline_str
429 \tl_set:Nn \l_@@_baseline_str c
```

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

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

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

```
431 \bool_new:N \l_@@_parallelize_diags_bool
432 \bool_set_true:N \l_@@_parallelize_diags_bool
```

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

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

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

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

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

```
435 \clist_new:N \l_@@_except_corners_clist
436 \dim_new:N \l_@@_notes_above_space_dim
437 \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.).

```
438 \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).

```
439 \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.

```
440 \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”.

```
441 \bool_new:N \l_@@_medium_nodes_bool
442 \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`).

```
443 \dim_new:N \l_@@_left_margin_dim
444 \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`.

```
445 \dim_new:N \l_@@_extra_left_margin_dim
446 \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.

```
447 \tl_new:N \l_@@_end_of_row_tl
448 \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 “:”.

```
449 \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 function 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.

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

```
451 \keys_define:nn { NiceMatrix / xdots }
452 {
453   line-style .code:n =
454   {
455     \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.

```
456     { \cs_if_exist_p:N \tikzpicture }
457     { \str_if_eq_p:nn { #1 } { standard } }
458     { \tl_set:Nn \l_@@_xdots_line_style_tl { #1 } }
459     { \@@_error:n { bad-option-for-line-style } }
460   } ,
461   line-style .value_required:n = true ,
462   color .tl_set:N = \l_@@_xdots_color_tl ,
463   color .value_required:n = true ,
464   shorten .dim_set:N = \l_@@_xdots_shorten_dim ,
465   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 `_`.

```
466   down .tl_set:N = \l_@@_xdots_down_tl ,
467   up .tl_set:N = \l_@@_xdots_up_tl ,
```

The key `draw-first`, which is meant to be used only with `\Ddots` and `\Iddots`, which be caught when `\Ddots` or `\Iddots` is used (during the construction of the array and not when we draw the dotted lines).

```
468   draw-first .code:n = \prg_do_nothing: ,
469   unknown .code:n = \@@_error:n { Unknown-option-for-~xdots }
470 }
```

```
471 \keys_define:nn { NiceMatrix / rules }
472 {
473   color .tl_set:N = \l_@@_rules_color_tl ,
474   color .value_required:n = true ,
475   width .dim_set:N = \arrayrulewidth ,
476   width .value_required:n = true
477 }
```

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

```
478 \keys_define:nn { NiceMatrix / Global }
479 {
480   standard-cline .bool_set:N = \l_@@_standard_cline_bool ,
481   standard-cline .default:n = true ,
482   cell-space-top-limit .dim_set:N = \l_@@_cell_space_top_limit_dim ,
483   cell-space-top-limit .value_required:n = true ,
484   cell-space-bottom-limit .dim_set:N = \l_@@_cell_space_bottom_limit_dim ,
```

```

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

```

With the option `renew-dots`, the commands `\cdots`, `\ldots`, `\vdots`, `\ddots`, etc. are redefined and behave like the commands `\Cdots`, `\Ldots`, `\Vdots`, `\Ddots`, etc.

```

512 renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
513 renew-dots .value_forbidden:n = true ,
514 nullify-dots .bool_set:N = \l_@@_nullify_dots_bool ,
515 create-medium-nodes .bool_set:N = \l_@@_medium_nodes_bool ,
516 create-large-nodes .bool_set:N = \l_@@_large_nodes_bool ,
517 create-extra-nodes .meta:n =
518 { create-medium-nodes , create-large-nodes } ,
519 left-margin .dim_set:N = \l_@@_left_margin_dim ,
520 left-margin .default:n = \arraycolsep ,
521 right-margin .dim_set:N = \l_@@_right_margin_dim ,
522 right-margin .default:n = \arraycolsep ,
523 margin .meta:n = { left-margin = #1 , right-margin = #1 } ,
524 margin .default:n = \arraycolsep ,
525 extra-left-margin .dim_set:N = \l_@@_extra_left_margin_dim ,
526 extra-right-margin .dim_set:N = \l_@@_extra_right_margin_dim ,
527 extra-margin .meta:n =
528 { extra-left-margin = #1 , extra-right-margin = #1 } ,
529 extra-margin .value_required:n = true ,
530 }

```

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

```

531 \keys_define:nn { NiceMatrix / Env }
532 {
533   except-corners .clist_set:N = \l_@@_except_corners_clist ,
534   except-corners .default:n = { NW , SW , NE , SE } ,
535   hvlines-except-corners .code:n =
536   {
537     \clist_set:Nn \l_@@_except_corners_clist { #1 }
538     \bool_set_true:N \l_@@_vlines_bool
539     \bool_set_true:N \l_@@_hlines_bool
540   } ,

```

```

541 hvlines-except-corners .default:n = { NW , SW , NE , SE } ,
542 code-before .code:n =
543 {
544   \tl_if_empty:nF { #1 }
545   {
546     \tl_put_right:Nn \l_@@_code_before_tl { #1 }
547     \bool_set_true:N \l_@@_code_before_bool
548   }
549 } ,

```

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

```

550 c .code:n = \tl_set:Nn \l_@@_baseline_str c ,
551 t .code:n = \tl_set:Nn \l_@@_baseline_str t ,
552 b .code:n = \tl_set:Nn \l_@@_baseline_str b ,
553 baseline .tl_set:N = \l_@@_baseline_str ,
554 baseline .value_required:n = true ,
555 columns-width .code:n =
556   \tl_if_eq:nnTF { #1 } { auto }
557   { \bool_set_true:N \l_@@_auto_columns_width_bool }
558   { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,
559 columns-width .value_required:n = true ,
560 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.

```

561 \legacy_if:nF { measuring@ }
562 {
563   \str_set:Nn \l_tmpa_str { #1 }
564   \seq_if_in:NVTF \g_@@_names_seq \l_tmpa_str
565   { \@@_error:nn { Duplicate-name } { #1 } }
566   { \seq_gput_left:NV \g_@@_names_seq \l_tmpa_str }
567   \str_set_eq:NN \l_@@_name_str \l_tmpa_str
568 } ,
569 name .value_required:n = true ,
570 code-after .tl_gset:N = \g_nicematrix_code_after_tl ,
571 code-after .value_required:n = true ,
572 colortbl-like .code:n =
573   \bool_set_true:N \l_@@_colortbl_like_bool
574   \bool_set_true:N \l_@@_code_before_bool ,
575 colortbl-like .value_forbidden:n = true
576 }
577 \keys_define:nn { NiceMatrix / notes }
578 {
579   para .bool_set:N = \l_@@_notes_para_bool ,
580   para .default:n = true ,
581   code-before .tl_set:N = \l_@@_notes_code_before_tl ,
582   code-before .value_required:n = true ,
583   code-after .tl_set:N = \l_@@_notes_code_after_tl ,
584   code-after .value_required:n = true ,
585   bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
586   bottomrule .default:n = true ,
587   style .code:n = \cs_set:Nn \@@_notes_style:n { #1 } ,
588   style .value_required:n = true ,
589   label-in-tabular .code:n =
590     \cs_set:Nn \@@_notes_label_in_tabular:n { #1 } ,
591   label-in-tabular .value_required:n = true ,
592   label-in-list .code:n =
593     \cs_set:Nn \@@_notes_label_in_list:n { #1 } ,
594   label-in-list .value_required:n = true ,
595   enumitem-keys .code:n =
596     {
597       \bool_if:NTF \c_@@_in_preamble_bool

```

```

598     {
599         \AtBeginDocument
600         {
601             \bool_if:NT \c_@@_enumitem_loaded_bool
602             { \setlist* [ tabularnotes ] { #1 } }
603         }
604     }
605     {
606         \bool_if:NT \c_@@_enumitem_loaded_bool
607         { \setlist* [ tabularnotes ] { #1 } }
608     }
609 },
610 enumitem-keys .value_required:n = true ,
611 enumitem-keys-para .code:n =
612 {
613     \bool_if:NTF \c_@@_in_preamble_bool
614     {
615         \AtBeginDocument
616         {
617             \bool_if:NT \c_@@_enumitem_loaded_bool
618             { \setlist* [ tabularnotes* ] { #1 } }
619         }
620     }
621     {
622         \bool_if:NT \c_@@_enumitem_loaded_bool
623         { \setlist* [ tabularnotes* ] { #1 } }
624     }
625 },
626 enumitem-keys-para .value_required:n = true ,
627 unknown .code:n = \@@_error:n { Unknown-key-for-notes }
628 }

```

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

```

629 \keys_define:nn { NiceMatrix }
630 {
631     NiceMatrixOptions .inherit:n =
632     { NiceMatrix / Global } ,
633     NiceMatrixOptions / xdots .inherit:n = NiceMatrix / xdots ,
634     NiceMatrixOptions / rules .inherit:n = NiceMatrix / rules ,
635     NiceMatrixOptions / notes .inherit:n = NiceMatrix / notes ,
636     NiceMatrix .inherit:n =
637     {
638         NiceMatrix / Global ,
639         NiceMatrix / Env ,
640     } ,
641     NiceMatrix / xdots .inherit:n = NiceMatrix / xdots ,
642     NiceMatrix / rules .inherit:n = NiceMatrix / rules ,
643     NiceTabular .inherit:n =
644     {
645         NiceMatrix / Global ,
646         NiceMatrix / Env
647     } ,
648     NiceTabular / xdots .inherit:n = NiceMatrix / xdots ,
649     NiceTabular / rules .inherit:n = NiceMatrix / rules ,
650     NiceArray .inherit:n =
651     {
652         NiceMatrix / Global ,
653         NiceMatrix / Env ,
654     } ,
655     NiceArray / xdots .inherit:n = NiceMatrix / xdots ,
656     NiceArray / rules .inherit:n = NiceMatrix / rules ,
657     pNiceArray .inherit:n =

```

```

658     {
659         NiceMatrix / Global ,
660         NiceMatrix / Env ,
661     } ,
662     pNiceArray / xdots .inherit:n = NiceMatrix / xdots ,
663     pNiceArray / rules .inherit:n = NiceMatrix / rules ,
664 }

```

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

```

665 \keys_define:nn { NiceMatrix / NiceMatrixOptions }
666 {
667     last-col .code:n = \tl_if_empty:nF { #1 }
668                 { \@@_error:n { last-col~non-empty~for~NiceMatrixOptions } }
669                 \int_zero:N \l_@@_last_col_int ,
670     small .bool_set:N = \l_@@_small_bool ,
671     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.

```

672     renew-matrix .code:n = \@@_renew_matrix: ,
673     renew-matrix .value_forbidden:n = true ,

```

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

```

674     transparent .meta:n = { renew-dots , renew-matrix } ,
675     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}`.

```

676     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.

```

677     columns-width .code:n =
678         \tl_if_eq:nnTF { #1 } { auto }
679         { \@@_error:n { Option~auto~for~columns-width } }
680         { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,

```

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

```

681     allow-duplicate-names .code:n =
682         \@@_msg_redirect_name:nn { Duplicate-name } { none } ,
683     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`).

```

684     letter-for-dotted-lines .code:n =
685     {
686         \tl_if_single_token:nTF { #1 }
687         { \str_set:Nx \l_@@_letter_for_dotted_lines_str { #1 } }
688         { \@@_error:n { Bad-value-for-letter-for-dotted-lines } }
689     } ,
690     letter-for-dotted-lines .value_required:n = true ,
691     notes .code:n = \keys_set:nn { NiceMatrix / notes } { #1 } ,
692     notes .value_required:n = true ,
693     unknown .code:n = \@@_error:n { Unknown~key~for~NiceMatrixOptions }
694 }

```

```

695 \str_new:N \l_@@_letter_for_dotted_lines_str
696 \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.

```

697 \NewDocumentCommand \NiceMatrixOptions { m }
698 { \keys_set:nn { NiceMatrix / NiceMatrixOptions } { #1 } }

```

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

```

699 \keys_define:nn { NiceMatrix / NiceMatrix }
700 {
701   last-col .code:n = \tl_if_empty:nTF {#1}
702     {
703       \bool_set_true:N \l_@@_last_col_without_value_bool
704       \int_set:Nn \l_@@_last_col_int { -1 }
705     }
706     { \int_set:Nn \l_@@_last_col_int { #1 } } ,
707   l .code:n = \tl_set:Nn \l_@@_type_of_col_tl l ,
708   r .code:n = \tl_set:Nn \l_@@_type_of_col_tl r ,
709   small .bool_set:N = \l_@@_small_bool ,
710   small .value_forbidden:n = true ,
711   unknown .code:n = \@@_error:n { Unknown-option-for-NiceMatrix }
712 }

```

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

```

713 \keys_define:nn { NiceMatrix / NiceArray }
714 {

```

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.

```

715   small .bool_set:N = \l_@@_small_bool ,
716   small .value_forbidden:n = true ,
717   last-col .code:n = \tl_if_empty:nF { #1 }
718     { \@@_error:n { last-col-non-empty-for-NiceArray } }
719     \int_zero:N \l_@@_last_col_int ,
720   notes / para .bool_set:N = \l_@@_notes_para_bool ,
721   notes / para .default:n = true ,
722   notes / bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
723   notes / bottomrule .default:n = true ,
724   tabularnote .tl_set:N = \l_@@_tabularnote_tl ,
725   tabularnote .value_required:n = true ,
726   unknown .code:n = \@@_error:n { Unknown-option-for-NiceArray }
727 }
728 \keys_define:nn { NiceMatrix / pNiceArray }
729 {
730   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
731   last-col .code:n = \tl_if_empty:nF {#1}
732     { \@@_error:n { last-col-non-empty-for-NiceArray } }
733     \int_zero:N \l_@@_last_col_int ,
734   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
735   small .bool_set:N = \l_@@_small_bool ,
736   small .value_forbidden:n = true ,
737   unknown .code:n = \@@_error:n { Unknown-option-for-NiceMatrix }
738 }

```

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


```

739 \keys_define:nn { NiceMatrix / NiceTabular }
740 {
741   notes / para .bool_set:N = \l_@@_notes_para_bool ,
742   notes / para .default:n = true ,
743   notes / bottomrule .bool_set:N = \l_@@_notes_bottomrule_bool ,
744   notes / bottomrule .default:n = true ,
745   tabularnote .tl_set:N = \l_@@_tabularnote_tl ,
746   tabularnote .value_required:n = true ,
747   last-col .code:n = \tl_if_empty:nF {#1}
748     { \@@_error:n { last-col~non~empty~for~NiceArray } }
749     \int_zero:N \l_@@_last_col_int ,
750   unknown .code:n = \@@_error:n { Unknown~option~for~NiceTabular }
751 }

```

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}`).

```

752 \cs_new_protected:Npn \@@_Cell:
753 {

```

At the beginning of the cell, we link `\CodeAfter` to a command which do *not* begin with `\omit` (whereas the standard version of `\CodeAfter` begins with `\omit`).

```

754   \cs_set_eq:NN \CodeAfter \@@_CodeAfter_i:n

```

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

```

755   \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.

```

756   \int_compare:nNnT \c@jCol = 1
757     { \int_compare:nNnT \l_@@_first_col_int = 1 \@@_begin_of_row: }
758   \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).

```

759   \hbox_set:Nw \l_@@_cell_box
760   \bool_if:NF \l_@@_NiceTabular_bool
761   {
762     \c_math_toggle_token
763     \bool_if:NT \l_@@_small_bool \scriptstyle
764   }

```

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.

```

765   \int_compare:nNnTF \c@iRow = 0
766   {
767     \int_compare:nNnT \c@jCol > 0
768     {
769       \l_@@_code_for_first_row_tl
770       \xglobal \colorlet { nicematrix-first-row } { . }
771     }
772   }
773   {
774     \int_compare:nNnT \c@iRow = \l_@@_last_row_int
775     {
776       \l_@@_code_for_last_row_tl
777       \xglobal \colorlet { nicematrix-last-row } { . }

```

```

778     }
779 }
780 }

```

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.

```

781 \cs_new_protected:Npn \@@_begin_of_row:
782 {
783   \int_gincr:N \c@iRow
784   \dim_gset_eq:NN \g_@@_dp_ante_last_row_dim \g_@@_dp_last_row_dim
785   \dim_gset:Nn \g_@@_dp_last_row_dim { \box_dp:N \@arstrutbox }
786   \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
787   \pgfpicture
788   \pgfrememberpicturepositiononpagetrue
789   \pgfcoordinate
790     { \@@_env: - row - \int_use:N \c@iRow - base }
791     { \pgfpoint \c_zero_dim { 0.5 \arrayrulewidth } }
792   \str_if_empty:NF \l_@@_name_str
793   {
794     \pgfnodealias
795       { \l_@@_name_str - row - \int_use:N \c@iRow - base }
796       { \@@_env: - row - \int_use:N \c@iRow - base }
797   }
798   \endpgfpicture
799 }

```

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.

```

800 \cs_new_protected:Npn \@@_update_for_first_and_last_row:
801 {
802   \int_compare:nNnTF \c@iRow = 0
803   {
804     \dim_gset:Nn \g_@@_dp_row_zero_dim
805       { \dim_max:nn \g_@@_dp_row_zero_dim { \box_dp:N \l_@@_cell_box } }
806     \dim_gset:Nn \g_@@_ht_row_zero_dim
807       { \dim_max:nn \g_@@_ht_row_zero_dim { \box_ht:N \l_@@_cell_box } }
808   }
809   {
810     \int_compare:nNnT \c@iRow = 1
811     {
812       \dim_gset:Nn \g_@@_ht_row_one_dim
813         { \dim_max:nn \g_@@_ht_row_one_dim { \box_ht:N \l_@@_cell_box } }
814     }
815   }
816 }
817 \cs_new_protected:Npn \@@_rotate_cell_box:
818 {
819   \box_rotate:Nn \l_@@_cell_box { 90 }
820   \int_compare:nNnT \c@iRow = \l_@@_last_row_int
821   {
822     \vbox_set_top:Nn \l_@@_cell_box
823     {
824       \vbox_to_zero:n { }
825       \skip_vertical:n { - \box_ht:N \@arstrutbox + 0.8 ex }
826       \box_use:N \l_@@_cell_box
827     }
828   }
829   \bool_gset_false:N \g_@@_rotate_bool
830 }

```

```

831 \cs_new_protected:Npn \@@_adjust_size_box:
832 {
833   \dim_compare:nNnT \g_@@_blocks_wd_dim > \c_zero_dim
834   {
835     \box_set_wd:Nn \l_@@_cell_box
836     { \dim_max:nn { \box_wd:N \l_@@_cell_box } \g_@@_blocks_wd_dim }
837     \dim_gzero:N \g_@@_blocks_wd_dim
838   }
839   \dim_compare:nNnT \g_@@_blocks_dp_dim > \c_zero_dim
840   {
841     \box_set_dp:Nn \l_@@_cell_box
842     { \dim_max:nn { \box_dp:N \l_@@_cell_box } \g_@@_blocks_dp_dim }
843     \dim_gzero:N \g_@@_blocks_dp_dim
844   }
845   \dim_compare:nNnT \g_@@_blocks_ht_dim > \c_zero_dim
846   {
847     \box_set_ht:Nn \l_@@_cell_box
848     { \dim_max:nn { \box_ht:N \l_@@_cell_box } \g_@@_blocks_ht_dim }
849     \dim_gzero:N \g_@@_blocks_ht_dim
850   }
851 }

852 \cs_new_protected:Npn \@@_end_Cell:
853 {
854   \@@_math_toggle_token:
855   \hbox_set_end:
856   \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
857   \@@_adjust_size_box:
858   \box_set_ht:Nn \l_@@_cell_box
859   { \box_ht:N \l_@@_cell_box + \l_@@_cell_space_top_limit_dim }
860   \box_set_dp:Nn \l_@@_cell_box
861   { \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”).

```

862   \dim_gset:Nn \g_@@_max_cell_width_dim
863   { \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”.

```

864   \@@_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.

```

865   \bool_if:NTF \g_@@_empty_cell_bool
866   { \box_use_drop:N \l_@@_cell_box }
867   {
868     \bool_lazy_or:nnTF

```

```

869         \g_@@_not_empty_cell_bool
870         { \dim_compare_p:nNn { \box_wd:N \l_@@_cell_box } > \c_zero_dim }
871         \@@_node_for_the_cell:
872         { \box_use_drop:N \l_@@_cell_box }
873     }
874     \bool_gset_false:N \g_@@_empty_cell_bool
875     \bool_gset_false:N \g_@@_not_empty_cell_bool
876 }

```

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

```

877 \cs_new_protected:Npn \@@_node_for_the_cell:
878 {
879     \pgfpicture
880     \pgfsetbaseline \c_zero_dim
881     \pgfrememberpicturepositiononpagetrue
882     \pgfset
883     {
884         inner~sep = \c_zero_dim ,
885         minimum~width = \c_zero_dim
886     }
887     \pgfnode
888     { rectangle }
889     { base }
890     { \box_use_drop:N \l_@@_cell_box }
891     { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
892     { }
893     \str_if_empty:NF \l_@@_name_str
894     {
895         \pgfnodealias
896         { \l_@@_name_str - \int_use:N \c@iRow - \int_use:N \c@jCol }
897         { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
898     }
899     \endpgfpicture
900 }

```

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

For example, for the following matrix,

```

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

```

The first argument is a boolean which indicates whether you must put the instruction on the left or on the right on the list of instructions.

```

901 \cs_new_protected:Npn \@@_instruction_of_type:nnn #1 #2 #3
902 {

```

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).

```

903     \bool_if:nTF { #1 } \tl_gput_right:cx \tl_gput_right:cx
904     { g_@@_ #2 _ lines _ tl }
905     {

```

```

906     \use:c { @@ _ draw _ #2 : nnn }
907     { \int_use:N \c@iRow }
908     { \int_use:N \c@jCol }
909     { \exp_not:n { #3 } }
910   }
911 }

```

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`.

```

912 \cs_new_protected:Npn \@@_revtex_array:
913 {
914   \cs_set_eq:NN \@acol1 \@arrayacol
915   \cs_set_eq:NN \@acolr \@arrayacol
916   \cs_set_eq:NN \@acol \@arrayacol
917   \cs_set_nopar:Npn \@halignto { }
918   \@array@array
919 }
920 \cs_new_protected:Npn \@@_array:
921 {
922   \bool_if:NTF \c_@@_revtex_bool
923     \@@_revtex_array:
924   {
925     \bool_if:NTF \l_@@_NiceTabular_bool
926       { \dim_set_eq:NN \col@sep \tabcolsep }
927       { \dim_set_eq:NN \col@sep \arraycolsep }
928     \dim_compare:nNnTF \l_@@_tabular_width_dim = \c_zero_dim
929       { \cs_set_nopar:Npn \@halignto { } }
930       { \cs_set_nopar:Npx \@halignto { to \dim_use:N \l_@@_tabular_width_dim } }

```

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

```

931   \@tabarray
932 }

```

`\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.

```

933   [ \str_if_eq:VnTF \l_@@_baseline_str c c t ]
934 }

```

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.

```

935 \cs_set_eq:NN \@@_old_ialign: \ialign

```

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

```

936 \cs_new_protected:Npn \@@_create_row_node:
937 {

```

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

```

938   \hbox
939   {
940     \bool_if:NT \l_@@_code_before_bool
941     {
942       \vtop
943       {
944         \skip_vertical:N 0.5\arrayrulewidth
945         \pgfsys@markposition { \@@_env: - row - \@@_succ:n \c@iRow }
946         \skip_vertical:N -0.5\arrayrulewidth
947       }
948     }
949     \pgfpicture
950     \pgfrememberpicturepositiononpagetrue
951     \pgfcoordinate { \@@_env: - row - \@@_succ:n \c@iRow }
952     { \pgfpoint \c_zero_dim { - 0.5 \arrayrulewidth } }

```

```

953     \str_if_empty:NF \l_@@_name_str
954     {
955         \pgfnodealias
956         { \l_@@_name_str - row - \int_use:N \c@iRow }
957         { \@@_env: - row - \int_use:N \c@iRow }
958     }
959     \endpgfpicture
960 }
961 }

```

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

```

962 \cs_new:Npn \@@_everycr: { \noalign { \@@_everycr_i: } }
963 \cs_new_protected:Npn \@@_everycr_i:
964 {
965     \int_gzero:N \c@jCol
966     \bool_if:NF \g_@@_row_of_col_done_bool
967     {
968         \@@_create_row_node:

```

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

```

969     \bool_if:NT \l_@@_hlines_bool
970     {

```

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.

```

971         \int_compare:nNnT \c@iRow > { -1 }
972         {
973             \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@`.

```

974         { \hrule height \arrayrulewidth width \c_zero_dim }
975     }
976 }
977 }
978 }

```

The command `\@@_newcolumntype` is the command `\newcolumntype` of `array` without the warnings for redefinitions of columns types (we will use it to redefine the columns types `w` and `W`).

```

979 \cs_set_protected:Npn \@@_newcolumntype #1
980 {
981     \cs_set:cpn { NC @ find @ #1 } ##1 #1 { \NC@ { ##1 } }
982     \peek_meaning:NTF [
983         { \newcol@ #1 }
984         { \newcol@ #1 [ 0 ] }
985     }

```

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

```

986 \cs_set_protected:Npn \@@_renew_dots:
987 {
988     \cs_set_eq:NN \ldots \@@_Ldots
989     \cs_set_eq:NN \cdots \@@_Cdots
990     \cs_set_eq:NN \vdots \@@_Vdots
991     \cs_set_eq:NN \ddots \@@_Ddots
992     \cs_set_eq:NN \iddots \@@_Iddots
993     \cs_set_eq:NN \dots \@@_Ldots
994     \cs_set_eq:NN \hdotsfor \@@_Hdotsfor:
995 }

```

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

```

996 \cs_new_protected:Npn \@@_colortbl_like:
997 {
998   \cs_set_eq:NN \cellcolor \@@_cellcolor_tabular
999   \cs_set_eq:NN \rowcolor \@@_rowcolor_tabular
1000   \cs_set_eq:NN \columncolor \@@_columncolor_preamble
1001 }

```

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

```

1002 \cs_new_protected:Npn \@@_pre_array:
1003 {

```

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 redefinition³⁷.

```

1004   \bool_if:NT \c_@@_booktabs_loaded_bool
1005   { \tl_put_left:Nn \@BTnormal \@@_create_row_node: }
1006   \box_clear_new:N \l_@@_cell_box
1007   \cs_if_exist:NT \theiRow
1008   { \int_set_eq:NN \l_@@_old_iRow_int \c@iRow }
1009   \int_gzero_new:N \c@iRow
1010   \cs_if_exist:NT \thejCol
1011   { \int_set_eq:NN \l_@@_old_jCol_int \c@jCol }
1012   \int_gzero_new:N \c@jCol
1013   \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}`).

```

1014   \bool_if:NT \l_@@_small_bool
1015   {
1016     \cs_set_nopar:Npn \arraystretch { 0.47 }
1017     \dim_set:Nn \arraycolsep { 1.45 pt }
1018   }

```

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`.

```

1019   \cs_set_nopar:Npn \ialign
1020   {
1021     \bool_if:NTF \c_@@_colortbl_loaded_bool
1022     {
1023       \CT@everycr
1024       {
1025         \noalign { \cs_gset_eq:NN \CT@row@color \prg_do_nothing: }
1026         \@@_everycr:
1027       }
1028     }
1029     { \everycr { \@@_everycr: } }
1030     \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`³⁸ and `\extrarowheight`

³⁷cf. `\nicematrix@redefine@check@rerun`

³⁸The option `small` of `nicematrix` changes (among other) the value of `\arraystretch`. This is done, of course, before the call of `{array}`.

(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`.

```

1031      \dim_gzero_new:N \g_@@_dp_row_zero_dim
1032      \dim_gset:Nn \g_@@_dp_row_zero_dim { \box_dp:N \@arstrutbox }
1033      \dim_gzero_new:N \g_@@_ht_row_zero_dim
1034      \dim_gset:Nn \g_@@_ht_row_zero_dim { \box_ht:N \@arstrutbox }
1035      \dim_gzero_new:N \g_@@_ht_row_one_dim
1036      \dim_gset:Nn \g_@@_ht_row_one_dim { \box_ht:N \@arstrutbox }
1037      \dim_gzero_new:N \g_@@_dp_ante_last_row_dim
1038      \dim_gzero_new:N \g_@@_ht_last_row_dim
1039      \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
1040      \dim_gzero_new:N \g_@@_dp_last_row_dim
1041      \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`.

```

1042      \cs_set_eq:NN \ialign \@_old_ialign:
1043      \halign
1044  }

```

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).

```

1045      \cs_set_eq:NN \@_old_ldots \ldots
1046      \cs_set_eq:NN \@_old_cdots \cdots
1047      \cs_set_eq:NN \@_old_vdots \vdots
1048      \cs_set_eq:NN \@_old_ddots \ddots
1049      \cs_set_eq:NN \@_old_iddots \iddots
1050      \bool_if:NTF \l_@@_standard_cline_bool
1051      { \cs_set_eq:NN \cline \@_standard_cline }
1052      { \cs_set_eq:NN \cline \@_cline }
1053      \cs_set_eq:NN \Ldots \@_Ldots
1054      \cs_set_eq:NN \Cdots \@_Cdots
1055      \cs_set_eq:NN \Vdots \@_Vdots
1056      \cs_set_eq:NN \Ddots \@_Ddots
1057      \cs_set_eq:NN \Iddots \@_Iddots
1058      \cs_set_eq:NN \hdottedline \@_hdottedline:
1059      \cs_set_eq:NN \Hline \@_Hline:
1060      \cs_set_eq:NN \Hspace \@_Hspace:
1061      \cs_set_eq:NN \Hdotsfor \@_Hdotsfor:
1062      \cs_set_eq:NN \Vdotsfor \@_Vdotsfor:
1063      \cs_set_eq:NN \multicolumn \@_multicolumn:n
1064      \cs_set_eq:NN \Block \@_Block:
1065      \cs_set_eq:NN \rotate \@_rotate:
1066      \cs_set_eq:NN \OnlyMainNiceMatrix \@_OnlyMainNiceMatrix:n
1067      \cs_set_eq:NN \dotfill \@_old_dotfill:
1068      \cs_set_eq:NN \CodeAfter \@_CodeAfter:
1069      \cs_set_eq:NN \diagbox \@_diagbox:nn
1070      \cs_set_eq:NN \NotEmpty \@_NotEmpty:
1071      \bool_if:NT \l_@@_colortbl_like_bool \@_colortbl_like:
1072      \bool_if:NT \l_@@_renew_dots_bool \@_renew_dots:

```

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

```

1073      \seq_gclear_new:N \g_@@_multicolumn_cells_seq
1074      \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).

```

1075      \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`).

```
1076 \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.

```
1077 \int_gzero_new:N \g_@@_col_total_int
1078 \cs_set_eq:NN \@ifnextchar \new@ifnextchar
1079 \@@_renew_NC@rewrite@S:
1080 \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.

```
1081 \tl_gclear_new:N \g_@@_Cdots_lines_tl
1082 \tl_gclear_new:N \g_@@_Ldots_lines_tl
1083 \tl_gclear_new:N \g_@@_Vdots_lines_tl
1084 \tl_gclear_new:N \g_@@_Ddots_lines_tl
1085 \tl_gclear_new:N \g_@@_Iddots_lines_tl
1086 \tl_gclear_new:N \g_@@_HVDotsfor_lines_tl

1087 \tl_gclear_new:N \g_nicematrix_code_before_tl
1088 }
```

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

The environment `{NiceArrayWithDelims}`

```
1089 \NewDocumentEnvironment { NiceArrayWithDelims } { m m O { } m ! O { } }
1090 {
1091 \@@_provide_pgfsyspdfmark:
1092 \bool_if:NT \c_@@_footnote_bool \savenotes
```

The aim of the following `\bgroup` (the corresponding `\egroup` is, of course, at the end of the environment) is to be able to put an exposant to a matrix in a mathematical formula.

```
1093 \bgroup
1094 \tl_set:Nn \l_@@_left_delim_tl { #1 }
1095 \tl_set:Nn \l_@@_right_delim_tl { #2 }
1096 \int_gzero:N \g_@@_block_box_int
1097 \dim_zero:N \g_@@_width_last_col_dim
1098 \dim_zero:N \g_@@_width_first_col_dim
1099 \bool_gset_false:N \g_@@_row_of_col_done_bool
1100 \str_if_empty:NT \g_@@_name_env_str
1101 { \str_gset:Nn \g_@@_name_env_str { NiceArrayWithDelims } }
1102 \@@_adapt_S_column:
1103 \bool_if:NTF \l_@@_NiceTabular_bool
1104 \mode_leave_vertical:
1105 \@@_test_if_math_mode:
1106 \bool_if:NT \l_@@_in_env_bool { \@@_fatal:n { Yet~in~env } }
1107 \bool_set_true:N \l_@@_in_env_bool
```

The command `\CT@arc@` contains the instruction of color for the rules of the array³⁹. 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.

```
1108 \cs_gset_eq:NN \@@_old_CT@arc@ \CT@arc@
```

³⁹e.g. `\color[rgb]{0.5,0.5,0}`

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

```

1109 \cs_if_exist:NT \tikz@library@external@loaded
1110 {
1111     \tikzexternaldisable
1112     \cs_if_exist:NT \ifstandalone
1113     { \tikzset { external / optimize = false } }
1114 }

```

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

```

1115 \int_gincr:N \g_@@_env_int
1116 \bool_if:NF \l_@@_block_auto_columns_width_bool
1117 { \dim_gzero_new:N \g_@@_max_cell_width_dim }

```

The sequence `\g_@@_blocks_seq` will contain the characteristics of the blocks (specified by `\Block`) of the array. The sequence `\g_@@_pos_of_blocks_seq` will contain only the position of the blocks. Of course, this is redundant but it's for efficiency.

```

1118 \seq_gclear:N \g_@@_blocks_seq
1119 \seq_gclear:N \g_@@_pos_of_blocks_seq
1120 \seq_gclear:N \g_@@_pos_of_xdots_seq

```

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

```

1121 \tl_if_exist:cT { g_@@_code_before_ \int_use:N \g_@@_env_int _ tl }
1122 {
1123     \bool_set_true:N \l_@@_code_before_bool
1124     \exp_args:NNv \tl_put_right:Nn \l_@@_code_before_tl
1125     { g_@@_code_before_ \int_use:N \g_@@_env_int _ tl }
1126 }

```

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`.

```

1127 \bool_if:NTF \l_@@_NiceArray_bool
1128 { \keys_set:nn { NiceMatrix / NiceArray } }
1129 { \keys_set:nn { NiceMatrix / pNiceArray } }
1130 { #3 , #5 }

1131 \tl_if_empty:NF \l_@@_rules_color_tl
1132 { \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.

```

1133 \bool_if:NT \l_@@_code_before_bool
1134 {
1135     \seq_if_exist:cT { @@_size_ \int_use:N \g_@@_env_int _ seq }
1136     {

```

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).

```

1137 \int_zero_new:N \c@iRow
1138 \int_set:Nn \c@iRow
1139 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 2 }
1140 \int_zero_new:N \c@jCol
1141 \int_set:Nn \c@jCol
1142 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 4 }

```

We have to adjust the values of `\c@iRow` and `\c@jCol` to take into account the potential last row and last column. A value of `-2` for `\l_@@_last_row_int` means that there is no last row. Idem for the columns.

```

1143 \int_compare:nNnF \l_@@_last_row_int = { -2 }
1144 { \int_decr:N \c@iRow }

```

```

1145 \int_compare:nNnF \l_@@_last_col_int = { -2 }
1146 { \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.

```

1147 \pgfsys@markposition { \@@_env: - position }
1148 \pgfsys@getposition { \@@_env: - position } \@@_picture_position:
1149 \pgfpicture

```

First, the creation of the row nodes.

```

1150 \int_step_inline:nnn
1151 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 1 }
1152 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 2 + 1 }
1153 {
1154 \pgfsys@getposition { \@@_env: - row - ##1 } \@@_node_position:
1155 \pgfcoordinate { \@@_env: - row - ##1 }
1156 { \pgfpointdiff \@@_picture_position: \@@_node_position: }
1157 }

```

Now, the creation of the col nodes.

```

1158 \int_step_inline:nnn
1159 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 3 }
1160 { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 4 + 1 }
1161 {
1162 \pgfsys@getposition { \@@_env: - col - ##1 } \@@_node_position:
1163 \pgfcoordinate { \@@_env: - col - ##1 }
1164 { \pgfpointdiff \@@_picture_position: \@@_node_position: }
1165 }
1166 \endpgfpicture
1167 \group_begin:
1168 \bool_if:NT \c_@@_tikz_loaded_bool
1169 {
1170 \tikzset
1171 {
1172 every-picture / .style =
1173 { overlay , name-prefix = \@@_env: - }
1174 }
1175 }
1176 \cs_set_eq:NN \cellcolor \@@_cellcolor
1177 \cs_set_eq:NN \rectanglecolor \@@_rectanglecolor
1178 \cs_set_eq:NN \rowcolor \@@_rowcolor
1179 \cs_set_eq:NN \rowcolors \@@_rowcolors
1180 \cs_set_eq:NN \columncolor \@@_columncolor
1181 \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.

```

1182 \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
1183 \l_@@_code_before_tl
1184 \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
1185 \group_end:
1186 }
1187 }

```

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).

```

1188 \int_compare:nNnT \l_@@_last_row_int > { -2 }
1189 {
1190 \tl_put_right:Nn \@@_update_for_first_and_last_row:
1191 {
1192 \dim_gset:Nn \g_@@_ht_last_row_dim
1193 { \dim_max:nn \g_@@_ht_last_row_dim { \box_ht:N \l_@@_cell_box } }
1194 \dim_gset:Nn \g_@@_dp_last_row_dim
1195 { \dim_max:nn \g_@@_dp_last_row_dim { \box_dp:N \l_@@_cell_box } }
1196 }

```

```

1197     }
1198     \int_compare:nNnT \l_@@_last_row_int = { -1 }
1199     {
1200         \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.

```

1201     \str_if_empty:NTF \l_@@_name_str
1202     {
1203         \cs_if_exist:cT { @@_last_row_ \int_use:N \g_@@_env_int }
1204         {
1205             \int_set:Nn \l_@@_last_row_int
1206             { \use:c { @@_last_row_ \int_use:N \g_@@_env_int } }
1207         }
1208     }
1209     {
1210         \cs_if_exist:cT { @@_last_row_ \l_@@_name_str }
1211         {
1212             \int_set:Nn \l_@@_last_row_int
1213             { \use:c { @@_last_row_ \l_@@_name_str } }
1214         }
1215     }
1216 }

```

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).

```

1217     \int_compare:nNnT \l_@@_last_col_int = { -1 }
1218     {
1219         \str_if_empty:NTF \l_@@_name_str
1220         {
1221             \cs_if_exist:cT { @@_last_col_ \int_use:N \g_@@_env_int }
1222             {
1223                 \int_set:Nn \l_@@_last_col_int
1224                 { \use:c { @@_last_col_ \int_use:N \g_@@_env_int } }
1225             }
1226         }
1227         {
1228             \cs_if_exist:cT { @@_last_col_ \l_@@_name_str }
1229             {
1230                 \int_set:Nn \l_@@_last_col_int
1231                 { \use:c { @@_last_col_ \l_@@_name_str } }
1232             }
1233         }
1234     }

```

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

```

1235     \@@_pre_array:

```

We compute the width of the two delimiters.

```

1236     \dim_zero_new:N \l_@@_left_delim_dim
1237     \dim_zero_new:N \l_@@_right_delim_dim
1238     \bool_if:NTF \l_@@_NiceArray_bool
1239     {
1240         \dim_gset:Nn \l_@@_left_delim_dim { 2 \arraycolsep }
1241         \dim_gset:Nn \l_@@_right_delim_dim { 2 \arraycolsep }
1242     }
1243     {

```

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

```

1244     \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 #1 $ }
1245     \dim_set:Nn \l_@@_left_delim_dim { \box_wd:N \l_tmpa_box }
1246     \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 #2 $ }
1247     \dim_set:Nn \l_@@_right_delim_dim { \box_wd:N \l_tmpa_box }
1248 }

```

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

```
1249 \box_clear_new:N \l_@@_the_array_box
```

If the user has loaded `nicematrix` with the option `define-L-C-R`, he will be able to use `L`, `C` and `R` instead of `l`, `c` and `r` in the preambles of the environments of `nicematrix` (it's a compatibility mode since `L`, `C` and `R` were mandatory before version 5.0).

```
1250 \bool_if:NT \c_@@_define_L_C_R_bool \@@_define_L_C_R:
```

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

```
1251 \@@_construct_preamble:n { #4 }
```

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

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

```
1252 \hbox_set:Nw \l_@@_the_array_box
1253 \skip_horizontal:N \l_@@_left_margin_dim
1254 \skip_horizontal:N \l_@@_extra_left_margin_dim
1255 \c_math_toggle_token
1256 \bool_if:NTF \l_@@_light_syntax_bool
1257   { \use:c { @@-light-syntax } }
1258   { \use:c { @@-normal-syntax } }
1259 }
1260 {
1261   \bool_if:NTF \l_@@_light_syntax_bool
1262     { \use:c { end @@-light-syntax } }
1263     { \use:c { end @@-normal-syntax } }
1264   \c_math_toggle_token
1265   \skip_horizontal:N \l_@@_right_margin_dim
1266   \skip_horizontal:N \l_@@_extra_right_margin_dim
1267   \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).

```
1268 \int_compare:nNnT \l_@@_last_row_int > { -2 }
1269   {
1270     \bool_if:NF \l_@@_last_row_without_value_bool
1271     {
1272       \int_compare:nNnF \l_@@_last_row_int = \c@iRow
1273       {
1274         \@@_error:n { Wrong~last~row }
1275         \int_gset_eq:NN \l_@@_last_row_int \c@iRow
1276       }
1277     }
1278   }
```

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”.⁴⁰

```
1279 \int_gset_eq:NN \c@jCol \g_@@_col_total_int
1280 \bool_if:nTF \g_@@_last_col_found_bool
1281   { \int_gdecr:N \c@jCol }
1282   {
1283     \int_compare:nNnT \l_@@_last_col_int > { -1 }
1284     { \@@_error:n { last~col~not~used } }
1285   }
1286 \bool_if:NF \l_@@_Matrix_bool
```

⁴⁰We remind that the potential “first column” (exterior) has the number 0.

```

1287 {
1288   \int_compare:nNnT \c@jCol < \g_@@_static_num_of_col_int
1289     { \@@_error:n { columns-not-used } }
1290 }

```

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

```

1291 \int_gset_eq:NN \g_@@_row_total_int \c@iRow
1292 \int_compare:nNnT \l_@@_last_row_int > { -1 } { \int_gdecr:N \c@iRow }

```

Now, we begin the real construction in the output flow of TeX. First, we take into account a potential “first column” (we remind that this “first column” has been constructed in an overlapping position and that we have computed its width in `\g_@@_width_first_col_dim`: see p. 92).

```

1293 \int_compare:nNnT \l_@@_first_col_int = 0
1294 {
1295   \skip_horizontal:N \col@sep
1296   \skip_horizontal:N \g_@@_width_first_col_dim
1297 }

```

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

Remark that, in all cases, `@@_use_arraybox_with_notes_c:` is used.

```

1298 \bool_if:NTF \l_@@_NiceArray_bool
1299 {
1300   \str_case:VnF \l_@@_baseline_str
1301   {
1302     b \@@_use_arraybox_with_notes_b:
1303     c \@@_use_arraybox_with_notes_c:
1304   }
1305   \@@_use_arraybox_with_notes:
1306 }

```

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).

```

1307 {
1308   \int_compare:nNnTF \l_@@_first_row_int = 0
1309   {
1310     \dim_set_eq:NN \l_tmpa_dim \g_@@_dp_row_zero_dim
1311     \dim_add:Nn \l_tmpa_dim \g_@@_ht_row_zero_dim
1312   }
1313   { \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”.⁴¹

```

1314 \int_compare:nNnTF \l_@@_last_row_int > { -2 }
1315 {
1316   \dim_set_eq:NN \l_tmpb_dim \g_@@_ht_last_row_dim
1317   \dim_add:Nn \l_tmpb_dim \g_@@_dp_last_row_dim
1318 }
1319 { \dim_zero:N \l_tmpb_dim }
1320 \hbox_set:Nn \l_tmpa_box
1321 {
1322   \c_math_toggle_token
1323   \left #1
1324   \vcenter
1325   {

```

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.

```

1326   \skip_vertical:N -\l_tmpa_dim
1327   \skip_vertical:N -\arrayrulewidth
1328   \hbox
1329   {

```

⁴¹A 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).

```

1330         \bool_if:NTF \l_@@_NiceTabular_bool
1331         { \skip_horizontal:N -\tabcolsep }
1332         { \skip_horizontal:N -\arraycolsep }
1333     \@@_use_arraybox_with_notes_c:
1334     \bool_if:NTF \l_@@_NiceTabular_bool
1335     { \skip_horizontal:N -\tabcolsep }
1336     { \skip_horizontal:N -\arraycolsep }
1337 }

```

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

```

1338     \skip_vertical:N -\l_tmpb_dim
1339     \skip_vertical:N \arrayrulewidth
1340 }
1341 \right #2
1342 \c_math_toggle_token
1343 }

```

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.

```

1344     \bool_if:NTF \l_@@_max_delimiter_width_bool
1345     { \@@_put_box_in_flow_bis:nn { #1 } { #2 } }
1346     \@@_put_box_in_flow:
1347 }

```

We take into account a potential “last column” (this “last column” has been constructed in an overlapping position and we have computed its width in `\g_@@_width_last_col_dim`: see p. 93).

```

1348     \bool_if:NT \g_@@_last_col_found_bool
1349     {
1350         \skip_horizontal:N \g_@@_width_last_col_dim
1351         \skip_horizontal:N \col@sep
1352     }
1353     \@@_after_array:

```

The aim of the following `\egroup` (the corresponding `\bgroup` is, of course, at the beginning of the environment) is to be able to put an exposant to a matrix in a mathematical formula.

```

1354     \egroup
1355     \bool_if:NT \c_@@_footnote_bool \endsavenotes
1356 }

```

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

We construct the preamble of the array

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

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

```

1357 \cs_new_protected:Npn \@@_construct_preamble:n #1
1358 {

```

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

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

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

```

1359     \group_begin:

```

If we are in an environment without explicit preamble, we have nothing to do (excepted the treatment on both sides of the preamble which will be done at the end).

```

1360 \bool_if:NTF \l_@@_Matrix_bool
1361 { \tl_gset:Nn \g_@@_preamble_tl { #1 } }
1362 {
1363   \@@_newcolumnntype w [ 2 ] { \@@_w: { ##1 } { ##2 } }
1364   \@@_newcolumnntype W [ 2 ] { \@@_W: { ##1 } { ##2 } }

```

First, we have to store our preamble in the token register `\@temptokena` (those “token registers” are not supported by `expl3`).

```

1365 \@temptokena { #1 }

```

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

```

1366 \@tempswatrue

```

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

```

1367 \@whilesw \if@tempswa \fi { \@tempswafalse \the \NC@list }

```

Now, we have to “patch” that preamble by transforming some columns. We will insert in the TeX flow the preamble in its actual form (that is to say after the “expansion”) following by a marker `\q_stop` and we will consume these tokens constructing the (new form of the) preamble in `\g_@@_preamble_tl`. This is done recursively with the command `\@@_patch_preamble:n`. In the same time, we will count the columns with the counter `\c@jCol`.

```

1368 \int_gzero_new:N \c@jCol
1369 \bool_if:NTF \l_@@_vlines_bool
1370 {
1371   \tl_gset:Nn \g_@@_preamble_tl
1372   { ! { \skip_horizontal:N \arrayrulewidth } }
1373 }
1374 { \tl_gclear:N \g_@@_preamble_tl }

```

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

```

1375 \int_zero:N \l_tmpa_int

```

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

```

1376 \exp_after:wN \@@_patch_preamble:n \the \@temptokena \q_stop
1377 \int_gset_eq:NN \g_@@_static_num_of_col_int \c@jCol
1378 }

```

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

```

1379 \bool_if:NT \l_@@_colortbl_like_bool
1380 {
1381   \regex_replace_all:NnN
1382   \c_@@_columncolor_regex
1383   { \c { @@_columncolor_preamble } }
1384   \g_@@_preamble_tl
1385 }

```

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

```

1386 \int_compare:nNnTF \l_@@_first_col_int = 0
1387 { \tl_gput_left:NV \g_@@_preamble_tl \c_@@_preamble_first_col_tl }
1388 {
1389   \bool_lazy_all:nT
1390   {
1391     \l_@@_NiceArray_bool
1392     { \bool_not_p:n \l_@@_NiceTabular_bool }
1393     { \bool_not_p:n \l_@@_vlines_bool }
1394     { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
1395   }
1396   { \tl_gput_left:Nn \g_@@_preamble_tl { @ { } } }
1397 }
1398 \int_compare:nNnTF \l_@@_last_col_int > { -1 }
1399 { \tl_gput_right:NV \g_@@_preamble_tl \c_@@_preamble_last_col_tl }

```



```

1400 {
1401   \bool_lazy_all:nT
1402   {
1403     \l_@@_NiceArray_bool
1404     { \bool_not_p:n \l_@@_NiceTabular_bool }
1405     { \bool_not_p:n \l_@@_vlines_bool }
1406     { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
1407   }
1408   { \tl_gput_right:Nn \g_@@_preamble_tl { @ { } } }
1409 }

```

We add a last column to raise a good error message when the user put more columns than allowed by its preamble. However, for technical reasons, it's not possible to do that in `{NiceTabular*}` (`\l_@@_tabular_width_dim=0pt`).

```

1410 \dim_compare:nNnT \l_@@_tabular_width_dim = \c_zero_dim
1411 {
1412   \tl_gput_right:Nn
1413   \g_@@_preamble_tl
1414   { > { \@@_error_too_much_cols: } 1 }
1415 }

```

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

```

1416 \group_end:
1417 }

```

```

1418 \cs_new_protected:Npn \@@_patch_preamble:n #1
1419 {
1420   \str_case:nnF { #1 }
1421   {
1422     c { \@@_patch_preamble_i:n #1 }
1423     l { \@@_patch_preamble_i:n #1 }
1424     r { \@@_patch_preamble_i:n #1 }
1425     > { \@@_patch_preamble_ii:nn #1 }
1426     ! { \@@_patch_preamble_ii:nn #1 }
1427     @ { \@@_patch_preamble_ii:nn #1 }
1428     | { \@@_patch_preamble_iii:n #1 }
1429     p { \@@_patch_preamble_iv:nnn t #1 }
1430     m { \@@_patch_preamble_iv:nnn c #1 }
1431     b { \@@_patch_preamble_iv:nnn b #1 }
1432     \@@_w: { \@@_patch_preamble_v:nnnn { } #1 }
1433     \@@_W: { \@@_patch_preamble_v:nnnn { \cs_set_eq:NN \hss \hfil } #1 }
1434     \@@_true_c: { \@@_patch_preamble_vi:n #1 }
1435     \q_stop { }
1436   }
1437   {
1438     \str_if_eq:VnTF \l_@@_letter_for_dotted_lines_str { #1 }
1439     { \@@_patch_preamble_vii:n #1 }
1440     { \@@_fatal:nn { unknown~column~type } { #1 } }
1441   }
1442 }

```

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

```

1443 \cs_new_protected:Npn \@@_patch_preamble_i:n #1
1444 {
1445   \tl_gput_right:Nn \g_@@_preamble_tl
1446   {
1447     > { \@@_Cell: \tl_set:Nn \l_@@_cell_type_tl { #1 } }
1448     #1
1449     < \@@_end_Cell:
1450   }

```

We increment the counter of columns.

```

1451 \int_gincr:N \c@jCol
1452 \@@_patch_preamble_viii:n
1453 }

```

For >, ! and @

```

1454 \cs_new_protected:Npn \@@_patch_preamble_ii:nn #1 #2
1455 {
1456 \tl_gput_right:Nn \g_@@_preamble_tl { #1 { #2 } }
1457 \@@_patch_preamble:n
1458 }

```

For |

```

1459 \cs_new_protected:Npn \@@_patch_preamble_iii:n #1
1460 {

```

\l_tmpa_int is the number of successive occurrences of |

```

1461 \int_incr:N \l_tmpa_int
1462 \@@_patch_preamble_iii_i:n
1463 }

1464 \cs_new_protected:Npn \@@_patch_preamble_iii_i:n #1
1465 {
1466 \str_if_eq:nnTF { #1 } |
1467 { \@@_patch_preamble_iii:n | }
1468 {
1469 \tl_gput_right:Nx \g_@@_preamble_tl
1470 {
1471 \exp_not:N !
1472 {
1473 \skip_horizontal:n
1474 {
1475 \dim_eval:n
1476 {
1477 \arrayrulewidth * \l_tmpa_int
1478 + \doublerulesep * ( \l_tmpa_int - 1)
1479 }
1480 }
1481 }
1482 }
1483 \tl_gput_right:Nx \g_@@_internal_code_after_tl
1484 { \@@_vline:nn { \@@_succ:n \c@jCol } { \int_use:N \l_tmpa_int } }
1485 \int_zero:N \l_tmpa_int
1486 \@@_patch_preamble:n #1
1487 }
1488 }

```

For p, m and b

```

1489 \cs_new_protected:Npn \@@_patch_preamble_iv:nnn #1 #2 #3
1490 {
1491 \tl_gput_right:Nn \g_@@_preamble_tl
1492 {
1493 > {
1494 \@@_Cell:
1495 \begin { minipage } [ #1 ] { #3 }
1496 \mode_leave_vertical:
1497 \box_use:N \@arstrutbox
1498 }
1499 c
1500 < { \box_use:N \@arstrutbox \end { minipage } \@@_end_Cell: }
1501 }

```

We increment the counter of columns.

```

1502     \int_gincr:N \c@jCol
1503     \@@_patch_preamble_viii:n
1504 }

```

For w and W

```

1505 \cs_new_protected:Npn \@@_patch_preamble_v:nnnn #1 #2 #3 #4
1506 {
1507     \tl_gput_right:Nn \g_@@_preamble_tl
1508     {
1509         > {
1510             \hbox_set:Nw \l_@@_cell_box
1511             \@@_Cell:
1512             \tl_set:Nn \l_@@_cell_type_tl { #3 }
1513         }
1514         c
1515         < {
1516             \@@_end_Cell:
1517             #1
1518             \hbox_set_end:
1519             \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
1520             \@@_adjust_size_box:
1521             \makebox [ #4 ] [ #3 ] { \box_use_drop:N \l_@@_cell_box }
1522         }
1523     }

```

We increment the counter of columns.

```

1524     \int_gincr:N \c@jCol
1525     \@@_patch_preamble_viii:n
1526 }

```

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

```

1527 \cs_new_protected:Npn \@@_patch_preamble_vi:n #1
1528 {
1529     \tl_gput_right:Nn \g_@@_preamble_tl { c }

```

We increment the counter of columns.

```

1530     \int_gincr:N \c@jCol
1531     \@@_patch_preamble_viii:n
1532 }

1533 \cs_new_protected:Npn \@@_patch_preamble_vii:n #1
1534 {
1535     \tl_gput_right:Nn \g_@@_preamble_tl
1536     { ! { \skip_horizontal:N 2\l_@@_radius_dim } }

```

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

```

1537     \tl_gput_right:Nx \g_@@_internal_code_after_tl
1538     { \@@_vdottedline:n { \int_use:N \c@jCol } }
1539     \@@_patch_preamble:n
1540 }

```

After a specifier of column, we have to test whether there is one or several <{..} because, after those potential <{...}, we have to insert !{\skip_horizontal:N ...} when the key vlines is used.

```

1541 \cs_new_protected:Npn \@@_patch_preamble_viii:n #1
1542 {
1543     \str_if_eq:nnTF { #1 } { < }
1544     \@@_patch_preamble_ix:n
1545     {
1546         \bool_if:NT \l_@@_vlines_bool
1547         {
1548             \tl_gput_right:Nn \g_@@_preamble_tl
1549             { ! { \skip_horizontal:N \arrayrulewidth } }
1550         }

```

```

1551     \@@_patch_preamble:n { #1 }
1552   }
1553 }
1554 \cs_new_protected:Npn \@@_patch_preamble_ix:n #1
1555 {
1556   \tl_gput_right:Nn \g_@@_preamble_tl { < { #1 } }
1557   \@@_patch_preamble_viii:n
1558 }

```

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`).

```

1559 \cs_new_protected:Npn \@@_put_box_in_flow:
1560 {
1561   \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }
1562   \box_set_dp:Nn \l_tmpa_box { \box_dp:N \l_tmpa_box + \l_tmpb_dim }
1563   \str_if_eq:VnTF \l_@@_baseline_str { c }
1564     { \box_use_drop:N \l_tmpa_box }
1565     \@@_put_box_in_flow_i:
1566 }

```

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).

```

1567 \cs_new_protected:Npn \@@_put_box_in_flow_i:
1568 {
1569   \pgfpicture
1570     \@@_qpoint:n { row - 1 }
1571     \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1572     \@@_qpoint:n { row - \@@_succ:n \c@iRow }
1573     \dim_gadd:Nn \g_tmpa_dim \pgf@y
1574     \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).

```

1575   \str_if_in:NnTF \l_@@_baseline_str { line- }
1576   {
1577     \int_set:Nn \l_tmpa_int
1578     {
1579       \str_range:Nnn
1580         \l_@@_baseline_str
1581         6
1582         { \str_count:N \l_@@_baseline_str }
1583     }
1584     \@@_qpoint:n { row - \int_use:N \l_tmpa_int }
1585   }
1586   {
1587     \str_case:VnF \l_@@_baseline_str
1588     {
1589       { t } { \int_set:Nn \l_tmpa_int 1 }
1590       { b } { \int_set_eq:NN \l_tmpa_int \c@iRow }
1591     }
1592     { \int_set:Nn \l_tmpa_int \l_@@_baseline_str }
1593     \bool_lazy_or:nnT
1594       { \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int }
1595       { \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int }
1596     {
1597       \@@_error:n { bad~value~for~baseline }
1598       \int_set:Nn \l_tmpa_int 1
1599     }
1600     \@@_qpoint:n { row - \int_use:N \l_tmpa_int - base }

```

We take into account the position of the mathematical axis.

```

1601         \dim_gsub:Nn \g_tmpa_dim { \fontdimen22 \textfont2 }
1602     }
1603     \dim_gsub:Nn \g_tmpa_dim \pgf@y

```

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

```

1604     \endpgfpicture
1605     \box_move_up:nn \g_tmpa_dim { \box_use_drop:N \l_tmpa_box }
1606     \box_use_drop:N \l_tmpa_box
1607 }

```

The following command is *always* used by `{NiceArrayWithDelims}` (even if, in fact, there is no tabular notes: in fact, it's not possible to know whether there is or not before the composition of the blocks).

```

1608 \cs_new_protected:Npn \@@_use_arraybox_with_notes_c:
1609 {

```

We need a `{minipage}` because we will insert a LaTeX list for the tabular notes (that means that a `\vtop{\hsize=...}` is not enough).

```

1610     \begin { minipage } [ t ] { \box_wd:N \l_@@_the_array_box }
1611     \box_use_drop:N \l_@@_the_array_box

```

We have to draw the blocks right now because there may be tabular notes in some blocks (which are not mono-column: the blocks which are mono-column have been composed in boxes yet)... and we have to create (potentially) the extra nodes before creating the blocks since there are `medium` nodes to create for the blocks.

```

1612     \@@_create_extra_nodes:
1613     \seq_if_empty:NF \g_@@_blocks_seq \@@_draw_blocks:
1614     \bool_lazy_or:nnT
1615     { \int_compare_p:nNn \c@tabularnote > 0 }
1616     { ! \tl_if_empty_p:V \l_@@_tabularnote_tl }
1617     \@@_insert_tabularnotes:
1618     \end { minipage }
1619 }

1620 \cs_new_protected:Npn \@@_insert_tabularnotes:
1621 {
1622     \skip_vertical:N 0.65ex

```

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

```

1623     \group_begin:
1624     \l_@@_notes_code_before_tl
1625     \tl_if_empty:NF \l_@@_tabularnote_tl { \l_@@_tabularnote_tl \par }

```

We compose the tabular notes with a list of `enumitem`. The `\strut` and the `\unskip` are designed to give the ability to put a `\bottomrule` at the end of the notes with a good vertical space.

```

1626     \int_compare:nNnT \c@tabularnote > 0
1627     {
1628         \bool_if:NTF \l_@@_notes_para_bool
1629         {
1630             \begin { tabularnotes* }
1631             \seq_map_inline:Nn \g_@@_tabularnotes_seq { \item ##1 } \strut
1632             \end { tabularnotes* }

```

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

```

1633         \par
1634     }
1635     {
1636         \tabularnotes
1637         \seq_map_inline:Nn \g_@@_tabularnotes_seq { \item ##1 } \strut
1638         \endtabularnotes
1639     }
1640 }
1641 \unskip

```

```

1642 \group_end:
1643 \bool_if:NT \l_@@_notes_bottomrule_bool
1644 {
1645     \bool_if:NTF \c_@@_booktabs_loaded_bool
1646     {

```

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

```

1647     \skip_vertical:N \aboverulesep
\CT@arc@ is the specification of color defined by colortbl but you use it even if colortbl is not loaded.
1648     { \CT@arc@ \hrule height \heavyrulewidth }
1649     }
1650     { \@@_error:n { bottomule~without~booktabs } }
1651     }
1652 \l_@@_notes_code_after_tl
1653 \seq_gclear:N \g_@@_tabularnotes_seq
1654 \int_gzero:N \c@tabularnote
1655 }

```

The case of `baseline` equal to `b`. Remember that, when the key `b` is used, the `{array}` (of `array`) is constructed with the option `t` (and not `b`). Now, we do the translation to take into account the option `b`.

```

1656 \cs_new_protected:Npn \@@_use_arraybox_with_notes_b:
1657 {
1658     \pgfpicture
1659     \@@_qpoint:n { row - 1 }
1660     \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1661     \@@_qpoint:n { row - \int_use:N \c@iRow - base }
1662     \dim_gsub:Nn \g_tmpa_dim \pgf@y
1663     \endpgfpicture
1664     \dim_gadd:Nn \g_tmpa_dim \arrayrulewidth
1665     \int_compare:nNnT \l_@@_first_row_int = 0
1666     {
1667         \dim_gadd:Nn \g_tmpa_dim \g_@@_ht_row_zero_dim
1668         \dim_gadd:Nn \g_tmpa_dim \g_@@_dp_row_zero_dim
1669     }
1670     \box_move_up:nn \g_tmpa_dim { \hbox { \@@_use_arraybox_with_notes_c: } }
1671 }

```

Now, the general case.

```

1672 \cs_new_protected:Npn \@@_use_arraybox_with_notes:
1673 {

```

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

```

1674     \str_if_eq:VnT \l_@@_baseline_str { t }
1675     { \tl_set:Nn \l_@@_baseline_str { 1 } }

```

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

```

1676     \pgfpicture
1677     \@@_qpoint:n { row - 1 }
1678     \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1679     \str_if_in:NnTF \l_@@_baseline_str { line- }
1680     {
1681         \int_set:Nn \l_tmpa_int
1682         {
1683             \str_range:Nnn
1684             \l_@@_baseline_str
1685             6
1686             { \str_count:N \l_@@_baseline_str }
1687         }
1688         \@@_qpoint:n { row - \int_use:N \l_tmpa_int }
1689     }
1690     {
1691         \int_set:Nn \l_tmpa_int \l_@@_baseline_str

```

```

1692 \bool_lazy_or:nnT
1693 { \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int }
1694 { \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int }
1695 {
1696   \@@_error:n { bad~value~for~baseline }
1697   \int_set:Nn \l_tmpa_int 1
1698 }
1699 \@@_qpoint:n { row - \int_use:N \l_tmpa_int - base }
1700 }
1701 \dim_gsub:Nn \g_tmpa_dim \pgf@y
1702 \endpgfpicture
1703 \dim_gadd:Nn \g_tmpa_dim \arrayrulewidth
1704 \int_compare:nNnT \l_@@_first_row_int = 0
1705 {
1706   \dim_gadd:Nn \g_tmpa_dim \g_@@_ht_row_zero_dim
1707   \dim_gadd:Nn \g_tmpa_dim \g_@@_dp_row_zero_dim
1708 }
1709 \box_move_up:nn \g_tmpa_dim { \hbox { \@@_use_arraybox_with_notes_c: } }
1710 }

```

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.

```

1711 \cs_new_protected:Npn \@@_put_box_in_flow_bis:nn #1 #2
1712 {

```

We will compute the real width of both delimiters used.

```

1713 \dim_zero_new:N \l_@@_real_left_delim_dim
1714 \dim_zero_new:N \l_@@_real_right_delim_dim
1715 \hbox_set:Nn \l_tmpb_box
1716 {
1717   \c_math_toggle_token
1718   \left #1
1719   \vcenter
1720   {
1721     \vbox_to_ht:nn
1722     { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1723     { }
1724   }
1725   \right .
1726   \c_math_toggle_token
1727 }
1728 \dim_set:Nn \l_@@_real_left_delim_dim
1729 { \box_wd:N \l_tmpb_box - \nulldelimiterspace }
1730 \hbox_set:Nn \l_tmpb_box
1731 {
1732   \c_math_toggle_token
1733   \left .
1734   \vbox_to_ht:nn
1735   { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1736   { }
1737   \right #2
1738   \c_math_toggle_token
1739 }
1740 \dim_set:Nn \l_@@_real_right_delim_dim
1741 { \box_wd:N \l_tmpb_box - \nulldelimiterspace }

```

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

```

1742 \skip_horizontal:N \l_@@_left_delim_dim
1743 \skip_horizontal:N -\l_@@_real_left_delim_dim
1744 \@@_put_box_in_flow:
1745 \skip_horizontal:N \l_@@_right_delim_dim
1746 \skip_horizontal:N -\l_@@_real_right_delim_dim
1747 }

```

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

```
1748 \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).

```
1749 {
1750   \peek_meaning_ignore_spaces:NTF \end \@@_analyze_end:Nn
```

Here is the call to `\array` (we have a dedicated macro `\@@_array`: because of compatibility with the classes `revtex4-1` and `revtex4-2`).

```
1751   { \exp_args:NV \@@_array: \g_@@_preamble_tl }
1752 }
1753 {
1754   \@@_create_col_nodes:
1755   \endarray
1756 }
```

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

```
1757 \NewDocumentEnvironment { @@-light-syntax } { b }
1758 {
```

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`.

```
1759   \tl_if_empty:nT { #1 } { \@@_fatal:n { empty-environment } }
1760   \tl_map_inline:nn { #1 }
1761   {
1762     \tl_if_eq:nnT { ##1 } { & }
1763     { \@@_fatal:n { ampersand-in-light-syntax } }
1764     \tl_if_eq:nnT { ##1 } { \ }
1765     { \@@_fatal:n { double-backslash-in-light-syntax } }
1766   }
```

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

```
1767   \@@_light_syntax_i #1 \CodeAfter \q_stop
1768 }
```

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`.

```
1769 { }
1770 \cs_new_protected:Npn \@@_light_syntax_i #1\CodeAfter #2\q_stop
1771 {
1772   \tl_gput_right:Nn \g_nicematrix_code_after_tl { #2 }
```

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

```
1773   \seq_gclear_new:N \g_@@_rows_seq
1774   \tl_set_rescan:Nno \l_@@_end_of_row_tl { } \l_@@_end_of_row_tl
1775   \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.

```
1776   \int_compare:nNnT \l_@@_last_row_int = { -1 }
1777   { \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`).

```

1778 \exp_args:NV \@@_array: \g_@@_preamble_tl
We need a global affectation because, when executing \l_tmpa_tl, we will exit the first cell of the
array.
1779 \seq_gpop_left:NN \g_@@_rows_seq \l_tmpa_tl
1780 \exp_args:NV \@@_line_with_light_syntax_i:n \l_tmpa_tl
1781 \seq_map_function:NN \g_@@_rows_seq \@@_line_with_light_syntax:n
1782 \@@_create_col_nodes:
1783 \endarray
1784 }
1785 \cs_new_protected:Npn \@@_line_with_light_syntax:n #1
1786 { \tl_if_empty:NF { #1 } { \ \ \@@_line_with_light_syntax_i:n { #1 } } }
1787 \cs_new_protected:Npn \@@_line_with_light_syntax_i:n #1
1788 {
1789 \seq_gclear_new:N \g_@@_cells_seq
1790 \seq_gset_split:Nnn \g_@@_cells_seq { ~ } { #1 }
1791 \seq_gpop_left:NN \g_@@_cells_seq \l_tmpa_tl
1792 \l_tmpa_tl
1793 \seq_map_inline:Nn \g_@@_cells_seq { & ##1 }
1794 }

```

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).

```

1795 \cs_new_protected:Npn \@@_analyze_end:Nn #1 #2
1796 {
1797 \str_if_eq:VnT \g_@@_name_env_str { #2 }
1798 { \@@_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.

```

1799 \end { #2 }
1800 }

```

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).

```

1801 \cs_new:Npn \@@_create_col_nodes:
1802 {
1803 \crrc
1804 \int_compare:nNnT \l_@@_first_col_int = 0
1805 {
1806 \omit
1807 \hbox_overlap_left:n
1808 {
1809 \bool_if:NT \l_@@_code_before_bool
1810 { \pgfsys@markposition { \@@_env: - col - 0 } }
1811 \pgfpicture
1812 \pgfrememberpicturepositiononpagetrue
1813 \pgfcoordinate { \@@_env: - col - 0 } \pgfpintorigin
1814 \str_if_empty:NF \l_@@_name_str
1815 { \pgfnodelalias { \l_@@_name_str - col - 0 } { \@@_env: - col - 0 } }
1816 \endpgfpicture
1817 \skip_horizontal:N 2\col@sep
1818 \skip_horizontal:N \g_@@_width_first_col_dim
1819 }
1820 &
1821 }
1822 \omit

```

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

```
1823 \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`).

```
1824 \int_compare:nNnTF \l_@@_first_col_int = 0
1825 {
1826   \bool_if:NT \l_@@_code_before_bool
1827   {
1828     \hbox
1829     {
1830       \skip_horizontal:N -0.5\arrayrulewidth
1831       \pgfsys@markposition { \@@_env: - col - 1 }
1832       \skip_horizontal:N 0.5\arrayrulewidth
1833     }
1834   }
1835   \pgfpicture
1836   \pgfrememberpicturepositiononpagetrue
1837   \pgfcoordinate { \@@_env: - col - 1 }
1838   { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
1839   \str_if_empty:NF \l_@@_name_str
1840   { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
1841   \endpgfpicture
1842 }
1843 {
1844   \bool_if:NT \l_@@_code_before_bool
1845   {
1846     \hbox
1847     {
1848       \skip_horizontal:N 0.5 \arrayrulewidth
1849       \pgfsys@markposition { \@@_env: - col - 1 }
1850       \skip_horizontal:N -0.5\arrayrulewidth
1851     }
1852   }
1853   \pgfpicture
1854   \pgfrememberpicturepositiononpagetrue
1855   \pgfcoordinate { \@@_env: - col - 1 }
1856   { \pgfpoint { 0.5 \arrayrulewidth } \c_zero_dim }
1857   \str_if_empty:NF \l_@@_name_str
1858   { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
1859   \endpgfpicture
1860 }
```

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.

```
1861 \skip_gset:Nn \g_tmpa_skip { 0 pt~plus 1 fill }
1862 \bool_if:NF \l_@@_auto_columns_width_bool
1863 { \dim_compare:nNnT \l_@@_columns_width_dim > \c_zero_dim }
1864 {
1865   \bool_lazy_and:nnTF
1866   \l_@@_auto_columns_width_bool
1867   { \bool_not_p:n \l_@@_block_auto_columns_width_bool }
1868   { \skip_gset_eq:NN \g_tmpa_skip \g_@@_max_cell_width_dim }
1869   { \skip_gset_eq:NN \g_tmpa_skip \l_@@_columns_width_dim }
1870   \skip_gadd:Nn \g_tmpa_skip { 2 \col@sep }
1871 }
1872 \skip_horizontal:N \g_tmpa_skip
1873 \hbox
1874 {
```

```

1875 \bool_if:NT \l_@@_code_before_bool
1876 {
1877     \hbox
1878     {
1879         \skip_horizontal:N -0.5\arrayrulewidth
1880         \pgfsys@markposition { \@@_env: - col - 2 }
1881         \skip_horizontal:N 0.5\arrayrulewidth
1882     }
1883 }
1884 \pgfpicture
1885 \pgfrememberpicturepositiononpagetrue
1886 \pgfcoordinate { \@@_env: - col - 2 }
1887 { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
1888 \str_if_empty:NF \l_@@_name_str
1889 { \pgfnodealias { \l_@@_name_str - col - 2 } { \@@_env: - col - 2 } }
1890 \endpgfpicture
1891 }

```

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.

```

1892 \int_gset:Nn \g_tmpa_int 1
1893 \bool_if:NTF \g_@@_last_col_found_bool
1894 { \prg_replicate:nn { \g_@@_col_total_int - 2 } }
1895 { \prg_replicate:nn { \g_@@_col_total_int - 1 } }
1896 {
1897     &
1898     \omit
1899     \int_gincr:N \g_tmpa_int

```

The incrementation of the counter `\g_tmpa_int` must be done after the `\omit` of the cell.

```

1900 \skip_horizontal:N \g_tmpa_skip
1901 \bool_if:NT \l_@@_code_before_bool
1902 {
1903     \hbox
1904     {
1905         \skip_horizontal:N -0.5\arrayrulewidth
1906         \pgfsys@markposition { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1907         \skip_horizontal:N 0.5\arrayrulewidth
1908     }
1909 }

```

We create the col node on the right of the current column.

```

1910 \pgfpicture
1911 \pgfrememberpicturepositiononpagetrue
1912 \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1913 { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
1914 \str_if_empty:NF \l_@@_name_str
1915 {
1916     \pgfnodealias
1917     { \l_@@_name_str - col - \@@_succ:n \g_tmpa_int }
1918     { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1919 }
1920 \endpgfpicture
1921 }
1922 \bool_if:NT \g_@@_last_col_found_bool
1923 {
1924     \hbox_overlap_right:n
1925     {
1926         % \skip_horizontal:N \colsep
1927         \skip_horizontal:N \g_@@_width_last_col_dim
1928         \bool_if:NT \l_@@_code_before_bool
1929         {
1930             \pgfsys@markposition
1931             { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
1932         }

```

```

1933     \pgfpicture
1934     \pgfrememberpicturepositiononpagetrue
1935     \pgfcoordinate { \l_@@_env: - col - \l_@@_succ:n \g_@@_col_total_int }
1936     \pgfpointorigin
1937     \str_if_empty:NF \l_@@_name_str
1938     {
1939         \pgfnodealias
1940         { \l_@@_name_str - col - \l_@@_succ:n \g_@@_col_total_int }
1941         { \l_@@_env: - col - \l_@@_succ:n \g_@@_col_total_int }
1942     }
1943     \endpgfpicture
1944 }
1945 }
1946 \cr
1947 }

```

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

```

1948 \tl_const:Nn \c_@@_preamble_first_col_tl
1949 {
1950     >
1951     {
1952         \l_@@_begin_of_row:

```

The contents of the cell is constructed in the box `\l_@@_cell_box` because we have to compute some dimensions of this box.

```

1953     \hbox_set:Nw \l_@@_cell_box
1954     \l_@@_math_toggle_token:
1955     \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”.

```

1956     \bool_lazy_and:nnT
1957     { \int_compare_p:nNn \c@iRow > 0 }
1958     {
1959         \bool_lazy_or_p:nn
1960         { \int_compare_p:nNn \l_@@_last_row_int < 0 }
1961         { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
1962     }
1963     {
1964         \l_@@_code_for_first_col_tl
1965         \xglobal \colorlet { nicematrix-first-col } { . }
1966     }
1967 }

```

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`.

```

1968     l
1969     <
1970     {
1971         \l_@@_math_toggle_token:
1972         \hbox_set_end:
1973         \bool_if:NT \g_@@_rotate_bool \l_@@_rotate_cell_box:
1974         \l_@@_adjust_size_box:
1975         \l_@@_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.

```

1976     \dim_gset:Nn \g_@@_width_first_col_dim
1977     { \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.

```

1978     \hbox_overlap_left:n
1979     {
1980         \dim_compare:nNnTF { \box_wd:N \l_@@_cell_box } > \c_zero_dim

```

```

1981         \l_@@_node_for_the_cell:
1982         { \box_use_drop:N \l_@@_cell_box }
1983         \skip_horizontal:N \l_@@_left_delim_dim
1984         \skip_horizontal:N \l_@@_left_margin_dim
1985         \skip_horizontal:N \l_@@_extra_left_margin_dim
1986     }
1987     \bool_gset_false:N \g_@@_empty_cell_bool
1988     \skip_horizontal:N -2\col@sep
1989 }
1990 }

```

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

```

1991 \tl_const:Nn \c_@@_preamble_last_col_tl
1992 {
1993     >
1994     {

```

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

```

1995         \bool_gset_true:N \g_@@_last_col_found_bool
1996         \int_gincr:N \c@jCol
1997         \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.

```

1998         \hbox_set:Nw \l_@@_cell_box
1999         \@@_math_toggle_token:
2000         \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”.

```

2001         \int_compare:nNnT \c@iRow > 0
2002         {
2003             \bool_lazy_or:nnT
2004             { \int_compare_p:nNn \l_@@_last_row_int < 0 }
2005             { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
2006             {
2007                 \l_@@_code_for_last_col_tl
2008                 \xglobal \colorlet { nicematrix-last-col } { . }
2009             }
2010         }
2011     }
2012     1
2013     <
2014     {
2015         \@@_math_toggle_token:
2016         \hbox_set_end:
2017         \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:
2018         \@@_adjust_size_box:
2019         \@@_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.

```

2020         \dim_gset:Nn \g_@@_width_last_col_dim
2021         { \dim_max:nn \g_@@_width_last_col_dim { \box_wd:N \l_@@_cell_box } }
2022         \skip_horizontal:N -2\col@sep

```

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

```

2023         \hbox_overlap_right:n
2024         {
2025             \dim_compare:nNnT { \box_wd:N \l_@@_cell_box } > \c_zero_dim
2026             {
2027                 \skip_horizontal:N \l_@@_right_delim_dim
2028                 \skip_horizontal:N \l_@@_right_margin_dim
2029                 \skip_horizontal:N \l_@@_extra_right_margin_dim
2030                 \l_@@_node_for_the_cell:
2031             }

```

```

2032     }
2033     \bool_gset_false:N \g_@@_empty_cell_bool
2034 }
2035 }

```

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.

```

2036 \NewDocumentEnvironment { NiceArray } { }
2037 {
2038     \bool_set_true:N \l_@@_NiceArray_bool
2039     \str_if_empty:NT \g_@@_name_env_str
2040     { \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).

```

2041     \NiceArrayWithDelims . .
2042 }
2043 { \endNiceArrayWithDelims }

```

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

```

2044 \cs_new_protected:Npn \@@_def_env:nnn #1 #2 #3
2045 {
2046     \NewDocumentEnvironment { #1 NiceArray } { }
2047     {
2048         \str_if_empty:NT \g_@@_name_env_str
2049         { \str_gset:Nn \g_@@_name_env_str { #1 NiceArray } }
2050         \@@_test_if_math_mode:
2051         \NiceArrayWithDelims #2 #3
2052     }
2053     { \endNiceArrayWithDelims }
2054 }
2055 \@@_def_env:nnn p ( )
2056 \@@_def_env:nnn b [ ]
2057 \@@_def_env:nnn B \{ \}
2058 \@@_def_env:nnn v | |
2059 \@@_def_env:nnn V \| \|

```

The environment `{NiceMatrix}` and its variants

```

2060 \cs_new_protected:Npn \@@_begin_of_NiceMatrix:nn #1 #2
2061 {
2062     \bool_set_true:N \l_@@_Matrix_bool
2063     \use:c { #1 NiceArray }
2064     {
2065         *
2066         {
2067             \int_compare:nNnTF \l_@@_last_col_int < 0
2068             \c@MaxMatrixCols
2069             { \@@_pred:n \l_@@_last_col_int }
2070         }
2071         { > \@@_Cell: #2 < \@@_end_Cell: }
2072     }
2073 }
2074 \clist_map_inline:nn { { } , p , b , B , v , V }
2075 {
2076     \NewDocumentEnvironment { #1 NiceMatrix } { ! 0 { } }
2077     {

```

```

2078 \str_gset:Nn \g_@@_name_env_str { #1 NiceMatrix }
2079 \tl_set:Nn \l_@@_type_of_col_tl c
2080 \keys_set:nn { NiceMatrix / NiceMatrix } { ##1 }
2081 \exp_args:Nne \@@_begin_of_NiceMatrix:nn { #1 } \l_@@_type_of_col_tl
2082 }
2083 { \use:c { end #1 NiceArray } }
2084 }

```

The following command will be linked to `\NotEmpty` in the environments of `nicematrix`.

```

2085 \cs_new_protected:Npn \@@_NotEmpty:
2086 { \bool_gset_true:N \g_@@_not_empty_cell_bool }

```

The environments `{NiceTabular}` and `{NiceTabular*}`

```

2087 \NewDocumentEnvironment { NiceTabular } { 0 { } m ! 0 { } }
2088 {
2089 \str_gset:Nn \g_@@_name_env_str { NiceTabular }
2090 \keys_set:nn { NiceMatrix / NiceTabular } { #1 , #3 }
2091 \bool_set_true:N \l_@@_NiceTabular_bool
2092 \NiceArray { #2 }
2093 }
2094 { \endNiceArray }

2095 \NewDocumentEnvironment { NiceTabular* } { m 0 { } m ! 0 { } }
2096 {
2097 \str_gset:Nn \g_@@_name_env_str { NiceTabular* }
2098 \dim_set:Nn \l_@@_tabular_width_dim { #1 }
2099 \keys_set:nn { NiceMatrix / NiceTabular } { #2 , #4 }
2100 \bool_set_true:N \l_@@_NiceTabular_bool
2101 \NiceArray { #3 }
2102 }
2103 { \endNiceArray }

```

After the construction of the array

```

2104 \cs_new_protected:Npn \@@_after_array:
2105 {
2106 \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.

```

2107 \bool_if:NT \g_@@_last_col_found_bool
2108 { \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`.

```

2109 \bool_if:NT \l_@@_last_col_without_value_bool
2110 {
2111 \dim_set_eq:NN \l_@@_last_col_int \g_@@_col_total_int
2112 \iow_shipout:Nn \@mainaux \ExplSyntaxOn
2113 \iow_shipout:Nx \@mainaux
2114 {
2115 \cs_gset:cpn { @@_last_col_ \int_use:N \g_@@_env_int }
2116 { \int_use:N \g_@@_col_total_int }
2117 }
2118 \str_if_empty:NF \l_@@_name_str
2119 {
2120 \iow_shipout:Nx \@mainaux

```

```

2121         {
2122             \cs_gset:cpn { @@_last_col_ \l_@@_name_str }
2123             { \int_use:N \g_@@_col_total_int }
2124         }
2125     }
2126     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
2127 }

```

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.

```

2128     \bool_if:NT \l_@@_last_row_without_value_bool
2129     {
2130         \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.

```

2131     \bool_if:NF \l_@@_light_syntax_bool
2132     {
2133         \iow_shipout:Nn \@mainaux \ExplSyntaxOn
2134         \iow_shipout:Nx \@mainaux
2135         {
2136             \cs_gset:cpn { @@_last_row_ \int_use:N \g_@@_env_int }
2137             { \int_use:N \g_@@_row_total_int }
2138         }

```

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.

```

2139         \str_if_empty:NF \l_@@_name_str
2140         {
2141             \iow_shipout:Nx \@mainaux
2142             {
2143                 \cs_gset:cpn { @@_last_row_ \l_@@_name_str }
2144                 { \int_use:N \g_@@_row_total_int }
2145             }
2146         }
2147     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
2148 }
2149 }

```

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

```

2150     \bool_if:NT \l_@@_code_before_bool
2151     {
2152         \iow_now:Nn \@mainaux \ExplSyntaxOn
2153         \iow_now:Nx \@mainaux
2154         { \seq_clear_new:c { @@_size _ \int_use:N \g_@@_env_int _ seq } }
2155         \iow_now:Nx \@mainaux
2156         {
2157             \seq_gset_from_clist:cn { @@_size _ \int_use:N \g_@@_env_int _ seq }
2158             {
2159                 \int_use:N \l_@@_first_row_int ,
2160                 \int_use:N \g_@@_row_total_int ,
2161                 \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`.

```

2162         \bool_lazy_and:nnTF
2163         { \int_compare_p:nNn \l_@@_last_col_int > { -2 } }
2164         { \bool_not_p:n \g_@@_last_col_found_bool }
2165         \@@_succ:n
2166         \int_use:N
2167         \g_@@_col_total_int
2168     }

```

We write also the potential content of `\g_@@_pos_of_blocks_seq` (it will be useful if the commands `\rowcolors` is used with the key `respect-blocks`).

```

2169     \seq_gset_from_clist:cn

```



```

2170         { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq }
2171         { \seq_use:Nnnn \g_@@_pos_of_blocks_seq , , , }
2172     }
2173     \iow_now:Nn \@mainaux \ExplSyntaxOff
2174 }

```

By default, the diagonal lines will be parallelized⁴². 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.

```

2175     \bool_if:NT \l_@@_parallelize_diags_bool
2176     {
2177         \int_gzero_new:N \g_@@_ddots_int
2178         \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.

```

2179         \dim_gzero_new:N \g_@@_delta_x_one_dim
2180         \dim_gzero_new:N \g_@@_delta_y_one_dim
2181         \dim_gzero_new:N \g_@@_delta_x_two_dim
2182         \dim_gzero_new:N \g_@@_delta_y_two_dim
2183     }
2184     \int_zero_new:N \l_@@_initial_i_int
2185     \int_zero_new:N \l_@@_initial_j_int
2186     \int_zero_new:N \l_@@_final_i_int
2187     \int_zero_new:N \l_@@_final_j_int
2188     \bool_set_false:N \l_@@_initial_open_bool
2189     \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.

```

2190     \bool_if:NT \l_@@_small_bool
2191     {
2192         \dim_set:Nn \l_@@_radius_dim { 0.37 pt }
2193         \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.

```

2194         \dim_set:Nn \l_@@_xdots_shorten_dim { 0.6 \l_@@_xdots_shorten_dim }
2195     }

```

Now, we actually draw the dotted lines (specified by `\Cdots`, `\Vdots`, etc.).

```

2196     \@@_draw_dotted_lines:

```

The following computes the “corners” (made up of empty cells) but if there is no corner to compute, it will do nothing.

```

2197     \@@_compute_corners:

```

The sequence `\g_@@_pos_of_blocks_seq` must be “adjusted” (for the case where the user have written something like `\Block{1-}`).

```

2198     \@@_adjust_pos_of_blocks_seq:

```

The following code is only for efficiency. We determine whether the potential horizontal and vertical rules are “complete”, that is to say drawn in the whole array. We are sure that all the rules will be complete when there is no block, no virtual block (determined by a command such as `\Cdots`, `\Vdots`, etc.) and no corners. In that case, we switch to a shortcut version of `\@@_vline_i:nn` and `\@@_hline:nn`.

```

2199     \bool_lazy_all:nT
2200     {

```

⁴²It's possible to use the option `parallelize-diags` to disable this parallelization.

```

2201     { \seq_if_empty_p:N \g_@@_pos_of_blocks_seq }
2202     { \seq_if_empty_p:N \g_@@_pos_of_xdots_seq }
2203     { \seq_if_empty_p:N \l_@@_empty_corner_cells_seq }
2204   }
2205   {
2206     \cs_set_eq:NN \@@_vline_i:nn \@@_vline_i_complete:nn
2207     \cs_set_eq:NN \@@_hline_i:nn \@@_hline_i_complete:nn
2208   }
2209   \bool_if:NT \l_@@_hlines_bool \@@_draw_hlines:
2210   \bool_if:NT \l_@@_vlines_bool \@@_draw_vlines:
2211   \g_@@_internal_code_after_tl
2212   \tl_gclear:N \g_@@_internal_code_after_tl

```

Now, the code-after.

```

2213   \bool_if:NT \c_@@_tikz_loaded_bool
2214   {
2215     \tikzset
2216     {
2217       every-picture / .style =
2218       {
2219         overlay ,
2220         remember-picture ,
2221         name-prefix = \@@_env: -
2222       }
2223     }
2224   }
2225   \cs_set_eq:NN \line \@@_line

```

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

```

2226   \cs_set_eq:NN \CodeAfter \prg_do_nothing:

```

And here's the code-after:

```

2227   \g_nicematrix_code_after_tl
2228   \tl_gclear:N \g_nicematrix_code_after_tl
2229   \group_end:

```

`\g_nicematrix_code_before_tl` is for instructions in the cells of the array such as `\rowcolor` and `\cellcolor` (when the key `colortbl-like` is in force). These instructions will be written on the aux file to be added to the code-before in the next run.

```

2230   \tl_if_empty:NF \g_nicematrix_code_before_tl
2231   {

```

The command `\rowcolor` in `tabular` will in fact use `\rectanglecolor` in order to follow the behaviour of `\rowcolor` of `colortbl`. That's why there may be a command `\rectanglecolor` in `\g_nicematrix_code_before_tl`. In order to avoid an error during the expansion, we define a protected version of `\rectanglecolor`.

```

2232     \cs_set_protected:Npn \rectanglecolor { }
2233     \cs_set_protected:Npn \columncolor { }
2234     \iow_now:Nn \@mainaux \ExplSyntaxOn
2235     \iow_now:Nx \@mainaux
2236     {
2237       \tl_gset:cn
2238       { g_@@_code_before_ \int_use:N \g_@@_env_int _ tl }
2239       { \g_nicematrix_code_before_tl }
2240     }
2241     \iow_now:Nn \@mainaux \ExplSyntaxOff
2242     \bool_set_true:N \l_@@_code_before_bool
2243   }

2244   \str_gclear:N \g_@@_name_env_str
2245   \@@_restore_iRow_jCol:

```

The command `\CT@arc@` contains the instruction of color for the rules of the array⁴³. This command is used by `\CT@arc@` but we use it also for compatibility with `colortbl`. But we want also to be able to use color for the rules of the array when `colortbl` is *not* loaded. That's why we do the following instruction which is in the patch of the end of arrays done by `colortbl`.

```
2246 \cs_gset_eq:NN \CT@arc@ \@@_old_CT@arc@
2247 }
```

We remind that the first mandatory argument of the command `\Block` is the size of the block with the special format $i-j$. However, the user is allowed to omit i or j (or both). This will be interpreted as: the last row (resp. column) of the block will be the last row (resp. column) of the block (without the potential exterior row—resp. column—of the array). By convention, this is stored in `\g_@@_pos_of_blocks_seq` (and `\g_@@_blocks_seq`) as a number of rows (resp. columns) for the block equal to 100. It's possible, after the construction of the array, to replace these values by the correct ones (since we know the number of rows and columns of the array).

```
2248 \cs_new_protected:Npn \@@_adjust_pos_of_blocks_seq:
2249 {
2250   \seq_gclear:N \g_tmpa_seq
2251   \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
2252     { \@@_adjust_pos_of_blocks_seq_i:nnnn #1 }
2253   \seq_gset_eq:NN \g_@@_pos_of_blocks_seq \g_tmpa_seq
2254 }
2255 \cs_new_protected:Npn \@@_adjust_pos_of_blocks_seq_i:nnnn #1 #2 #3 #4
2256 {
2257   \int_compare:nNnTF { #3 } > { 99 }
2258     { \int_set_eq:NN \l_tmpa_int \c@iRow }
2259     { \int_set:Nn \l_tmpa_int { #3 } }
2260   \int_compare:nNnTF { #4 } > { 99 }
2261     { \int_set_eq:NN \l_tmpb_int \c@jCol }
2262     { \int_set:Nn \l_tmpb_int { #4 } }
2263   \seq_gput_right:Nx \g_tmpa_seq
2264     { { #1 } { #2 } { \int_use:N \l_tmpa_int } { \int_use:N \l_tmpb_int } }
2265 }
```

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).

```
2266 \AtBeginDocument
2267 {
2268   \cs_new_protected:Npx \@@_draw_dotted_lines:
2269     {
2270       \c_@@_pgfortikzpicture_tl
2271       \@@_draw_dotted_lines_i:
2272       \c_@@_endpgfortikzpicture_tl
2273     }
2274 }
```

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

```
2275 \cs_new_protected:Npn \@@_draw_dotted_lines_i:
2276 {
2277   \pgfrememberpicturepositiononpagetrue
2278   \pgf@relevantforpicturesizefalse
2279   \g_@@_HVdotsfor_lines_tl
2280   \g_@@_Vdots_lines_tl
2281   \g_@@_Ddots_lines_tl
2282   \g_@@_Idots_lines_tl
2283   \g_@@_Cdots_lines_tl
2284   \g_@@_Ldots_lines_tl
2285 }
```

⁴³e.g. `\color[rgb]{0.5,0.5,0}`

```

2286 \cs_new_protected:Npn \@@_restore_iRow_jCol:
2287 {
2288   \cs_if_exist:NT \theiRow { \int_gset_eq:NN \c@iRow \l_@@_old_iRow_int }
2289   \cs_if_exist:NT \thejCol { \int_gset_eq:NN \c@jCol \l_@@_old_jCol_int }
2290 }

```

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.

```

2291 \cs_new_protected:Npn \@@_find_extremities_of_line:nnnn #1 #2 #3 #4
2292 {

```

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

```

2293   \cs_set:cpn { @@ _ dotted _ #1 - #2 } { }

```

Initialization of variables.

```

2294   \int_set:Nn \l_@@_initial_i_int { #1 }
2295   \int_set:Nn \l_@@_initial_j_int { #2 }
2296   \int_set:Nn \l_@@_final_i_int { #1 }
2297   \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.

```

2298   \bool_set_false:N \l_@@_stop_loop_bool
2299   \bool_do_until:Nn \l_@@_stop_loop_bool
2300   {
2301     \int_add:Nn \l_@@_final_i_int { #3 }
2302     \int_add:Nn \l_@@_final_j_int { #4 }

```

We test if we are still in the matrix.

```

2303     \bool_set_false:N \l_@@_final_open_bool
2304     \int_compare:nNnTF \l_@@_final_i_int > \c@iRow
2305     {
2306       \int_compare:nNnTF { #3 } = 1
2307       { \bool_set_true:N \l_@@_final_open_bool }
2308     }

```

```

2309         \int_compare:nNnT \l_@@_final_j_int > \c@jCol
2310         { \bool_set_true:N \l_@@_final_open_bool }
2311     }
2312 }
2313 {
2314     \int_compare:nNnTF \l_@@_final_j_int < 1
2315     {
2316         \int_compare:nNnT { #4 } = { -1 }
2317         { \bool_set_true:N \l_@@_final_open_bool }
2318     }
2319     {
2320         \int_compare:nNnT \l_@@_final_j_int > \c@jCol
2321         {
2322             \int_compare:nNnT { #4 } = 1
2323             { \bool_set_true:N \l_@@_final_open_bool }
2324         }
2325     }
2326 }
2327 \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.

```

2328     {

```

We do a step backwards.

```

2329         \int_sub:Nn \l_@@_final_i_int { #3 }
2330         \int_sub:Nn \l_@@_final_j_int { #4 }
2331         \bool_set_true:N \l_@@_stop_loop_bool
2332     }

```

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`.

```

2333     {
2334         \cs_if_exist:cTF
2335         {
2336             @@ _ dotted _
2337             \int_use:N \l_@@_final_i_int -
2338             \int_use:N \l_@@_final_j_int
2339         }
2340         {
2341             \int_sub:Nn \l_@@_final_i_int { #3 }
2342             \int_sub:Nn \l_@@_final_j_int { #4 }
2343             \bool_set_true:N \l_@@_final_open_bool
2344             \bool_set_true:N \l_@@_stop_loop_bool
2345         }
2346         {
2347             \cs_if_exist:cTF
2348             {
2349                 pgf @ sh @ ns @ \@@_env:
2350                 - \int_use:N \l_@@_final_i_int
2351                 - \int_use:N \l_@@_final_j_int
2352             }
2353             { \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.

```

2354         {
2355             \cs_set:cpn
2356             {
2357                 @@ _ dotted _
2358                 \int_use:N \l_@@_final_i_int -
2359                 \int_use:N \l_@@_final_j_int

```

```

2360     }
2361     { }
2362   }
2363 }
2364 }
2365 }

```

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

```

2366 \bool_set_false:N \l_@@_stop_loop_bool
2367 \bool_do_until:Nn \l_@@_stop_loop_bool
2368 {
2369   \int_sub:Nn \l_@@_initial_i_int { #3 }
2370   \int_sub:Nn \l_@@_initial_j_int { #4 }
2371   \bool_set_false:N \l_@@_initial_open_bool
2372   \int_compare:nNnTF \l_@@_initial_i_int < 1
2373   {
2374     \int_compare:nNnTF { #3 } = 1
2375     { \bool_set_true:N \l_@@_initial_open_bool }
2376     {
2377       \int_compare:nNnT \l_@@_initial_j_int = 0
2378       { \bool_set_true:N \l_@@_initial_open_bool }
2379     }
2380   }
2381   {
2382     \int_compare:nNnTF \l_@@_initial_j_int < 1
2383     {
2384       \int_compare:nNnT { #4 } = 1
2385       { \bool_set_true:N \l_@@_initial_open_bool }
2386     }
2387     {
2388       \int_compare:nNnT \l_@@_initial_j_int > \c@jCol
2389       {
2390         \int_compare:nNnT { #4 } = { -1 }
2391         { \bool_set_true:N \l_@@_initial_open_bool }
2392       }
2393     }
2394   }
2395   \bool_if:NTF \l_@@_initial_open_bool
2396   {
2397     \int_add:Nn \l_@@_initial_i_int { #3 }
2398     \int_add:Nn \l_@@_initial_j_int { #4 }
2399     \bool_set_true:N \l_@@_stop_loop_bool
2400   }
2401   {
2402     \cs_if_exist:cTF
2403     {
2404       @@ _ dotted _
2405       \int_use:N \l_@@_initial_i_int -
2406       \int_use:N \l_@@_initial_j_int
2407     }
2408     {
2409       \int_add:Nn \l_@@_initial_i_int { #3 }
2410       \int_add:Nn \l_@@_initial_j_int { #4 }
2411       \bool_set_true:N \l_@@_initial_open_bool
2412       \bool_set_true:N \l_@@_stop_loop_bool
2413     }
2414     {
2415       \cs_if_exist:cTF
2416       {
2417         pgf @ sh @ ns @ \@@_env:
2418         - \int_use:N \l_@@_initial_i_int

```

```

2419         - \int_use:N \l_@@_initial_j_int
2420     }
2421     { \bool_set_true:N \l_@@_stop_loop_bool }
2422     {
2423         \cs_set:cpn
2424         {
2425             @@_dotted_
2426             \int_use:N \l_@@_initial_i_int -
2427             \int_use:N \l_@@_initial_j_int
2428         }
2429         { }
2430     }
2431 }
2432 }
2433 }

```

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.

```

2434 \seq_gput_right:Nx \g_@@_pos_of_xdots_seq
2435 {
2436     { \int_use:N \l_@@_initial_i_int }
2437     { \int_use:N \l_@@_initial_j_int }
2438     { \int_use:N \l_@@_final_i_int }
2439     { \int_use:N \l_@@_final_j_int }
2440 }
2441 }

2442 \cs_new_protected:Npn \@@_set_initial_coords:
2443 {
2444     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2445     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2446 }
2447 \cs_new_protected:Npn \@@_set_final_coords:
2448 {
2449     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2450     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2451 }
2452 \cs_new_protected:Npn \@@_set_initial_coords_from_anchor:n #1
2453 {
2454     \pgfpointanchor
2455     {
2456         \@@_env:
2457         - \int_use:N \l_@@_initial_i_int
2458         - \int_use:N \l_@@_initial_j_int
2459     }
2460     { #1 }
2461     \@@_set_initial_coords:
2462 }
2463 \cs_new_protected:Npn \@@_set_final_coords_from_anchor:n #1
2464 {
2465     \pgfpointanchor
2466     {
2467         \@@_env:
2468         - \int_use:N \l_@@_final_i_int
2469         - \int_use:N \l_@@_final_j_int
2470     }
2471     { #1 }
2472     \@@_set_final_coords:
2473 }

```

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.

```

2474 \cs_new_protected:Npn \@@_draw_Ldots:nnn #1 #2 #3
2475 {
2476   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2477   {
2478     \@@_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.

```

2479     \group_begin:
2480     \int_compare:nNnTF { #1 } = 0
2481     { \color { nicematrix-first-row } }
2482     {

```

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.

```

2483         \int_compare:nNnT { #1 } = \l_@@_last_row_int
2484         { \color { nicematrix-last-row } }
2485     }
2486     \keys_set:nn { NiceMatrix / xdots } { #3 }
2487     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2488     \@@_actually_draw_Ldots:
2489     \group_end:
2490   }
2491 }

```

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`.

```

2492 \cs_new_protected:Npn \@@_actually_draw_Ldots:
2493 {
2494   \bool_if:NTF \l_@@_initial_open_bool
2495   {
2496     \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2497     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2498     \dim_add:Nn \l_@@_x_initial_dim \col@sep
2499     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int - base }
2500     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2501   }
2502   { \@@_set_initial_coords_from_anchor:n { base-east } }
2503   \bool_if:NTF \l_@@_final_open_bool
2504   {
2505     \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
2506     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2507     \dim_sub:Nn \l_@@_x_final_dim \col@sep
2508     \@@_qpoint:n { row - \int_use:N \l_@@_final_i_int - base }
2509     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2510   }
2511   { \@@_set_final_coords_from_anchor:n { base-west } }

```

We raise the line of a quantity equal to the radius of the dots because we want the dots really “on” the line of text. Of course, maybe we should not do that when the option `line-style` is used (?).

```

2512   \dim_add:Nn \l_@@_y_initial_dim \l_@@_radius_dim
2513   \dim_add:Nn \l_@@_y_final_dim \l_@@_radius_dim
2514   \@@_draw_line:
2515 }

```


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.

```

2516 \cs_new_protected:Npn \@@_draw_Cdots:nnn #1 #2 #3
2517 {
2518   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2519   {
2520     \@@_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.

```

2521   \group_begin:
2522     \int_compare:nNnTF { #1 } = 0
2523     { \color { nicematrix-first-row } }
2524     {

```

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.

```

2525       \int_compare:nNnT { #1 } = \l_@@_last_row_int
2526       { \color { nicematrix-last-row } }
2527     }
2528     \keys_set:nn { NiceMatrix / xdots } { #3 }
2529     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2530     \@@_actually_draw_Cdots:
2531   \group_end:
2532 }
2533 }

```

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`.

```

2534 \cs_new_protected:Npn \@@_actually_draw_Cdots:
2535 {
2536   \bool_if:NTF \l_@@_initial_open_bool
2537   {
2538     \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2539     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2540     \dim_add:Nn \l_@@_x_initial_dim \col@sep
2541   }
2542   { \@@_set_initial_coords_from_anchor:n { mid~east } }
2543   \bool_if:NTF \l_@@_final_open_bool
2544   {
2545     \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
2546     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2547     \dim_sub:Nn \l_@@_x_final_dim \col@sep
2548   }
2549   { \@@_set_final_coords_from_anchor:n { mid~west } }
2550   \bool_lazy_and:nnTF
2551     \l_@@_initial_open_bool
2552     \l_@@_final_open_bool
2553   {
2554     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2555     \dim_set_eq:NN \l_tmpa_dim \pgf@y
2556     \@@_qpoint:n { row - \@@_succ:n \l_@@_initial_i_int }
2557     \dim_set:Nn \l_@@_y_initial_dim { ( \l_tmpa_dim + \pgf@y ) / 2 }
2558     \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim

```

```

2559     }
2560     {
2561         \bool_if:NT \l_@@_initial_open_bool
2562         { \dim_set_eq:NN \l_@@_y_initial_dim \l_@@_y_final_dim }
2563         \bool_if:NT \l_@@_final_open_bool
2564         { \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim }
2565     }
2566     \@@_draw_line:
2567 }

```

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.

```

2568 \cs_new_protected:Npn \@@_draw_Vdots:nnn #1 #2 #3
2569 {
2570     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2571     {
2572         \@@_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.

```

2573         \group_begin:
2574         \int_compare:nNnTF { #2 } = 0
2575         { \color { nicematrix-first-col } }
2576         {
2577             \int_compare:nNnT { #2 } = \l_@@_last_col_int
2578             { \color { nicematrix-last-col } }
2579         }
2580         \keys_set:nn { NiceMatrix / xdots } { #3 }
2581         \tl_if_empty:VF \l_@@_xdots_color_tl
2582         { \color { \l_@@_xdots_color_tl } }
2583         \@@_actually_draw_Vdots:
2584     \group_end:
2585 }
2586 }

```

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`.

```

2587 \cs_new_protected:Npn \@@_actually_draw_Vdots:
2588 {

```

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

```

2589     \bool_set_false:N \l_tmpa_bool
2590     \bool_lazy_or:nnF \l_@@_initial_open_bool \l_@@_final_open_bool
2591     {
2592         \@@_set_initial_coords_from_anchor:n { south-west }
2593         \@@_set_final_coords_from_anchor:n { north-west }
2594         \bool_set:Nn \l_tmpa_bool
2595         { \dim_compare_p:nNn \l_@@_x_initial_dim = \l_@@_x_final_dim }
2596     }

```

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

```

2597 \bool_if:NTF \l_@@_initial_open_bool
2598 {
2599   \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2600   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2601 }
2602 { \@@_set_initial_coords_from_anchor:n { south } }
2603 \bool_if:NTF \l_@@_final_open_bool
2604 {
2605   \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }
2606   \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2607 }
2608 { \@@_set_final_coords_from_anchor:n { north } }
2609 \bool_if:NTF \l_@@_initial_open_bool
2610 {
2611   \bool_if:NTF \l_@@_final_open_bool
2612   {
2613     \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2614     \dim_set_eq:NN \l_tmpa_dim \pgf@x
2615     \@@_qpoint:n { col - \@@_succ:n \l_@@_initial_j_int }
2616     \dim_set:Nn \l_@@_x_initial_dim { ( \pgf@x + \l_tmpa_dim ) / 2 }
2617     \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.

```

2618 \int_compare:nNnT \l_@@_last_col_int > { -2 }
2619 {
2620   \int_compare:nNnT \l_@@_initial_j_int = \g_@@_col_total_int
2621   {
2622     \dim_set_eq:NN \l_tmpa_dim \l_@@_right_margin_dim
2623     \dim_add:Nn \l_tmpa_dim \l_@@_extra_right_margin_dim
2624     \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
2625     \dim_add:Nn \l_@@_x_final_dim \l_tmpa_dim
2626   }
2627 }
2628 }
2629 { \dim_set_eq:NN \l_@@_x_initial_dim \l_@@_x_final_dim }
2630 }
2631 {
2632   \bool_if:NTF \l_@@_final_open_bool
2633   { \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim }
2634   {

```

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.

```

2635 \dim_compare:nNnF \l_@@_x_initial_dim = \l_@@_x_final_dim
2636 {
2637   \dim_set:Nn \l_@@_x_initial_dim
2638   {
2639     \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
2640     \l_@@_x_initial_dim \l_@@_x_final_dim
2641   }
2642   \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim
2643 }
2644 }
2645 }
2646 \@@_draw_line:
2647 }

```

For the diagonal lines, the situation is a bit more complicated because, by default, we parallelize the diagonal 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.

```

2648 \cs_new_protected:Npn \@@_draw_Ddots:nnn #1 #2 #3
2649 {
2650   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2651   {
2652     \@@_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.

```

2653     \group_begin:
2654       \keys_set:nn { NiceMatrix / xdots } { #3 }
2655       \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2656       \@@_actually_draw_Ddots:
2657     \group_end:
2658   }
2659 }

```

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`.

```

2660 \cs_new_protected:Npn \@@_actually_draw_Ddots:
2661 {
2662   \bool_if:NTF \l_@@_initial_open_bool
2663   {
2664     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2665     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2666     \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2667     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2668   }
2669   { \@@_set_initial_coords_from_anchor:n { south-east } }
2670   \bool_if:NTF \l_@@_final_open_bool
2671   {
2672     \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }
2673     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2674     \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
2675     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2676   }
2677   { \@@_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.

```

2678   \bool_if:NT \l_@@_parallelize_diags_bool
2679   {
2680     \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).

```

2681     \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.

```

2682     {
2683       \dim_gset:Nn \g_@@_delta_x_one_dim

```

```

2684         { \l_@@_x_final_dim - \l_@@_x_initial_dim }
2685         \dim_gset:Nn \g_@@_delta_y_one_dim
2686         { \l_@@_y_final_dim - \l_@@_y_initial_dim }
2687     }

```

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`.

```

2688     {
2689         \dim_set:Nn \l_@@_y_final_dim
2690         {
2691             \l_@@_y_initial_dim +
2692             ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2693             \dim_ratio:nn \g_@@_delta_y_one_dim \g_@@_delta_x_one_dim
2694         }
2695     }
2696 }
2697 \@@_draw_line:
2698 }

```

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.

```

2699 \cs_new_protected:Npn \@@_draw_Iddots:nnn #1 #2 #3
2700 {
2701     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2702     {
2703         \@@_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.

```

2704         \group_begin:
2705         \keys_set:nn { NiceMatrix / xdots } { #3 }
2706         \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2707         \@@_actually_draw_Iddots:
2708         \group_end:
2709     }
2710 }

```

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`.

```

2711 \cs_new_protected:Npn \@@_actually_draw_Iddots:
2712 {
2713     \bool_if:NTF \l_@@_initial_open_bool
2714     {
2715         \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2716         \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2717         \@@_qpoint:n { col - \@@_succ:n \l_@@_initial_j_int }
2718         \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2719     }
2720     { \@@_set_initial_coords_from_anchor:n { south-west } }
2721     \bool_if:NTF \l_@@_final_open_bool
2722     {
2723         \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }

```

```

2724     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2725     \@@_qpoint:n { col - \int_use:N \l_@@_final_j_int }
2726     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2727   }
2728   { \@@_set_final_coords_from_anchor:n { north-east } }
2729   \bool_if:NT \l_@@_parallelize_diags_bool
2730   {
2731     \int_gincr:N \g_@@_iddots_int
2732     \int_compare:nNnTF \g_@@_iddots_int = 1
2733     {
2734       \dim_gset:Nn \g_@@_delta_x_two_dim
2735       { \l_@@_x_final_dim - \l_@@_x_initial_dim }
2736       \dim_gset:Nn \g_@@_delta_y_two_dim
2737       { \l_@@_y_final_dim - \l_@@_y_initial_dim }
2738     }
2739     {
2740       \dim_set:Nn \l_@@_y_final_dim
2741       {
2742         \l_@@_y_initial_dim +
2743         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2744         \dim_ratio:nn \g_@@_delta_y_two_dim \g_@@_delta_x_two_dim
2745       }
2746     }
2747   }
2748   \@@_draw_line:
2749 }

```

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`

```

2750 \cs_new_protected:Npn \@@_draw_line:
2751 {
2752   \pgfrememberpicturepositiononpagetrue
2753   \pgf@relevantforpicturesizefalse
2754   \tl_if_eq:NNTF \l_@@_xdots_line_style_tl \c_@@_standard_tl
2755     \@@_draw_standard_dotted_line:
2756     \@@_draw_non_standard_dotted_line:
2757 }

```

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.

```

2758 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:
2759 {
2760   \begin { scope }
2761   \exp_args:No \@@_draw_non_standard_dotted_line:n
2762     { \l_@@_xdots_line_style_tl , \l_@@_xdots_color_tl }
2763 }

```

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.

```

2764 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:n #1
2765 {
2766   \draw
2767   [
2768     #1 ,
2769     shorten~> = \l_@@_xdots_shorten_dim ,
2770     shorten~< = \l_@@_xdots_shorten_dim ,
2771   ]
2772   ( \l_@@_x_initial_dim , \l_@@_y_initial_dim )
2773   -- node [ sloped , above ]
2774     { \c_math_toggle_token \scriptstyle \l_@@_xdots_up_tl \c_math_toggle_token }
2775   node [ sloped , below ]
2776     {
2777       \c_math_toggle_token
2778       \scriptstyle \l_@@_xdots_down_tl
2779       \c_math_toggle_token
2780     }
2781   ( \l_@@_x_final_dim , \l_@@_y_final_dim ) ;
2782   \end { scope }
2783 }
```

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

```

2784 \cs_new_protected:Npn \@@_draw_standard_dotted_line:
2785 {
```

First, we put the labels.

```

2786   \bool_lazy_and:nnF
2787     { \tl_if_empty_p:N \l_@@_xdots_up_tl }
2788     { \tl_if_empty_p:N \l_@@_xdots_down_tl }
2789   {
2790     \pgfscope
2791     \pgftransformshift
2792     {
2793       \pgfpointlineattime { 0.5 }
2794       { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
2795       { \pgfpoint \l_@@_x_final_dim \l_@@_y_final_dim }
2796     }
2797     \pgftransformrotate
2798     {
2799       \fp_eval:n
2800       {
2801         atand
2802         (
2803           \l_@@_y_final_dim - \l_@@_y_initial_dim ,
2804           \l_@@_x_final_dim - \l_@@_x_initial_dim
2805         )
2806       }
2807     }
2808     \pgfnode
2809     { rectangle }
2810     { south }
2811     {
2812       \c_math_toggle_token
2813       \scriptstyle \l_@@_xdots_up_tl
2814       \c_math_toggle_token
2815     }
2816     { }
2817     { \pgfusepath { } }
2818     \pgfnode
```

```

2819     { rectangle }
2820     { north }
2821     {
2822         \c_math_toggle_token
2823         \scriptstyle \l_@@_xdots_down_tl
2824         \c_math_toggle_token
2825     }
2826     { }
2827     { \pgfusepath { } }
2828 \endpgfscope
2829 }
2830 \pgfrememberpicturepositiononpagetrue
2831 \pgf@relevantforpicturesizefalse
2832 \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.

```

2833 \dim_zero_new:N \l_@@_l_dim
2834 \dim_set:Nn \l_@@_l_dim
2835 {
2836     \fp_to_dim:n
2837     {
2838         sqrt
2839         (
2840             ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) ^ 2
2841             +
2842             ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) ^ 2
2843         )
2844     }
2845 }

```

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.

```

2846 \bool_lazy_or:nnF
2847 { \dim_compare_p:nNn { \dim_abs:n \l_@@_l_dim } > \c_@@_max_l_dim }
2848 { \dim_compare_p:nNn \l_@@_l_dim = \c_zero_dim }
2849 \@@_draw_standard_dotted_line_i:
2850 \group_end:
2851 }
2852 \dim_const:Nn \c_@@_max_l_dim { 50 cm }
2853 \cs_new_protected:Npn \@@_draw_standard_dotted_line_i:
2854 {

```

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

```

2855 \bool_if:NTF \l_@@_initial_open_bool
2856 {
2857     \bool_if:NTF \l_@@_final_open_bool
2858     {
2859         \int_set:Nn \l_tmpa_int
2860         { \dim_ratio:nn \l_@@_l_dim \l_@@_inter_dots_dim }
2861     }
2862     {
2863         \int_set:Nn \l_tmpa_int
2864         {
2865             \dim_ratio:nn
2866             { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
2867             \l_@@_inter_dots_dim
2868         }
2869     }
2870 }
2871 {

```



```

2872 \bool_if:NTF \l_@@_final_open_bool
2873 {
2874   \int_set:Nn \l_tmpa_int
2875   {
2876     \dim_ratio:nn
2877     { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
2878     \l_@@_inter_dots_dim
2879   }
2880 }
2881 {
2882   \int_set:Nn \l_tmpa_int
2883   {
2884     \dim_ratio:nn
2885     { \l_@@_l_dim - 2 \l_@@_xdots_shorten_dim }
2886     \l_@@_inter_dots_dim
2887   }
2888 }
2889 }

```

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

```

2890 \dim_set:Nn \l_tmpa_dim
2891 {
2892   ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2893   \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
2894 }
2895 \dim_set:Nn \l_tmpb_dim
2896 {
2897   ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
2898   \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
2899 }

```

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`.

```

2900 \int_set:Nn \l_tmpb_int
2901 {
2902   \bool_if:NTF \l_@@_initial_open_bool
2903   { \bool_if:NTF \l_@@_final_open_bool 1 0 }
2904   { \bool_if:NTF \l_@@_final_open_bool 2 1 }
2905 }

```

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.

```

2906 \dim_gadd:Nn \l_@@_x_initial_dim
2907 {
2908   ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2909   \dim_ratio:nn
2910   { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
2911   { 2 \l_@@_l_dim }
2912   * \l_tmpb_int
2913 }
2914 \dim_gadd:Nn \l_@@_y_initial_dim
2915 {
2916   ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
2917   \dim_ratio:nn
2918   { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
2919   { 2 \l_@@_l_dim }
2920   * \l_tmpb_int
2921 }
2922 \pgf@relevantforpicturesizefalse
2923 \int_step_inline:nnn 0 \l_tmpa_int
2924 {

```

```

2925 \pgfpathcircle
2926 { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
2927 { \l_@@_radius_dim }
2928 \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
2929 \dim_add:Nn \l_@@_y_initial_dim \l_tmpb_dim
2930 }
2931 \pgfusepathqfill
2932 }

```

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.

```

2933 \AtBeginDocument
2934 {
2935   \tl_set:Nn \l_@@_argspec_tl { 0 { } E { _ ^ } { { } { } } }
2936   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
2937   \exp_args:NNV \NewDocumentCommand \@@_Ldots \l_@@_argspec_tl
2938   {
2939     \int_compare:nNnTF \c@jCol = 0
2940     { \@@_error:nn { in~first~col } \Ldots }
2941     {
2942       \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
2943       { \@@_error:nn { in~last~col } \Ldots }
2944       {
2945         \@@_instruction_of_type:nnn \c_false_bool { \Ldots }
2946         { #1 , down = #2 , up = #3 }
2947       }
2948     }
2949     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_ldots }
2950     \bool_gset_true:N \g_@@_empty_cell_bool
2951   }

2952   \exp_args:NNV \NewDocumentCommand \@@_Cdots \l_@@_argspec_tl
2953   {
2954     \int_compare:nNnTF \c@jCol = 0
2955     { \@@_error:nn { in~first~col } \Cdots }
2956     {
2957       \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
2958       { \@@_error:nn { in~last~col } \Cdots }
2959       {
2960         \@@_instruction_of_type:nnn \c_false_bool { \Cdots }
2961         { #1 , down = #2 , up = #3 }
2962       }
2963     }
2964     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_cdots }
2965     \bool_gset_true:N \g_@@_empty_cell_bool
2966   }

```

```

2967 \exp_args:NNV \NewDocumentCommand \@@_Vdots \l_@@_argspec_tl
2968 {
2969   \int_compare:nNnTF \c@iRow = 0
2970   { \@@_error:nn { in~first~row } \Vdots }
2971   {
2972     \int_compare:nNnTF \c@iRow = \l_@@_last_row_int
2973     { \@@_error:nn { in~last~row } \Vdots }
2974     {
2975       \@@_instruction_of_type:nnn \c_false_bool { Vdots }
2976       { #1 , down = #2 , up = #3 }
2977     }
2978   }
2979   \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_vdots }
2980   \bool_gset_true:N \g_@@_empty_cell_bool
2981 }

2982 \exp_args:NNV \NewDocumentCommand \@@_Ddots \l_@@_argspec_tl
2983 {
2984   \int_case:nnF \c@iRow
2985   {
2986     0 { \@@_error:nn { in~first~row } \Ddots }
2987     \l_@@_last_row_int { \@@_error:nn { in~last~row } \Ddots }
2988   }
2989   {
2990     \int_case:nnF \c@jCol
2991     {
2992       0 { \@@_error:nn { in~first~col } \Ddots }
2993       \l_@@_last_col_int { \@@_error:nn { in~last~col } \Ddots }
2994     }
2995     {
2996       \keys_set_known:nn { NiceMatrix / Ddots } { #1 }
2997       \@@_instruction_of_type:nnn \l_@@_draw_first_bool { Ddots }
2998       { #1 , down = #2 , up = #3 }
2999     }
3000   }
3001 }
3002 \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_ddots }
3003 \bool_gset_true:N \g_@@_empty_cell_bool
3004 }

3005 \exp_args:NNV \NewDocumentCommand \@@_Iddots \l_@@_argspec_tl
3006 {
3007   \int_case:nnF \c@iRow
3008   {
3009     0 { \@@_error:nn { in~first~row } \Iddots }
3010     \l_@@_last_row_int { \@@_error:nn { in~last~row } \Iddots }
3011   }
3012   {
3013     \int_case:nnF \c@jCol
3014     {
3015       0 { \@@_error:nn { in~first~col } \Iddots }
3016       \l_@@_last_col_int { \@@_error:nn { in~last~col } \Iddots }
3017     }
3018     {
3019       \keys_set_known:nn { NiceMatrix / Ddots } { #1 }
3020       \@@_instruction_of_type:nnn \l_@@_draw_first_bool { Iddots }
3021       { #1 , down = #2 , up = #3 }
3022     }
3023   }
3024 }
3025 \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_iddots }
3026 \bool_gset_true:N \g_@@_empty_cell_bool

```

```

3027     }
3028 }

```

End of the \AtBeginDocument.

Despite its name, the following set of keys will be used for \Ddots but also for \Iddots.

```

3029 \keys_define:nn { NiceMatrix / Ddots }
3030 {
3031   draw-first .bool_set:N = \l_@@_draw_first_bool ,
3032   draw-first .default:n = true ,
3033   draw-first .value_forbidden:n = true
3034 }

```

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

```

3035 \cs_new_protected:Npn \@@_Hspace:
3036 {
3037   \bool_gset_true:N \g_@@_empty_cell_bool
3038   \hspace
3039 }

```

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

```

3040 \cs_set_eq:NN \@@_old_multicolumn \multicolumn
3041 \cs_new:Npn \@@_multicolumn:nnn #1 #2 #3
3042 {
3043   % \begin{macrocode}
3044   % We have to act in an expandable way since it will begin by a |\multicolumn|.
3045   % \end{macrocode}
3046   \exp_args:NNe
3047   \@@_old_multicolumn
3048   { #1 }

```

We will have to replace \tl_lower_case:n in the future since it seems to be deprecated.

```

3049 {
3050   \exp_args:Ne \str_case:nn { \tl_lower_case:n { #2 } }
3051   {
3052     l { > \@@_Cell: l < \@@_end_Cell: }
3053     r { > \@@_Cell: r < \@@_end_Cell: }
3054     c { > \@@_Cell: c < \@@_end_Cell: }
3055     { l | } { > \@@_Cell: l < \@@_end_Cell: | }
3056     { r | } { > \@@_Cell: r < \@@_end_Cell: | }
3057     { c | } { > \@@_Cell: c < \@@_end_Cell: | }
3058     { | l } { | > \@@_Cell: l < \@@_end_Cell: }
3059     { | r } { | > \@@_Cell: r < \@@_end_Cell: }
3060     { | c } { | > \@@_Cell: c < \@@_end_Cell: }
3061     { | l | } { | > \@@_Cell: l < \@@_end_Cell: | }
3062     { | r | } { | > \@@_Cell: r < \@@_end_Cell: | }
3063     { | c | } { | > \@@_Cell: c < \@@_end_Cell: | }
3064   }
3065 }
3066 { #3 }

```

The \peek_remove_spaces:n is mandatory.

```

3067 \peek_remove_spaces:n
3068 {
3069   \int_compare:nNnT #1 > 1
3070   {
3071     \seq_gput_left:Nx \g_@@_multicolumn_cells_seq
3072     { \int_use:N \c@iRow - \int_use:N \c@jCol }
3073     \seq_gput_left:Nn \g_@@_multicolumn_sizes_seq { #1 }
3074     \seq_gput_right:Nx \g_@@_pos_of_blocks_seq
3075     {
3076       { \int_use:N \c@iRow }

```

```

3077         { \int_use:N \c@jCol }
3078         { \int_use:N \c@iRow }
3079         { \int_eval:n { \c@jCol + #1 - 1 } }
3080     }
3081 }
3082 \int_gadd:Nn \c@jCol { #1 - 1 }
3083 \int_compare:nNnT \c@jCol > \g_@@_col_total_int
3084 { \int_gset_eq:NN \g_@@_col_total_int \c@jCol }
3085 }
3086 }

```

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`.

```

3087 \cs_new:Npn \@@_Hdotsfor:
3088 {
3089     \int_compare:nNnTF \c@jCol = 0
3090     { \@@_error:n { Hdotsfor~in~col~0 } }
3091     {
3092         \multicolumn { 1 } { c } { }
3093         \@@_Hdotsfor_i
3094     }
3095 }

```

The command `\@@_Hdotsfor_i` is defined with `\NewDocumentCommand` because it has an optional argument. Note that such a command defined by `\NewDocumentCommand` is protected and that's why we have put the `\multicolumn` before (in the definition of `\@@_Hdotsfor:`).

```

3096 \AtBeginDocument
3097 {
3098     \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
3099     \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.

```

3100     \exp_args:NNV \NewDocumentCommand \@@_Hdotsfor_i \l_@@_argspec_tl
3101     {
3102         \tl_gput_right:Nx \g_@@_HVDotsfor_lines_tl
3103         {
3104             \@@_Hdotsfor:nnnn
3105             { \int_use:N \c@iRow }
3106             { \int_use:N \c@jCol }
3107             { #2 }
3108             {
3109                 #1 , #3 ,
3110                 down = \exp_not:n { #4 } , up = \exp_not:n { #5 }
3111             }
3112         }
3113         \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { c } { } }
3114     }
3115 }

```

Enf of `\AtBeginDocument`.

```

3116 \cs_new_protected:Npn \@@_Hdotsfor:nnnn #1 #2 #3 #4
3117 {
3118     \bool_set_false:N \l_@@_initial_open_bool
3119     \bool_set_false:N \l_@@_final_open_bool

```

For the row, it's easy.

```

3120     \int_set:Nn \l_@@_initial_i_int { #1 }
3121     \int_set_eq:NN \l_@@_final_i_int \l_@@_initial_i_int

```

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

```

3122 \int_compare:nNnTF #2 = 1
3123 {
3124   \int_set:Nn \l_@@_initial_j_int 1
3125   \bool_set_true:N \l_@@_initial_open_bool
3126 }
3127 {
3128   \cs_if_exist:cTF
3129   {
3130     pgf @ sh @ ns @ \@@_env:
3131     - \int_use:N \l_@@_initial_i_int
3132     - \int_eval:n { #2 - 1 }
3133   }
3134   { \int_set:Nn \l_@@_initial_j_int { #2 - 1 } }
3135   {
3136     \int_set:Nn \l_@@_initial_j_int { #2 }
3137     \bool_set_true:N \l_@@_initial_open_bool
3138   }
3139 }
3140 \int_compare:nNnTF { #2 + #3 - 1 } = \c@jCol
3141 {
3142   \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
3143   \bool_set_true:N \l_@@_final_open_bool
3144 }
3145 {
3146   \cs_if_exist:cTF
3147   {
3148     pgf @ sh @ ns @ \@@_env:
3149     - \int_use:N \l_@@_final_i_int
3150     - \int_eval:n { #2 + #3 }
3151   }
3152   { \int_set:Nn \l_@@_final_j_int { #2 + #3 } }
3153   {
3154     \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
3155     \bool_set_true:N \l_@@_final_open_bool
3156   }
3157 }
3158 \group_begin:
3159 \int_compare:nNnTF { #1 } = 0
3160 { \color { nicematrix-first-row } }
3161 {
3162   \int_compare:nNnT { #1 } = \g_@@_row_total_int
3163   { \color { nicematrix-last-row } }
3164 }
3165 \keys_set:nn { NiceMatrix / xdots } { #4 }
3166 \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3167 \@@_actually_draw_Ldots:
3168 \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).

```

3169 \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
3170 { \cs_set:cpn { @@ _ dotted _ #1 - ##1 } { } }
3171 }

3172 \AtBeginDocument
3173 {
3174   \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
3175   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
3176   \exp_args:NNV \NewDocumentCommand \@@_Vdotsfor: \l_@@_argspec_tl
3177   {

```

```

3178 \tl_gput_right:Nx \g_@@_HVdotsfor_lines_tl
3179 {
3180   \@@_Vdotsfor:nnnn
3181   { \int_use:N \c@iRow }
3182   { \int_use:N \c@jCol }
3183   { #2 }
3184   {
3185     #1 , #3 ,
3186     down = \exp_not:n { #4 } , up = \exp_not:n { #5 }
3187   }
3188 }
3189 }
3190 }

```

Enf of \AtBeginDocument.

```

3191 \cs_new_protected:Npn \@@_Vdotsfor:nnnn #1 #2 #3 #4
3192 {
3193   \bool_set_false:N \l_@@_initial_open_bool
3194   \bool_set_false:N \l_@@_final_open_bool

```

For the column, it's easy.

```

3195 \int_set:Nn \l_@@_initial_j_int { #2 }
3196 \int_set_eq:NN \l_@@_final_j_int \l_@@_initial_j_int

```

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

```

3197 \int_compare:nNnTF #1 = 1
3198 {
3199   \int_set:Nn \l_@@_initial_i_int 1
3200   \bool_set_true:N \l_@@_initial_open_bool
3201 }
3202 {
3203   \cs_if_exist:cTF
3204   {
3205     pgf @ sh @ ns @ \@@_env:
3206     - \int_eval:n { #1 - 1 }
3207     - \int_use:N \l_@@_initial_j_int
3208   }
3209   { \int_set:Nn \l_@@_initial_i_int { #1 - 1 } }
3210   {
3211     \int_set:Nn \l_@@_initial_i_int { #1 }
3212     \bool_set_true:N \l_@@_initial_open_bool
3213   }
3214 }
3215 \int_compare:nNnTF { #1 + #3 - 1 } = \c@iRow
3216 {
3217   \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }
3218   \bool_set_true:N \l_@@_final_open_bool
3219 }
3220 {
3221   \cs_if_exist:cTF
3222   {
3223     pgf @ sh @ ns @ \@@_env:
3224     - \int_eval:n { #1 + #3 }
3225     - \int_use:N \l_@@_final_j_int
3226   }
3227   { \int_set:Nn \l_@@_final_i_int { #1 + #3 } }
3228   {
3229     \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }
3230     \bool_set_true:N \l_@@_final_open_bool
3231   }
3232 }
3233 \group_begin:
3234 \int_compare:nNnTF { #2 } = 0
3235 { \color { nicematrix-first-col } }

```

```

3236 {
3237   \int_compare:nNnT { #2 } = \g_@@_col_total_int
3238   { \color { nicematrix-last-col } }
3239 }
3240 \keys_set:nn { NiceMatrix / xdots } { #4 }
3241 \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3242 \@@_actually_draw_Vdots:
3243 \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).

```

3244 \int_step_inline:nnn { #1 } { #1 + #3 - 1 }
3245 { \cs_set:cpn { @@ _ dotted _ ##1 - #2 } { } }
3246 }

```

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

```

3247 \cs_new_protected:Npn \@@_rotate: { \bool_gset_true:N \g_@@_rotate_bool }

```

The command `\line` accessible in code-after

In the `code-after`, the command `\@@_line:nn` will be linked to `\line`. This command takes two arguments which are the specifications of two cells in the array (in the format i - j) and draws a dotted line between these cells.

First, we write a command with an argument of the format i - j and applies the command `\int_eval:n` to i and j ; this must *not* be protected (and is, of course fully expandable).⁴⁴

```

3248 \cs_new:Npn \@@_double_int_eval:n #1-#2 \q_stop
3249 { \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).

```

3250 \AtBeginDocument
3251 {
3252   \tl_set:Nn \l_@@_argspec_tl { 0 { } m m ! 0 { } E { _ ^ } { { } { } } }
3253   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
3254   \exp_args:NNV \NewDocumentCommand \@@_line \l_@@_argspec_tl
3255   {
3256     \group_begin:
3257     \keys_set:nn { NiceMatrix / xdots } { #1 , #4 , down = #5 , up = #6 }
3258     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
3259     \use:e
3260     {
3261       \@@_line_i:nn
3262       { \@@_double_int_eval:n #2 \q_stop }
3263       { \@@_double_int_eval:n #3 \q_stop }
3264     }
3265     \group_end:
3266   }
3267 }
3268 \cs_new_protected:Npn \@@_line_i:nn #1 #2
3269 {
3270   \bool_set_false:N \l_@@_initial_open_bool
3271   \bool_set_false:N \l_@@_final_open_bool

```

⁴⁴Indeed, we want that the user may use the command `\line` in `code-after` with LaTeX counters in the arguments — with the command `\value`.


```

3272 \bool_if:nTF
3273 {
3274   \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #1 }
3275   ||
3276   \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #2 }
3277 }
3278 {
3279   \@@_error:nnn { unknown~cell~for~line~in~code~after } { #1 } { #2 }
3280 }
3281 { \@@_draw_line_ii:nn { #1 } { #2 } }
3282 }
3283 \AtBeginDocument
3284 {
3285   \cs_new_protected:Npx \@@_draw_line_ii:nn #1 #2
3286   {

```

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:`.

```

3287   \c_@@_pgfortikzpicture_tl
3288   \@@_draw_line_iii:nn { #1 } { #2 }
3289   \c_@@_endpgfortikzpicture_tl
3290 }
3291 }

```

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

```

3292 \cs_new_protected:Npn \@@_draw_line_iii:nn #1 #2
3293 {
3294   \pgfrememberpicturepositiononpagetrue
3295   \pgfpointshapeborder { \@@_env: - #1 } { \@@_qpoint:n { #2 } }
3296   \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
3297   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
3298   \pgfpointshapeborder { \@@_env: - #2 } { \@@_qpoint:n { #1 } }
3299   \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
3300   \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
3301   \@@_draw_line:
3302 }

```

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

Colors of cells, rows and columns

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

```

3303 \cs_set_protected:Npn \@@_cut_on_hyphen:w #1-#2\q_stop
3304 {
3305   \tl_set:Nn \l_tmpa_tl { #1 }
3306   \tl_set:Nn \l_tmpb_tl { #2 }
3307 }

```

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

```

3308 \NewDocumentCommand \@@_rowcolor { 0 { } m m }
3309 {
3310   \tl_if_blank:nF { #2 }
3311   {
3312     \pgfpicture
3313     \pgf@relevantforpicturesizefalse
3314     \tl_if_empty:nTF { #1 } \color { \color [ #1 ] } { #2 }

```

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

```

3315 \@@_qpoint:n { col - 1 }
3316 \int_compare:nNnTF \l_@@_first_col_int = 0
3317 { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
3318 { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
3319 \@@_qpoint:n { col - \@@_succ:n \c@jCol }
3320 \dim_set:Nn \l_tmpa_dim { \pgf@x + 0.5 \arrayrulewidth }
3321 \clist_map_inline:nn { #3 }
3322 {
3323   \tl_set:Nn \l_tmpa_tl { ##1 }
3324   \tl_if_in:NnTF \l_tmpa_tl { - }
3325   { \@@_cut_on_hyphen:w ##1 \q_stop }
3326   { \@@_cut_on_hyphen:w ##1 - ##1 \q_stop }
3327   \tl_if_empty:NT \l_tmpa_tl { \tl_set:Nn \l_tmpa_tl { 1 } }
3328   \tl_if_empty:NT \l_tmpb_tl
3329   { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@iRow } }
3330   \int_compare:nNnT \l_tmpb_tl > \c@iRow
3331   { \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`.

```

3332 \@@_qpoint:n { row - \@@_succ:n \l_tmpb_tl }
3333 \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
3334 \@@_qpoint:n { row - \l_tmpa_tl }
3335 \dim_set:Nn \l_tmpd_dim { \pgf@y + 0.5 \arrayrulewidth }
3336 \pgfpathrectanglecorners
3337 { \pgfpoint \l_tmpc_dim \l_tmpd_dim }
3338 { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3339 }
3340 \pgfusepathqfill
3341 \endpgfpicture
3342 }
3343 }

```

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

```

3344 \NewDocumentCommand \@@_columncolor { 0 { } m m }
3345 {
3346   \tl_if_blank:nF { #2 }
3347   {
3348     \pgfpicture
3349     \pgf@relevantforpicturesizefalse
3350     \tl_if_empty:nTF { #1 } \color { \color [ #1 ] } { #2 }
3351     \@@_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.

```

3352 \dim_set:Nn \l_tmpa_dim { \pgf@y + 0.5 \arrayrulewidth }
3353 \@@_qpoint:n { row - \@@_succ:n \c@iRow }
3354 \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
3355 \clist_map_inline:nn { #3 }
3356 {
3357   \tl_set:Nn \l_tmpa_tl { ##1 }
3358   \tl_if_in:NnTF \l_tmpa_tl { - }
3359   { \@@_cut_on_hyphen:w ##1 \q_stop }
3360   { \@@_cut_on_hyphen:w ##1 - ##1 \q_stop }
3361   \tl_if_empty:NT \l_tmpa_tl { \tl_set:Nn \l_tmpa_tl { 1 } }
3362   \tl_if_empty:NT \l_tmpb_tl
3363   { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
3364   \int_compare:nNnT \l_tmpb_tl > \c@jCol
3365   { \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`.

```

3366 \@@_qpoint:n { col - \l_tmpa_tl }
3367 \int_compare:nNnTF \l_@@_first_col_int = \l_tmpa_tl
3368 { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
3369 { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }

```

```

3370         \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
3371         \dim_set:Nn \l_tmpd_dim { \pgf@x + 0.5 \arrayrulewidth }
3372         \pgfpathrectanglecorners
3373         { \pgfpoint \l_tmpc_dim \l_tmpa_dim }
3374         { \pgfpoint \l_tmpd_dim \l_tmpb_dim }
3375     }
3376     \pgfusepathqfill
3377     \endpgfpicture
3378 }
3379 }

```

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

```

3380 \NewDocumentCommand \@@_cellcolor { 0 { } m m }
3381 {
3382     \tl_if_blank:nF { #2 }
3383     {
3384         \pgfpicture
3385         \pgf@relevantforpicturesizefalse
3386         \tl_if_empty:nTF { #1 } \color { \color [ #1 ] } { #2 }
3387         \clist_map_inline:nn { #3 }
3388         {
3389             \@@_cut_on_hyphen:w ##1 \q_stop
3390             \@@_qpoint:n { row - \l_tmpa_tl }
3391             \bool_lazy_and:nnT
3392             { \int_compare_p:n { \l_tmpa_tl <= \c@iRow } }
3393             { \int_compare_p:n { \l_tmpb_tl <= \c@jCol } }
3394             {
3395                 \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
3396                 \@@_qpoint:n { row - \@@_succ:n \l_tmpa_tl }
3397                 \dim_set:Nn \l_tmpa_dim { \pgf@y + 0.5 \arrayrulewidth }
3398                 \@@_qpoint:n { col - \l_tmpb_tl }
3399                 \int_compare:nNnTF \l_@@_first_col_int = \l_tmpb_tl
3400                 { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
3401                 { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
3402                 \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
3403                 \dim_set:Nn \l_tmpd_dim { \pgf@x + 0.5 \arrayrulewidth }
3404                 \pgfpathrectanglecorners
3405                 { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
3406                 { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
3407             }
3408         }
3409         \pgfusepathqfill
3410         \endpgfpicture
3411     }
3412 }

```

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

```

3413 \NewDocumentCommand \@@_rectanglecolor { 0 { } m m m }
3414 {
3415     \tl_if_blank:nF { #2 }
3416     {
3417         \pgfpicture
3418         \pgf@relevantforpicturesizefalse
3419         \tl_if_empty:nTF { #1 } \color { \color [ #1 ] } { #2 }
3420         \@@_cut_on_hyphen:w #3 \q_stop
3421         \bool_lazy_and:nnT
3422         { \int_compare_p:n { \l_tmpa_tl <= \c@iRow } }
3423         { \int_compare_p:n { \l_tmpb_tl <= \c@jCol } }
3424         {
3425             \@@_qpoint:n { row - \l_tmpa_tl }
3426             \dim_set:Nn \l_tmpb_dim { \pgf@y + 0.5 \arrayrulewidth }
3427             \@@_qpoint:n { col - \l_tmpb_tl }

```

```

3428 \int_compare:nNnTF \l_@@_first_col_int = \l_tmpb_tl
3429 { \dim_set:Nn \l_tmpc_dim { \pgf@x - 0.5 \arrayrulewidth } }
3430 { \dim_set:Nn \l_tmpc_dim { \pgf@x + 0.5 \arrayrulewidth } }
3431 \@@_cut_on_hyphen:w #4 \q_stop
3432 \int_compare:nNnT \l_tmpa_tl > \c@iRow
3433 { \tl_set:Nx \l_tmpa_tl { \int_use:N \c@iRow } }
3434 \int_compare:nNnT \l_tmpb_tl > \c@jCol
3435 { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
3436 \@@_qpoint:n { row - \@@_succ:n \l_tmpa_tl }
3437 \dim_set:Nn \l_tmpa_dim { \pgf@y + 0.5 \arrayrulewidth }
3438 \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
3439 \dim_set:Nn \l_tmpd_dim { \pgf@x + 0.5 \arrayrulewidth }
3440 \pgfpathrectanglecorners
3441 { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
3442 { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
3443 \pgfusepathqfill
3444 }
3445 \endpgfpicture
3446 }
3447 }

```

The command `\rowcolors` (accessible in the code-before) is inspired by the command `\rowcolors` of the package `xcolor` (with the option `table`). However, the command `\rowcolors` of `nicematrix` has *not* the optional argument of the command `\rowcolors` of `xcolor`. Here is an example: `\rowcolors{1}{blue!10}{}[respect-blocks]`. The last optional argument is for options. As of now, there is only one key available : `respect-blocks`.

```

3448 \keys_define:nn { NiceMatrix / rowcolors }
3449 {
3450   respect-blocks .bool_set:N = \l_@@_respect_blocks_bool ,
3451   respect-blocks .default:n = true ,
3452   unknown .code:n = \@@_error:n { Unknown-option-for-rowcolors }
3453 }
3454 \NewDocumentCommand \@@_rowcolors { 0 { } m m m 0 { } }
3455 {
3456   \keys_set:nn { NiceMatrix / rowcolors } { #5 }
3457   \bool_lazy_and:nNTF
3458     \l_@@_respect_blocks_bool
3459     { \cs_if_exist_p:c { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq } }
3460     { \@@_rowcolors_i:nnnn { #1 } { #2 } { #3 } { #4 } }
3461     {
3462       \int_step_inline:nnn { #2 } { \int_use:N \c@iRow }
3463       {
3464         \int_if_odd:nTF { ##1 }
3465         { \@@_rowcolor [ #1 ] { #3 } }
3466         { \@@_rowcolor [ #1 ] { #4 } }
3467         { ##1 }
3468       }
3469     }
3470 }
3471 \cs_new_protected:Npn \@@_rowcolors_i:nnnn #1 #2 #3 #4
3472 {
3473   \seq_set_eq:Nc \l_tmpb_seq
3474   { c_@@_pos_of_blocks_ \int_use:N \g_@@_env_int _ seq }

```

We don't want to take into account a block which is completely in the “first column” of (number 0) or in the “last column”.

```

3475 \seq_set_filter:NNn \l_tmpa_seq \l_tmpb_seq
3476 { \@@_not_in_exterior_p:nnnn ##1 }

```

The counter `\l_tmpa_int` will be the index of the loop.

```

3477 \int_set:Nn \l_tmpa_int { #2 }

```

The boolean `\l_tmpa_bool` will indicate whereas we are in a row of the first color or of the second color.

```
3478 \bool_set_false:N \l_tmpa_bool
```

We recall that, in the code-before, `\c@iRow` is the total number of rows of the array (excepted the potential exterior rows).

```
3479 \int_do_until:nNn \l_tmpa_int > \c@iRow
3480 {
3481   \seq_set_filter:Nn \l_tmpb_seq \l_tmpa_seq
3482   { \@@_intersect_our_row_p:nnnn ##1 }
```

We compute in `\l_tmpb_int` the last row covered by a block.

```
3483   \int_set_eq:NN \l_tmpb_int \l_tmpa_int
3484   \seq_map_inline:Nn \l_tmpb_seq { \@@_rowcolors_ii:nnnn ##1 }
3485   \bool_if:NTF \l_tmpa_bool
3486   {
3487     \@@_rowcolor [ #1 ] { #4 }
3488     { \int_use:N \l_tmpa_int - \int_use:N \l_tmpb_int }
3489     \bool_set_false:N \l_tmpa_bool
3490   }
3491   {
3492     \@@_rowcolor [ #1 ] { #3 }
3493     { \int_use:N \l_tmpa_int - \int_use:N \l_tmpb_int }
3494     \bool_set_true:N \l_tmpa_bool
3495   }
3496   \int_set:Nn \l_tmpa_int { \l_tmpb_int + 1 }
3497 }
3498 }
```

```
3499 \cs_new_protected:Npn \@@_rowcolors_ii:nnnn #1 #2 #3 #4
3500 {
3501   \int_compare:nNnT { #3 } > \l_tmpb_int
3502   { \int_set:Nn \l_tmpb_int { #3 } }
3503 }
```

```
3504 \prg_new_conditional:Nnn \@@_not_in_exterior:nnnn p
3505 {
3506   \bool_lazy_or:nnTF
3507   { \int_compare_p:nNn { #4 } = \c_zero_int }
3508   { \int_compare_p:nNn { #2 } = { \@@_succ:n { \c@jCol } } }
3509   \prg_return_false:
3510   \prg_return_true:
3511 }
```

The following command return true when the block intersects the row `\l_tmpa_int`.

```
3512 \prg_new_conditional:Nnn \@@_intersect_our_row:nnnn p
3513 {
3514   \bool_if:nTF
3515   {
3516     \int_compare_p:n { #1 <= \l_tmpa_int }
3517     &&
3518     \int_compare_p:n { \l_tmpa_int <= #3 }
3519   }
3520   \prg_return_true:
3521   \prg_return_false:
3522 }
```

```
3523 \NewDocumentCommand \@@_chessboardcolors { 0 { } m m }
3524 {
3525   \int_step_inline:nn { \int_use:N \c@iRow }
3526   {
```

```

3527 \int_step_inline:nn { \int_use:N \c@jCol }
3528 {
3529   \int_if_even:nTF { ####1 + #1 }
3530   { \@@_cellcolor [ #1 ] { #2 } }
3531   { \@@_cellcolor [ #1 ] { #3 } }
3532   { ##1 - ####1 }
3533 }
3534 }
3535 }

```

When the user uses the key `colortbl`-like, the following command will be linked to `\cellcolor` in the tabular.

```

3536 \NewDocumentCommand \@@_cellcolor_tabular { 0 { } m }
3537 {
3538   \tl_gput_right:Nx \g_nicematrix_code_before_tl
3539   { \cellcolor [ #1 ] { #2 } { \int_use:N \c@iRow - \int_use:N \c@jCol } }
3540 }

```

When the user uses the key `rowcolor-in-tabular`, the following command will be linked to `\rowcolor` in the tabular.

```

3541 \NewDocumentCommand \@@_rowcolor_tabular { 0 { } m }
3542 {
3543   \tl_gput_right:Nx \g_nicematrix_code_before_tl
3544   {
3545     \exp_not:N \rectanglecolor [ #1 ] { #2 }
3546     { \int_use:N \c@iRow - \int_use:N \c@jCol }
3547     { \int_use:N \c@iRow - \exp_not:n { \int_use:N \c@jCol } }
3548   }
3549 }

```

```

3550 \NewDocumentCommand \@@_columncolor_preamble { 0 { } m }
3551 {
3552   \int_compare:nNnT \c@iRow = 1
3553   {

```

You use `gput_left` because we want the specification of colors for the columns drawn before the specifications of color for the rows (and the cells).

```

3554   \tl_gput_left:Nx \g_nicematrix_code_before_tl
3555   { \exp_not:N \columncolor [ #1 ] { #2 } { \int_use:N \c@jCol } }
3556 }
3557 }

```

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`.

```

3558 \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`.

```

3559 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix:n #1
3560 {
3561   \int_compare:nNnTF \l_@@_first_col_int = 0

```

```

3562 { \@@_OnlyMainNiceMatrix_i:n { #1 } }
3563 {
3564   \int_compare:nNnTF \c@jCol = 0
3565   {
3566     \int_compare:nNnF \c@iRow = { -1 }
3567     { \int_compare:nNnF \c@iRow = { \l_@@_last_row_int - 1 } { #1 } }
3568   }
3569   { \@@_OnlyMainNiceMatrix_i:n { #1 } }
3570 }
3571 }

```

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.

```

3572 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix_i:n #1
3573 {
3574   \int_compare:nNnF \c@iRow = 0
3575   { \int_compare:nNnF \c@iRow = \l_@@_last_row_int { #1 } }
3576 }

```

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

The following command will be executed in the `internal-code-after`. The rule will be drawn *before* the column `#1`. `#2` is the number of consecutive occurrences of `|`.

```

3577 \cs_new_protected:Npn \@@_vline:nn #1 #2
3578 {

```

The following test is for the case where the user don't use all the columns specified in the preamble of the environment (for instance, a preamble of `|c|c|c|` but only two columns used).

```

3579   \int_compare:nNnT { #1 } < { \c@jCol + 2 }
3580   {
3581     \pgfpicture
3582     \@@_vline_i:nn { #1 } { #2 }
3583     \endpgfpicture
3584   }
3585 }
3586 \cs_new_protected:Npn \@@_vline_i:nn #1 #2
3587 {

```

`\l_tmpa_tl` is the number of row and `\l_tmpb_tl` the number of column. When we have found a row corresponding to a rule to draw, we note its number in `\l_tmptc_tl`.

```

3588   \tl_set:Nx \l_tmpb_tl { #1 }
3589   \tl_clear_new:N \l_tmptc_tl
3590   \int_step_variable:nNn \c@iRow \l_tmpa_tl
3591   {

```

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.

```

3592   \bool_gset_true:N \g_tmpa_bool
3593   \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
3594   { \@@_test_if_vline_in_block:nnnn #1 }
3595   \seq_map_inline:Nn \g_@@_pos_of_xdots_seq
3596   { \@@_test_if_vline_in_block:nnnn #1 }
3597   \clist_if_empty:NF \l_@@_except_corners_clist
3598   \@@_test_in_corner_v:
3599   \bool_if:NTF \g_tmpa_bool
3600   {
3601     \tl_if_empty:NT \l_tmptc_tl

```

We keep in memory that we have a rule to draw.

```

3602         { \tl_set_eq:NN \l_tmpc_tl \l_tmpa_tl }
3603     }
3604     {
3605         \tl_if_empty:NF \l_tmpc_tl
3606         {
3607             \@@_vline_ii:nnnn
3608             { #1 }
3609             { #2 }
3610             \l_tmpc_tl
3611             { \int_eval:n { \l_tmpa_tl - 1 } }
3612             \tl_clear:N \l_tmpc_tl
3613         }
3614     }
3615 }
3616 \tl_if_empty:NF \l_tmpc_tl
3617 {
3618     \@@_vline_ii:nnnn
3619     { #1 }
3620     { #2 }
3621     \l_tmpc_tl
3622     { \int_use:N \c@iRow }
3623     \tl_clear:N \l_tmpc_tl
3624 }
3625 }

3626 \cs_new_protected:Npn \@@_test_in_corner_v:
3627 {
3628     \int_compare:nNnTF \l_tmpb_tl = { \@@_succ:n \c@jCol }
3629     {
3630         \seq_if_in:NxT
3631         \l_@@_empty_corner_cells_seq
3632         { \l_tmpa_tl - \@@_pred:n \l_tmpb_tl }
3633         { \bool_set_false:N \g_tmpa_bool }
3634     }
3635     {
3636         \seq_if_in:NxT
3637         \l_@@_empty_corner_cells_seq
3638         { \l_tmpa_tl - \l_tmpb_tl }
3639         {
3640             \int_compare:nNnTF \l_tmpb_tl = 1
3641             { \bool_set_false:N \g_tmpa_bool }
3642             {
3643                 \seq_if_in:NxT
3644                 \l_@@_empty_corner_cells_seq
3645                 { \l_tmpa_tl - \@@_pred:n \l_tmpb_tl }
3646                 { \bool_set_false:N \g_tmpa_bool }
3647             }
3648         }
3649     }
3650 }

```

#1 is the number of the column; #2 is the number of vertical rules to draw (with potentially a color between); #3 and #4 are the number of the rows between which the rule has to be drawn.

```

3651 \cs_new_protected:Npn \@@_vline_ii:nnnn #1 #2 #3 #4
3652 {
3653     \pgfrememberpicturepositiononpagetrue
3654     \pgf@relevantforpicturesizefalse
3655     \@@_qpoint:n { row - #3 }
3656     \dim_set_eq:NN \l_tmpa_dim \pgf@y
3657     \@@_qpoint:n { col - #1 }
3658     \dim_set_eq:NN \l_tmpb_dim \pgf@x

```



```

3659 \@@_qpoint:n { row - \@@_succ:n { #4 } }
3660 \dim_set_eq:NN \l_tmpc_dim \pgf@y
3661 \bool_lazy_and:nnT
3662   { \int_compare_p:nNn { #2 } > 1 }
3663   { ! \tl_if_blank_p:V \CT@drsc@ }
3664   {
3665     \group_begin:
3666     \CT@drsc@
3667     \dim_add:Nn \l_tmpa_dim { 0.5 \arrayrulewidth }
3668     \dim_sub:Nn \l_tmpc_dim { 0.5 \arrayrulewidth }
3669     \dim_set:Nn \l_tmpd_dim
3670       { \l_tmpb_dim - ( \doublerulesep + \arrayrulewidth ) * ( #2 - 1 ) }
3671     \pgfpathrectanglecorners
3672       { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
3673       { \pgfpoint \l_tmpd_dim \l_tmpc_dim }
3674     \pgfusepathqfill
3675     \group_end:
3676   }
3677 \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
3678 \pgfpathlineto { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
3679 \prg_replicate:nn { #2 - 1 }
3680 {
3681   \dim_sub:Nn \l_tmpb_dim \arrayrulewidth
3682   \dim_sub:Nn \l_tmpb_dim \doublerulesep
3683   \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
3684   \pgfpathlineto { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
3685 }
3686 \CT@arc@
3687 \pgfsetlinewidth { 1.1 \arrayrulewidth }
3688 \pgfsetrectcap
3689 \pgfusepathqstroke
3690 }

```

The following draws a complete vertical rule in the column #1 (#2 is the number of consecutive rules specified by the number of | in the preamble). This command will be used if there is no block in the array (and the key `except-corners` is not used).

```

3691 \cs_new_protected:Npn \@@_vline_i_complete:nn #1 #2
3692   { \@@_vline_ii:nnnn { #1 } { #2 } 1 { \int_use:N \c@iRow } }

```

The command `\@@_draw_hlines:` draws all the vertical rules excepted in the blocks, in the virtual blocks (determined by a command such as `\Cdots`) and in the corners (if the key `except-corners` is used).

```

3693 \cs_new_protected:Npn \@@_draw_vlines:
3694   {
3695     \int_step_inline:nnn
3696       { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
3697       { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@jCol } \c@jCol }
3698       { \@@_vline:nn { ##1 } 1 }
3699   }

```

The horizontal rules

The following command will be executed in the `internal-code-after`. The rule will be drawn *before* the row #1. #2 is the number of consecutive occurrences of `\Hline`.

```

3700 \cs_new_protected:Npn \@@_hline:nn #1 #2
3701   {
3702     \pgfpicture
3703     \@@_hline_i:nn { #1 } { #2 }
3704     \endpgfpicture
3705   }

```

```

3706 \cs_new_protected:Npn \@@_hline_i:nn #1 #2
3707 {

```

\l_tmpa_tl is the number of row and \l_tmpb_tl the number of column. Whe, we have found a column corresponding to a rule to draw, we note its number in \l_tmpe_tl.

```

3708 \tl_set:Nn \l_tmpa_tl { #1 }
3709 \tl_clear_new:N \l_tmpe_tl
3710 \int_step_variable:nNn \c@jCol \l_tmpb_tl
3711 {

```

The boolean \g_tmpe_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_tmpe_bool to false and the small horizontal rule won't be drawn.

```

3712 \bool_gset_true:N \g_tmpe_bool
3713 \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
3714 { \@@_test_if_hline_in_block:nnnn #1 }
3715 \seq_map_inline:Nn \g_@@_pos_of_xdots_seq
3716 { \@@_test_if_hline_in_block:nnnn #1 }
3717 \clist_if_empty:NF \l_@@_except_corners_clist \@@_test_in_corner_h:
3718 \bool_if:NTF \g_tmpe_bool
3719 {
3720 \tl_if_empty:NT \l_tmpe_tl

```

We keep in memory that we have a rule to draw.

```

3721 { \tl_set_eq:NN \l_tmpe_tl \l_tmpb_tl }
3722 }
3723 {
3724 \tl_if_empty:NF \l_tmpe_tl
3725 {
3726 \@@_hline_ii:nnnn
3727 { #1 }
3728 { #2 }
3729 \l_tmpe_tl
3730 { \int_eval:n { \l_tmpb_tl - 1 } }
3731 \tl_clear:N \l_tmpe_tl
3732 }
3733 }
3734 }
3735 \tl_if_empty:NF \l_tmpe_tl
3736 {
3737 \@@_hline_ii:nnnn
3738 { #1 }
3739 { #2 }
3740 \l_tmpe_tl
3741 { \int_use:N \c@jCol }
3742 \tl_clear:N \l_tmpe_tl
3743 }
3744 }

```

```

3745 \cs_new_protected:Npn \@@_test_in_corner_h:
3746 {
3747 \int_compare:nNnTF \l_tmpe_tl = { \@@_succ:n \c@iRow }
3748 {
3749 \seq_if_in:NxT
3750 \l_@@_empty_corner_cells_seq
3751 { \@@_pred:n \l_tmpe_tl - \l_tmpb_tl }
3752 { \bool_set_false:N \g_tmpe_bool }
3753 }
3754 {
3755 \seq_if_in:NxT
3756 \l_@@_empty_corner_cells_seq
3757 { \l_tmpe_tl - \l_tmpb_tl }
3758 {

```

```

3759         \int_compare:nNnTF \l_tmpa_tl = 1
3760         { \bool_set_false:N \g_tmpa_bool }
3761         {
3762             \seq_if_in:NxT
3763             \l_@@_empty_corner_cells_seq
3764             { \@@_pred:n \l_tmpa_tl - \l_tmpb_tl }
3765             { \bool_set_false:N \g_tmpa_bool }
3766         }
3767     }
3768 }
3769 }

```

#1 is the number of the row; #2 is the number of horizontal rules to draw (with potentially a color between); #3 and #4 are the number of the columns between which the rule has to be drawn.

```

3770 \cs_new_protected:Npn \@@_hline_ii:nnnn #1 #2 #3 #4
3771 {
3772     \pgfrememberpicturepositiononpagetrue
3773     \pgf@relevantforpicturesizefalse
3774     \@@_qpoint:n { col - #3 }
3775     \dim_set_eq:NN \l_tmpa_dim \pgf@x
3776     \@@_qpoint:n { row - #1 }
3777     \dim_set_eq:NN \l_tmpb_dim \pgf@y
3778     \@@_qpoint:n { col - \@@_succ:n { #4 } }
3779     \dim_set_eq:NN \l_tmpc_dim \pgf@x
3780     \bool_lazy_and:nnT
3781     { \int_compare_p:nNn { #2 } > 1 }
3782     { ! \tl_if_blank_p:V \CT@drsc@ }
3783     {
3784         \group_begin:
3785         \CT@drsc@
3786         \dim_set:Nn \l_tmpd_dim
3787         { \l_tmpb_dim - ( \doublerulesep + \arrayrulewidth ) * ( #2 - 1 ) }
3788         \pgfpathrectanglecorners
3789         { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3790         { \pgfpoint \l_tmpc_dim \l_tmpd_dim }
3791         \pgfusepathqfill
3792         \group_end:
3793     }
3794     \pgfpathmoveto { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3795     \pgfpathlineto { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
3796     \prg_replicate:nn { #2 - 1 }
3797     {
3798         \dim_sub:Nn \l_tmpb_dim \arrayrulewidth
3799         \dim_sub:Nn \l_tmpb_dim \doublerulesep
3800         \pgfpathmoveto { \pgfpoint \l_tmpa_dim \l_tmpb_dim }
3801         \pgfpathlineto { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
3802     }
3803     \CT@arc@
3804     \pgfsetlinewidth { 1.1 \arrayrulewidth }
3805     \pgfsetrectcap
3806     \pgfusepathqstroke
3807 }

```

```

3808 \cs_new_protected:Npn \@@_hline_i_complete:nn #1 #2
3809 { \@@_hline_ii:nnnn { #1 } { #2 } 1 { \int_use:N \c@jCol } }

```

The command `\@@_draw_hlines:` draws all the horizontal rules excepted in the blocks (even the virtual drawn determined by commands such as `\Cdots` and in the corners (if the key `except-corners` is used).

```

3810 \cs_new_protected:Npn \@@_draw_hlines:
3811 {

```

```

3812 \int_step_inline:nnn
3813 { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
3814 { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@iRow } \c@iRow }
3815 { \@@_hline:nn { ##1 } 1 }
3816 }

```

The command `\@@_Hline:` will be linked to `\Hline` in the environments of `nicematrix`.

```

3817 \cs_set:Npn \@@_Hline: { \noalign { \ifnum 0 = ` } \fi \@@_Hline_i:n { 1 } }

```

The argument of the command `\@@_Hline_i:n` is the number of successive `\Hline` found.

```

3818 \cs_set:Npn \@@_Hline_i:n #1
3819 {
3820   \peek_meaning_ignore_spaces:NTF \Hline
3821   { \@@_Hline_ii:nn { #1 + 1 } }
3822   { \@@_Hline_iii:n { #1 } }
3823 }
3824 \cs_set:Npn \@@_Hline_ii:nn #1 #2 { \@@_Hline_i:n { #1 } }
3825 \cs_set:Npn \@@_Hline_iii:n #1
3826 {
3827   \skip_vertical:n
3828   {
3829     \arrayrulewidth * ( #1 )
3830     + \doublerulesep * ( \int_max:nn 0 { #1 - 1 } )
3831   }
3832   \tl_gput_right:Nx \g_@@_internal_code_after_tl
3833   { \@@_hline:nn { \@@_succ:n { \c@iRow } } { #1 } }
3834   \ifnum 0 = ` { \fi }
3835 }

```

The key hvlines

The following command tests whether the current position in the array (given by `\l_tmpa_tl` for the row and `\l_tmpb_tl` for the col) would provide an horizontal rule towards the right in the block delimited by the four arguments `#1`, `#2`, `#3` and `#4`. If this rule would be in the block (it must not be drawn), the boolean `\l_tmpa_bool` is set to false.

```

3836 \cs_new_protected:Npn \@@_test_if_hline_in_block:nnnn #1 #2 #3 #4
3837 {
3838   \bool_lazy_all:nT
3839   {
3840     { \int_compare_p:nNn \l_tmpa_tl > { #1 } }
3841     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
3842     { \int_compare_p:nNn \l_tmpb_tl > { #2 - 1 } }
3843     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
3844   }
3845   { \bool_gset_false:N \g_tmpa_bool }
3846 }

```

The same for vertical rules.

```

3847 \cs_new_protected:Npn \@@_test_if_vline_in_block:nnnn #1 #2 #3 #4
3848 {
3849   \bool_lazy_all:nT
3850   {
3851     { \int_compare_p:nNn \l_tmpa_tl > { #1 - 1 } }
3852     { \int_compare_p:nNn \l_tmpa_tl < { #3 + 1 } }
3853     { \int_compare_p:nNn \l_tmpb_tl > { #2 } }
3854     { \int_compare_p:nNn \l_tmpb_tl < { #4 + 1 } }
3855   }
3856   { \bool_gset_false:N \g_tmpa_bool }
3857 }

```

The key except-corners

When the key `except-corners` is raised, the rules are not drawn in the corners. Of course, we have to compute the corners before we begin to draw the rules.

```
3858 \cs_new_protected:Npn \@@_compute_corners:
3859 {
```

The sequence `\l_@@_empty_corner_cells_seq` will be the sequence of all the empty cells (and not in a block) considered in the corners of the array.

```
3860   \seq_clear_new:N \l_@@_empty_corner_cells_seq
3861   \clist_map_inline:Nn \l_@@_except_corners_clist
3862   {
3863     \str_case:nnF { ##1 }
3864     {
3865       { NW }
3866       { \@@_compute_a_corner:nnnnnn 1 1 1 1 \c@iRow \c@jCol }
3867       { NE }
3868       { \@@_compute_a_corner:nnnnnn 1 \c@jCol 1 { -1 } \c@iRow 1 }
3869       { SW }
3870       { \@@_compute_a_corner:nnnnnn \c@iRow 1 { -1 } 1 1 \c@jCol }
3871       { SE }
3872       { \@@_compute_a_corner:nnnnnn \c@iRow \c@jCol { -1 } { -1 } 1 1 }
3873     }
3874     { \@@_error:nn { bad~corner } { ##1 } }
3875   }
3876 }
```

“Computing a corner” is determining all the empty cells (which are not in a block) that belong to that corner. These cells will be added to the sequence `\l_@@_empty_corner_cells_seq`.

The six arguments of `\@@_compute_a_corner:nnnnnn` are as follow:

- #1 and #2 are the number of row and column of the cell which is actually in the corner;
- #3 and #4 are the steps in rows and the step in columns when moving from the corner;
- #5 is the number of the final row when scanning the rows from the corner;
- #6 is the number of the final column when scanning the columns from the corner.

```
3877 \cs_new_protected:Npn \@@_compute_a_corner:nnnnnn #1 #2 #3 #4 #5 #6
3878 {
```

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.

```
3879   \bool_set_false:N \l_tmpa_bool
3880   \int_zero_new:N \l_@@_last_empty_row_int
3881   \int_set:Nn \l_@@_last_empty_row_int { #1 }
3882   \int_step_inline:nnnn { #1 } { #3 } { #5 }
3883   {
3884     \@@_test_if_cell_in_a_block:nn { ##1 } { \int_eval:n { #2 } }
3885     \bool_lazy_or:nnTF
3886     {
3887       \cs_if_exist_p:c
3888       { pgf @ sh @ ns @ \@@_env: - ##1 - \int_eval:n { #2 } }
3889     }
3890     \l_tmpb_bool
3891     { \bool_set_true:N \l_tmpa_bool }
3892     {
3893       \bool_if:NF \l_tmpa_bool
3894       { \int_set:Nn \l_@@_last_empty_row_int { ##1 } }
3895     }
3896   }
```

Now, you determine the last empty cell in the row of number 1.

```

3897 \bool_set_false:N \l_tmpa_bool
3898 \int_zero_new:N \l_@@_last_empty_column_int
3899 \int_set:Nn \l_@@_last_empty_column_int { #2 }
3900 \int_step_inline:nnnn { #2 } { #4 } { #6 }
3901 {
3902   \@@_test_if_cell_in_a_block:nn { \int_eval:n { #1 } } { ##1 }
3903   \bool_lazy_or:nnTF
3904     \l_tmpb_bool
3905     {
3906       \cs_if_exist_p:c
3907         { pgf @ sh @ ns @ \@@_env: - \int_eval:n { #1 } - ##1 }
3908     }
3909     { \bool_set_true:N \l_tmpa_bool }
3910     {
3911       \bool_if:NF \l_tmpa_bool
3912       { \int_set:Nn \l_@@_last_empty_column_int { ##1 } }
3913     }
3914 }

```

Now, we loop over the rows.

```

3915 \int_step_inline:nnnn { #1 } { #3 } \l_@@_last_empty_row_int
3916 {

```

We treat the row number ##1 with another loop.

```

3917 \bool_set_false:N \l_tmpa_bool
3918 \int_step_inline:nnnn { #2 } { #4 } \l_@@_last_empty_column_int
3919 {
3920   \@@_test_if_cell_in_a_block:nn { ##1 } { #####1 }
3921   \bool_lazy_or:nnTF
3922     \l_tmpb_bool
3923     {
3924       \cs_if_exist_p:c
3925         { pgf @ sh @ ns @ \@@_env: - ##1 - #####1 }
3926     }
3927     { \bool_set_true:N \l_tmpa_bool }
3928     {
3929       \bool_if:NF \l_tmpa_bool
3930       {
3931         \int_set:Nn \l_@@_last_empty_column_int { #####1 }
3932         \seq_put_right:Nn
3933           \l_@@_empty_corner_cells_seq
3934             { ##1 - #####1 }
3935       }
3936     }
3937 }
3938 }
3939 }

```

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`).

```

3940 \cs_new_protected:Npn \@@_test_if_cell_in_a_block:nn #1 #2
3941 {
3942   \int_set:Nn \l_tmpa_int { #1 }
3943   \int_set:Nn \l_tmpb_int { #2 }
3944   \bool_set_false:N \l_tmpb_bool
3945   \seq_map_inline:Nn \g_@@_pos_of_blocks_seq
3946     { \@@_test_if_cell_in_block:nnnnnn \l_tmpa_int \l_tmpb_int ##1 }
3947 }
3948 \cs_new_protected:Npn \@@_test_if_cell_in_block:nnnnnn #1 #2 #3 #4 #5 #6
3949 {
3950   \int_compare:nNnT { #3 } < { \@@_succ:n { #1 } }

```

```

3951 {
3952   \int_compare:nNnT { #1 } < { \@@_succ:n { #5 } }
3953   {
3954     \int_compare:nNnT { #4 } < { \@@_succ:n { #2 } }
3955     {
3956       \int_compare:nNnT { #2 } < { \@@_succ:n { #6 } }
3957       { \bool_set_true:N \l_tmpb_bool }
3958     }
3959   }
3960 }
3961 }

```

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`.

```

3962 \cs_new:Npn \@@_hdottedline:
3963 {
3964   \noalign { \skip_vertical:N 2\l_@@_radius_dim }
3965   \@@_hdottedline_i:
3966 }

```

On the other side, the following command should be protected.

```

3967 \cs_new_protected:Npn \@@_hdottedline_i:
3968 {

```

We write in the code-after the instruction that will actually draw the dotted line. It's not possible to draw this dotted line now because we don't know the length of the line (we don't even know the number of columns).

```

3969   \tl_gput_right:Nx \g_@@_internal_code_after_tl
3970   { \@@_hdottedline:n { \int_use:N \c@iRow } }
3971 }

```

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.

```

3972 \AtBeginDocument
3973 {

```

We recall that, when externalization is used, `\tikzpicture` and `\endtikzpicture` (or `\pgfpicture` and `\endpgfpicture`) must be directly “visible”. That's why we construct now a version of `\@@_hdottedline:n` with the right environment (`\begin{pgfpicture}\end{pgfpicture}` or `\begin{tikzpicture}...\end{tikzpicture}`).

```

3974   \cs_new_protected:Npx \@@_hdottedline:n #1
3975   {
3976     \bool_set_true:N \exp_not:N \l_@@_initial_open_bool
3977     \bool_set_true:N \exp_not:N \l_@@_final_open_bool
3978     \c_@@_pgfortikzpicture_tl
3979     \@@_hdottedline_i:n { #1 }
3980     \c_@@_endpgfortikzpicture_tl
3981   }
3982 }

```

The following command *must* be protected since it is used in the construction of `\@@_hdottedline:n`.

```

3983 \cs_new_protected:Npn \@@_hdottedline_i:n #1
3984 {
3985   \pgfrememberpicturepositiononpagetrue
3986   \@@_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).

```

3987 \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
3988 \dim_sub:Nn \l_@@_y_initial_dim \l_@@_radius_dim
3989 \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim

```

The dotted line will be extended if the user uses `margin` (or `left-margin` and `right-margin`).

The aim is that, by standard the dotted line fits between square brackets (`\hline` doesn’t).

```
\begin{bNiceMatrix}
```

```
1 & 2 & 3 & 4 \\\
```

```
\hline
```

```
1 & 2 & 3 & 4 \\\
```

```
\hdottedline
```

```
1 & 2 & 3 & 4
```

```
\end{bNiceMatrix}
```

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

But, if the user uses `margin`, the dotted line extends to have the same width as a `\hline`.

```
\begin{bNiceMatrix}[margin]
```

```
1 & 2 & 3 & 4 \\\
```

```
\hline
```

```
1 & 2 & 3 & 4 \\\
```

```
\hdottedline
```

```
1 & 2 & 3 & 4
```

```
\end{bNiceMatrix}
```

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

```

3990 \@@_qpoint:n { col - 1 }
3991 \dim_set:Nn \l_@@_x_initial_dim
3992 {
3993 \pgf@x +

```

We do a reduction by `\arraycolsep` for the environments with delimiters (and not for the other).

```

3994 \bool_if:NTF \l_@@_NiceArray_bool \c_zero_dim \arraycolsep
3995 - \l_@@_left_margin_dim
3996 }
3997 \@@_qpoint:n { col - \@@_succ:n \c@jCol }
3998 \dim_set:Nn \l_@@_x_final_dim
3999 {
4000 \pgf@x -
4001 \bool_if:NTF \l_@@_NiceArray_bool \c_zero_dim \arraycolsep
4002 + \l_@@_right_margin_dim
4003 }

```

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.

```

4004 \tl_set:Nn \l_tmpa_tl { ( }
4005 \tl_if_eq:NMF \l_@@_left_delim_tl \l_tmpa_tl
4006 { \dim_gadd:Nn \l_@@_x_initial_dim { 0.5 \l_@@_inter_dots_dim } }
4007 \tl_set:Nn \l_tmpa_tl { ) }
4008 \tl_if_eq:NMF \l_@@_right_delim_tl \l_tmpa_tl
4009 { \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`.

```

4010 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
4011 \@@_draw_line:
4012 }

```

Vertical dotted lines

```

4013 \cs_new_protected:Npn \@@_vdottedline:n #1
4014 {
4015 \bool_set_true:N \l_@@_initial_open_bool
4016 \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”.

```

4017   \bool_if:NTF \c_@@_tikz_loaded_bool
4018   {
4019     \tikzpicture
4020     \@@_vdottedline_i:n { #1 }
4021     \endtikzpicture
4022   }
4023   {
4024     \pgfpicture
4025     \@@_vdottedline_i:n { #1 }
4026     \endpgfpicture
4027   }
4028 }

```

```

4029 \cs_new_protected:Npn \@@_vdottedline_i:n #1
4030 {

```

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.

```

4031   \CT@arc@
4032   \pgfrememberpicturepositiononpagetrue
4033   \@@_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).

```

4034   \dim_set:Nn \l_@@_x_initial_dim { \pgf@x - \l_@@_radius_dim }
4035   \dim_set:Nn \l_@@_x_final_dim { \pgf@x - \l_@@_radius_dim }
4036   \@@_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.

```

4037   \dim_set:Nn \l_@@_y_initial_dim { \pgf@y - 0.5 \l_@@_inter_dots_dim }
4038   \@@_qpoint:n { row - \@@_succ:n \c@iRow }
4039   \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`.

```

4040   \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
4041   \@@_draw_line:
4042 }

```

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.

```

4043 \bool_new:N \l_@@_block_auto_columns_width_bool

```

As of now, there is only one option available for the environment `{NiceMatrixBlock}`.

```

4044 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
4045 {
4046   auto-columns-width .code:n =
4047   {
4048     \bool_set_true:N \l_@@_block_auto_columns_width_bool
4049     \dim_gzero_new:N \g_@@_max_cell_width_dim
4050     \bool_set_true:N \l_@@_auto_columns_width_bool
4051   }
4052 }

```

```

4053 \NewDocumentEnvironment { NiceMatrixBlock } { ! 0 { } }
4054 {
4055   \int_gincr:N \g_@@_NiceMatrixBlock_int
4056   \dim_zero:N \l_@@_columns_width_dim
4057   \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
4058   \bool_if:NT \l_@@_block_auto_columns_width_bool
4059   {
4060     \cs_if_exist:cT { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
4061     {
4062       \exp_args:Nnc \dim_set:Nn \l_@@_columns_width_dim
4063       { @@_max_cell_width _ \int_use:N \g_@@_NiceMatrixBlock_int }
4064     }
4065   }
4066 }

```

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).

```

4067 {
4068   \bool_if:NT \l_@@_block_auto_columns_width_bool
4069   {
4070     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
4071     \iow_shipout:Nx \@mainaux
4072     {
4073       \cs_gset:cpn
4074       { @@ _ 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.

```

4075       { \dim_eval:n { \g_@@_max_cell_width_dim + \arrayrulewidth } }
4076     }
4077     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
4078   }
4079 }

```

The extra nodes

First, two variants of the functions \dim_min:nn and \dim_max:nn.

```

4080 \cs_generate_variant:Nn \dim_min:nn { v n }
4081 \cs_generate_variant:Nn \dim_max:nn { v n }

```

The following command is called in \@@_use_arraybox_with_notes_c: just before the construction of the blocks (if the creation of medium nodes is required, medium nodes are also created for the blocks dans that construction uses the standard medium nodes).

```

4082 \cs_new_protected:Npn \@@_create_extra_nodes:
4083 {
4084   \bool_if:nTF \l_@@_medium_nodes_bool
4085   {
4086     \bool_if:NTF \l_@@_large_nodes_bool
4087     \@@_create_medium_and_large_nodes:
4088     \@@_create_medium_nodes:
4089   }
4090   { \bool_if:NT \l_@@_large_nodes_bool \@@_create_large_nodes: }
4091 }

```

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`.

```

4092 \cs_new_protected:Npn \@@_computations_for_medium_nodes:
4093 {
4094   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4095   {
4096     \dim_zero_new:c { l_@@_row_\@@_i: _min_dim }
4097     \dim_set_eq:cN { l_@@_row_\@@_i: _min_dim } \c_max_dim
4098     \dim_zero_new:c { l_@@_row_\@@_i: _max_dim }
4099     \dim_set:cn { l_@@_row_\@@_i: _max_dim } { - \c_max_dim }
4100   }
4101   \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
4102   {
4103     \dim_zero_new:c { l_@@_column_\@@_j: _min_dim }
4104     \dim_set_eq:cN { l_@@_column_\@@_j: _min_dim } \c_max_dim
4105     \dim_zero_new:c { l_@@_column_\@@_j: _max_dim }
4106     \dim_set:cn { l_@@_column_\@@_j: _max_dim } { - \c_max_dim }
4107   }

```

We begin the two nested loops over the rows and the columns of the array.

```

4108   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4109   {
4110     \int_step_variable:nnNn
4111     \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.

```

4112     {
4113       \cs_if_exist:cT
4114       { 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`.

```

4115     {
4116       \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { south-west }
4117       \dim_set:cn { l_@@_row_\@@_i: _min_dim }
4118       { \dim_min:vn { l_@@_row _ \@@_i: _min_dim } \pgf@y }
4119       \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
4120       {
4121         \dim_set:cn { l_@@_column _ \@@_j: _min_dim }
4122         { \dim_min:vn { l_@@_column _ \@@_j: _min_dim } \pgf@x }
4123       }

```

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`.

```

4124       \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { north-east }
4125       \dim_set:cn { l_@@_row _ \@@_i: _max_dim }
4126       { \dim_max:vn { l_@@_row _ \@@_i: _max_dim } \pgf@y }
4127       \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
4128       {
4129         \dim_set:cn { l_@@_column _ \@@_j: _max_dim }
4130         { \dim_max:vn { l_@@_column _ \@@_j: _max_dim } \pgf@x }
4131       }
4132     }
4133   }
4134 }

```

Now, we have to deal with empty rows or empty columns since we don't have created nodes in such rows and columns.

```

4135 \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4136 {
4137   \dim_compare:nNnT
4138     { \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } } = \c_max_dim
4139     {
4140       \@@_qpoint:n { row - \@@_i: - base }
4141       \dim_set:cn { l_@@_row _ \@@_i: _ max _ dim } \pgf@y
4142       \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim } \pgf@y
4143     }
4144   }
4145 \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
4146 {
4147   \dim_compare:nNnT
4148     { \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } } = \c_max_dim
4149     {
4150       \@@_qpoint:n { col - \@@_j: }
4151       \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim } \pgf@y
4152       \dim_set:cn { l_@@_column _ \@@_j: _ min _ dim } \pgf@y
4153     }
4154   }
4155 }

```

Here is the command `\@@_create_medium_nodes:`. When this command is used, the “medium nodes” are created.

```

4156 \cs_new_protected:Npn \@@_create_medium_nodes:
4157 {
4158   \pgfpicture
4159   \pgfrememberpicturepositiononpagetrue
4160   \pgf@relevantforpicturesizefalse
4161   \@@_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”.

```

4162   \tl_set:Nn \l_@@_suffix_tl { -medium }
4163   \@@_create_nodes:
4164   \endpgfpicture
4165 }

```

The command `\@@_create_large_nodes:` must be used when we want to create only the “large nodes” and not the medium ones⁴⁵. 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:`.

```

4166 \cs_new_protected:Npn \@@_create_large_nodes:
4167 {
4168   \pgfpicture
4169   \pgfrememberpicturepositiononpagetrue
4170   \pgf@relevantforpicturesizefalse
4171   \@@_computations_for_medium_nodes:
4172   \@@_computations_for_large_nodes:
4173   \tl_set:Nn \l_@@_suffix_tl { -large }
4174   \@@_create_nodes:
4175   \endpgfpicture
4176 }
4177 \cs_new_protected:Npn \@@_create_medium_and_large_nodes:
4178 {
4179   \pgfpicture
4180   \pgfrememberpicturepositiononpagetrue

```

⁴⁵If we want to create both, we have to use `\@@_create_medium_and_large_nodes:`

```

4181 \pgf@relevantforpicturesizefalse
4182 \@@_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”.

```

4183 \tl_set:Nn \l_@@_suffix_tl { - medium }
4184 \@@_create_nodes:
4185 \@@_computations_for_large_nodes:
4186 \tl_set:Nn \l_@@_suffix_tl { - large }
4187 \@@_create_nodes:
4188 \endpgfpicture
4189 }

```

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.

```

4190 \cs_new_protected:Npn \@@_computations_for_large_nodes:
4191 {
4192 \int_set:Nn \l_@@_first_row_int 1
4193 \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`.

```

4194 \int_step_variable:nNn { \c@iRow - 1 } \@@_i:
4195 {
4196 \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim }
4197 {
4198 (
4199 \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } +
4200 \dim_use:c { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
4201 )
4202 / 2
4203 }
4204 \dim_set_eq:cc { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
4205 { l_@@_row _ \@@_i: _ min _ dim }
4206 }
4207 \int_step_variable:nNn { \c@jCol - 1 } \@@_j:
4208 {
4209 \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim }
4210 {
4211 (
4212 \dim_use:c { l_@@_column _ \@@_j: _ max _ dim } +
4213 \dim_use:c
4214 { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
4215 )
4216 / 2
4217 }
4218 \dim_set_eq:cc { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
4219 { l_@@_column _ \@@_j: _ max _ dim }
4220 }

```

Here, we have to use `\dim_sub:cn` because of the number 1 in the name.

```

4221 \dim_sub:cn
4222 { l_@@_column _ 1 _ min _ dim }
4223 \l_@@_left_margin_dim
4224 \dim_add:cn
4225 { l_@@_column _ \int_use:N \c@jCol _ max _ dim }
4226 \l_@@_right_margin_dim
4227 }

```

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 $\backslash l_@@_suffix_tl$ (-medium or -large).

```

4228 \cs_new_protected:Npn \@@_create_nodes:
4229 {
4230   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
4231   {
4232     \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
4233     {

```

We draw the rectangular node for the cell ($\backslash@@_i$ - $\backslash@@_j$).

```

4234       \@@_pgf_rect_node:nnnnn
4235       { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
4236       { \dim_use:c { l_@@_column_ \@@_j: _min_dim } }
4237       { \dim_use:c { l_@@_row_ \@@_i: _min_dim } }
4238       { \dim_use:c { l_@@_column_ \@@_j: _max_dim } }
4239       { \dim_use:c { l_@@_row_ \@@_i: _max_dim } }
4240       \str_if_empty:NF \l_@@_name_str
4241       {
4242         \pgfnodealias
4243         { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
4244         { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
4245       }
4246     }
4247   }

```

Now, we create the nodes for the cells of the $\backslash multicolumn$. We recall that we have stored in $\backslash g_@@_multicolumn_cells_seq$ the list of the cells where a $\backslash multicolumn\{n\}\{\dots\}\{\dots\}$ with $n > 1$ was issued and in $\backslash g_@@_multicolumn_sizes_seq$ the correspondent values of n .

```

4248   \seq_mapthread_function:NNN
4249   \g_@@_multicolumn_cells_seq
4250   \g_@@_multicolumn_sizes_seq
4251   \@@_node_for_multicolumn:nn
4252 }

```

```

4253 \cs_new_protected:Npn \@@_extract_coords_values: #1 - #2 \q_stop
4254 {
4255   \cs_set_nopar:Npn \@@_i: { #1 }
4256   \cs_set_nopar:Npn \@@_j: { #2 }
4257 }

```

The command $\backslash@@_node_for_multicolumn:nn$ takes two arguments. The first is the position of the cell where the command $\backslash multicolumn\{n\}\{\dots\}\{\dots\}$ was issued in the format i - j and the second is the value of n (the length of the “multi-cell”).

```

4258 \cs_new_protected:Npn \@@_node_for_multicolumn:nn #1 #2
4259 {
4260   \@@_extract_coords_values: #1 \q_stop
4261   \@@_pgf_rect_node:nnnnn
4262   { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
4263   { \dim_use:c { l_@@_column_ \@@_j: _min_dim } }
4264   { \dim_use:c { l_@@_row_ \@@_i: _min_dim } }
4265   { \dim_use:c { l_@@_column_ \int_eval:n { \@@_j: +#2-1 } _max_dim } }
4266   { \dim_use:c { l_@@_row_ \@@_i: _max_dim } }
4267   \str_if_empty:NF \l_@@_name_str
4268   {
4269     \pgfnodealias
4270     { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
4271     { \int_use:N \g_@@_env_int - \@@_i: - \@@_j: \l_@@_suffix_tl }
4272   }
4273 }

```

The blocks

The code deals with the command `\Block`. This command has no direct link with the environment `{NiceMatrixBlock}`.

The options of the command `\Block` will be analyzed first in the cell of the array (and once again when the block will be put in the array in the `code-after`). Here is the set of keys for the first pass.

```
4274 \keys_define:nn { NiceMatrix / Block / FirstPass }
4275 {
4276   l .code:n = \tl_set:Nn \l_@@_pos_of_block_tl l ,
4277   l .value_forbidden:n = true ,
4278   r .code:n = \tl_set:Nn \l_@@_pos_of_block_tl r ,
4279   r .value_forbidden:n = true ,
4280   c .code:n = \tl_set:Nn \l_@@_pos_of_block_tl c ,
4281   c .value_forbidden:n = true ,
4282 }
```

The following command `\@@_Block:` will be linked to `\Block` in the environments of `nicematrix`. We define it with `\NewExpandableDocumentCommand` because it has an optional argument between `<` and `>` (for TeX instructions put before the math mode of the label). It's mandatory to use an expandable command.

```
4283 \NewExpandableDocumentCommand \@@_Block: { 0 { } m D < > { } m }
4284 {
```

If the first mandatory argument of the command (which is the size of the block with the syntax $i-j$) has not been provided by the user, you use `1-1` (that is to say a block of only one cell).

```
4285   \tl_if_blank:nTF { #2 } { \@@_Block_i 1-1 \q_stop } { \@@_Block_i #2 \q_stop }
4286   { #1 } { #3 } { #4 }
4287 }
```

With the following construction, we extract the values of i and j in the first mandatory argument of the command.

```
4288 \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.

```
4289 \cs_new_protected:Npn \@@_Block_ii:nnnnn #1 #2 #3 #4 #5
4290 {
4291   \tl_if_empty:nTF \l_@@_cell_type_tl
4292   { \tl_set:Nn \l_@@_pos_of_block_tl c }
4293   { \tl_set_eq:NN \l_@@_pos_of_block_tl \l_@@_cell_type_tl }
4294   \keys_set_known:nn { NiceMatrix / Block / FirstPass } { #3 }
```

We recall that `#1` and `#2` have been extracted from the first mandatory argument of `\Block` (which is of the syntax $i-j$). However, the user is allowed to omit i or j (or both). We detect that situation by replacing a missing value by 100 (it's a convention: when the block will actually be drawn these values will be detected and interpreted as *maximal possible value*).

```
4295   \tl_if_empty:nTF { #1 }
4296   { \int_set:Nn \l_tmpa_int { 100 } }
4297   { \int_set:Nn \l_tmpa_int { #1 } }
4298   \tl_if_empty:nTF { #2 }
4299   { \int_set:Nn \l_tmpb_int { 100 } }
4300   { \int_set:Nn \l_tmpb_int { #2 } }
4301   \tl_set:Nx \l_tmpa_tl
4302   {
4303     { \int_use:N \c@iRow }
4304     { \int_use:N \c@jCol }
4305     { \int_eval:n { \c@iRow + \l_tmpa_int - 1 } }
4306     { \int_eval:n { \c@jCol + \l_tmpb_int - 1 } }
4307   }
```

Now, `\l_tmpa_tl` contains an “object” corresponding to the position of the block with four components, each of them surrounded by curly brackets:

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

We store this information in the sequence `\g_@@_pos_of_blocks_seq`.

```
4308 \seq_gput_left:NV \g_@@_pos_of_blocks_seq \l_tmpa_tl
```

We also store a complete description of the block in the sequence `\g_@@_blocks_seq`. Of course, the sequences `\g_@@_pos_of_blocks_seq` and `\g_@@_blocks_seq` are redundant, but it’s for efficiency. In `\g_@@_blocks_seq`, each block is represented by an “object” with six components:

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

If the block is mono-column or mono-row, we have a special treatment. That’s why we have two macros: `\@@_Block_iii:nnnnn` and `\@@_Block_iv:nnnnn` (the five arguments of those macros are provided by currying).

```
4309 \bool_lazy_or:nnTF
4310 { \int_compare_p:nNn { \l_tmpa_int } = 1 }
4311 { \int_compare_p:nNn { \l_tmpb_int } = 1 }
4312 { \exp_args:Nxx \@@_Block_iii:nnnnn }
4313 { \exp_args:Nxx \@@_Block_iv:nnnnn }
4314 { \l_tmpa_int } { \l_tmpb_int } { #3 } { #4 } { #5 }
4315 }
```

The following macro is for the case of a `\Block` which is mono-row or mono-column (or both). In that case, the content of the block is composed right now in a box (because we have to take into account the dimensions of that box for the width of the current column or the height and the depth of the current row). However, that box will be put in the array *after the construction of the array* (by using PGF).

```
4316 \cs_new_protected:Npn \@@_Block_iii:nnnnn #1 #2 #3 #4 #5
4317 {
4318   \int_gincr:N \g_@@_block_box_int
4319   \cs_set_protected_nopar:Npn \diagbox ##1 ##2
4320   {
4321     \tl_gput_right:Nx \g_@@_internal_code_after_tl
4322     {
4323       \@@_actually_diagbox:nnnnnn
4324       { \int_use:N \c@iRow }
4325       { \int_use:N \c@jCol }
4326       { \int_eval:n { \c@iRow + #1 - 1 } }
4327       { \int_eval:n { \c@jCol + #2 - 1 } }
4328       { \exp_not:n { ##1 } } { \exp_not:n { ##2 } }
4329     }
4330   }
4331   \box_gclear_new:c
4332   { g_@@_block_box_int \int_use:N \g_@@_block_box_int _box }
4333   \hbox_gset:cn
4334   { g_@@_block_box_int \int_use:N \g_@@_block_box_int _box }
4335   {
```

For a mono-column block, if the user has specified a color for the column in the preamble of the array, we want to fix that color in the box we construct. We do that with `\set@color` and not `\color_ensure_current:` because that command seems to be bugged: it doesn’t work in XeLaTeX when `fontspec` is loaded.

```
4336   \int_compare:nNnT { #2 } = 1 \set@color
4337   \bool_if:NTF \l_@@_NiceTabular_bool
4338   {
4339     \group_begin:
4340     \cs_set:Npn \arraystretch { 1 }
4341     \dim_set_eq:NN \extrarowheight \c_zero_dim
4342     #4
```

If the box is rotated (the key `\rotate` may be in the previous #4), the tabular used for the content of the cell will be constructed with a format `c`. In the other cases, the tabular will be constructed with a format equal to the key of position of the box. In other words: the alignment internal to the

tabular is the same as the external alignment of the tabular (that is to say the position of the block in its zone of merged cells).

```

4343         \bool_if:NT \g_@@_rotate_bool
4344         { \tl_set:Nn \l_@@_pos_of_block_tl c }
4345         \exp_args:Nnx \begin { tabular }
4346         { @ { } \l_@@_pos_of_block_tl @ { } }
4347         #5
4348         \end { tabular }
4349         \group_end:
4350     }
4351     {
4352         \group_begin:
4353         \cs_set:Npn \arraystretch { 1 }
4354         \dim_set_eq:NN \extrarowheight \c_zero_dim
4355         #4
4356         \bool_if:NT \g_@@_rotate_bool
4357         { \tl_set:Nn \l_@@_pos_of_block_tl c }
4358         \c_math_toggle_token
4359         \exp_args:Nnx \begin { array }
4360         { @ { } \l_@@_pos_of_block_tl @ { } }
4361         #5
4362         \end { array }
4363         \c_math_toggle_token
4364         \group_end:
4365     }
4366 }
4367 \bool_if:NT \g_@@_rotate_bool
4368 {
4369     \box_grotate:cn
4370     { g_@@_block _ box _ \int_use:N \g_@@_block_box_int _ box }
4371     { 90 }
4372     \bool_gset_false:N \g_@@_rotate_bool
4373 }

```

If we are in a mono-column block, we take into account the width of that block for the width of the column.

```

4374     \int_compare:nNnT { #2 } = 1
4375     {
4376         \dim_gset:Nn \g_@@_blocks_wd_dim
4377         {
4378             \dim_max:nn
4379             \g_@@_blocks_wd_dim
4380             {
4381                 \box_wd:c
4382                 { g_@@_block _ box _ \int_use:N \g_@@_block_box_int _ box }
4383             }
4384         }
4385     }

```

If we are in a mono-row block, we take into account the height and the depth of that block for the height and the depth of the row.

```

4386     \int_compare:nNnT { #1 } = 1
4387     {
4388         \dim_gset:Nn \g_@@_blocks_ht_dim
4389         {
4390             \dim_max:nn
4391             \g_@@_blocks_ht_dim
4392             {
4393                 \box_ht:c
4394                 { g_@@_block _ box _ \int_use:N \g_@@_block_box_int _ box }
4395             }
4396         }
4397         \dim_gset:Nn \g_@@_blocks_dp_dim
4398         {

```

```

4399         \dim_max:nn
4400         \g_@@_blocks_dp_dim
4401         {
4402             \box_dp:c
4403             { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
4404         }
4405     }
4406 }
4407 \seq_gput_right:Nx \g_@@_blocks_seq
4408 {
4409     \l_tmpa_tl

```

In the list of options #3, maybe there is a key for the horizontal alignment (l, r or c). In that case, that key has been read and stored in \l_@@_pos_of_block_tl. However, maybe there were no key of the horizontal alignment and that's why we put a key corresponding to the value of \l_@@_pos_of_block_tl, which is fixed by the type of current column.

```

4410     { #3 , \l_@@_pos_of_block_tl }
4411     {
4412         \box_use_drop:c
4413         { g_@@_ block _ box _ \int_use:N \g_@@_block_box_int _ box }
4414     }
4415 }
4416 }

```

The following macro is for the standard case, where the block is not mono-row and not mono-column. In that case, the content of the block is *not* composed right now in a box. The composition in a box will be done further, just after the construction of the array.

```

4417 \cs_new_protected:Npn \@@_Block_iv:nnnnn #1 #2 #3 #4 #5
4418 {
4419     \seq_gput_right:Nx \g_@@_blocks_seq
4420     {
4421         \l_tmpa_tl
4422         { #3 }
4423         \exp_not:n
4424         {
4425             {
4426                 \bool_if:NTF \l_@@_NiceTabular_bool
4427                 {
4428                     \group_begin:
4429                     \cs_set:Npn \arraystretch { 1 }
4430                     \dim_set_eq:NN \extrarowheight \c_zero_dim
4431                     #4

```

If the box is rotated (the key \rotate may be in the previous #4), the tabular used for the content of the cell will be constructed with a format c. In the other cases, the tabular will be constructed with a format equal to the key of position of the box. In other words: the alignment internal to the tabular is the same as the external alignment of the tabular (that is to say the position of the block in its zone of merged cells).

```

4432         \bool_if:NT \g_@@_rotate_bool
4433         { \tl_set:Nn \l_@@_pos_of_block_tl c }
4434         \exp_args:Nnx \begin { tabular }
4435             { @ { } \l_@@_pos_of_block_tl @ { } }
4436             #5
4437         \end { tabular }
4438         \group_end:
4439     }
4440     {
4441         \group_begin:
4442         \cs_set:Npn \arraystretch { 1 }
4443         \dim_set_eq:NN \extrarowheight \c_zero_dim
4444         #4
4445         \bool_if:NT \g_@@_rotate_bool
4446         { \tl_set:Nn \l_@@_pos_of_block_tl c }

```

```

4447         \c_math_toggle_token
4448         \exp_args:Nnx \begin { array }
4449         { @ { } \l_@@_pos_of_block_tl @ { } } #5 \end { array }
4450         \c_math_toggle_token
4451         \group_end:
4452     }
4453 }
4454 }
4455 }
4456 }

```

We recall that the options of the command `\Block` are analyzed twice: first in the cell of the array and once again when the block will be put in the array *after the construction of the array* (by using PGF).

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?

```

4457 \keys_define:nn { NiceMatrix / Block / SecondPass }
4458 {
4459     tikz .tl_set:N = \l_@@_tikz_tl ,
4460     tikz .value_required:n = true ,
4461     color .tl_set:N = \l_@@_color_tl ,
4462     color .value_required:n = true ,
4463     l .code:n = \tl_set:Nn \l_@@_pos_of_block_tl l ,
4464     l .value_forbidden:n = true ,
4465     r .code:n = \tl_set:Nn \l_@@_pos_of_block_tl r ,
4466     r .value_forbidden:n = true ,
4467     c .code:n = \tl_set:Nn \l_@@_pos_of_block_tl c ,
4468     c .value_forbidden:n = true ,
4469     unknown .code:n = \@@_error:n { Unknown-key-for-Block }
4470 }

```

The command `\@@_draw_blocks:` will draw all the blocks. This command is used after the construction of the array. We have to revert to a clean version of `\ialign` because there may be tabulars in the `\Block` instructions that will be composed now.

```

4471 \cs_new_protected:Npn \@@_draw_blocks:
4472 {
4473     \cs_set_eq:NN \ialign \@@_old_ialign:
4474     \seq_map_inline:Nn \g_@@_blocks_seq { \@@_Block_iii:nnnnnn ##1 }
4475 }
4476 \cs_new_protected:Npn \@@_Block_iii:nnnnnn #1 #2 #3 #4 #5 #6
4477 {

```

The group is for the keys.

```

4478     \group_begin:
4479     \keys_set:nn { NiceMatrix / Block / SecondPass } { #5 }

```

The integer `\l_@@_last_row_int` will be the last row of the block and `\l_@@_last_col_int` its last column.

```

4480     \int_zero_new:N \l_@@_last_row_int
4481     \int_zero_new:N \l_@@_last_col_int

```

We remind that the first mandatory argument of the command `\Block` is the size of the block with the special format i - j . However, the user is allowed to omit i or j (or both). This will be interpreted as: the last row (resp. column) of the block will be the last row (resp. column) of the block (without the potential exterior row—resp. column—of the array). By convention, this is stored in `\g_@@_blocks_seq` as a number of rows (resp. columns) for the block equal to 100. That’s what we detect now.

```

4482     \int_compare:nNnTF { #3 } > { 99 }
4483     { \int_set_eq:NN \l_@@_last_row_int \c@iRow }
4484     { \int_set:Nn \l_@@_last_row_int { #3 } }

```

```

4485 \int_compare:nNnTF { #4 } > { 99 }
4486 { \int_set_eq:NN \l_@@_last_col_int \c@jCol }
4487 { \int_set:Nn \l_@@_last_col_int { #4 } }
4488 \tl_if_empty:NF \l_@@_color_tl
4489 {
4490   \tl_gput_right:Nx \g_nicematrix_code_before_tl
4491   {
4492     \exp_not:N \rectanglecolor
4493     { \l_@@_color_tl }
4494     { #1 - #2 }
4495     { \int_use:N \l_@@_last_row_int - \int_use:N \l_@@_last_col_int }
4496   }
4497 }

4498 \cs_set_protected_nopar:Npn \diagbox ##1 ##2
4499 {
4500   \tl_gput_right:Nx \g_@@_internal_code_after_tl
4501   {
4502     \@@_actually_diagbox:nnnnnn
4503     { #1 }
4504     { #2 }
4505     { \int_use:N \l_@@_last_row_int }
4506     { \int_use:N \l_@@_last_col_int }
4507     { \exp_not:n { ##1 } } { \exp_not:n { ##2 } }
4508   }
4509 }

4510 \bool_lazy_or:nnTF
4511 { \int_compare_p:nNn \l_@@_last_row_int > \g_@@_row_total_int }
4512 { \int_compare_p:nNn \l_@@_last_col_int > \g_@@_col_total_int }
4513 { \msg_error:nnnn { nicematrix } { Block~too~large } { #1 } { #2 } }
4514 {
4515   \hbox_set:Nn \l_@@_cell_box { #6 }
4516   \bool_if:NT \g_@@_rotate_bool \@@_rotate_cell_box:

```

Let's consider the following `{NiceTabular}`. Because of the instruction `!\hspace{1cm}` in the preamble which increases the space between the columns (by adding, in fact, that space to the previous column, that is to say the second column of the tabular), we will create *two* nodes relative to the block: the node `1-1-block` and the node `1-1-block-short`. The latter will be used by `nicematrix` to put the label of the node. The first one won't be used explicitly.

```

\begin{NiceTabular}{cc!\hspace{1cm}}c}
\Block{2-2}{our block} & & one & \\
& & two & \\
three & & four & five & \\
six & & seven & eight & \\
\end{NiceTabular}

```

We highlight the node `1-1-block`

our block		one
		two
three	four	five
six	seven	eight

We highlight the node `1-1-block-short`

our block		one
		two
three	four	five
six	seven	eight

The construction of the node corresponding to the merged cells.

```

4517 \pgfpicture
4518 \pgfrememberpicturepositiononpagetrue
4519 \pgf@relevantforpicturesizefalse
4520 \@@_qpoint:n { row - #1 }
4521 \dim_set_eq:NN \l_tmpa_dim \pgf@y

```

```

4522 \@@_qpoint:n { col - #2 }
4523 \dim_set_eq:NN \l_tmpb_dim \pgf@x
4524 \@@_qpoint:n { row - \@@_succ:n { \l_@@_last_row_int } }
4525 \dim_set_eq:NN \l_tmpc_dim \pgf@y
4526 \@@_qpoint:n { col - \@@_succ:n { \l_@@_last_col_int } }
4527 \dim_set_eq:NN \l_tmpd_dim \pgf@x

```

We construct the node for the block with the name (#1-#2-block).

The function \@@_pgf_rect_node:nnnnn takes in as arguments the name of the node and the four coordinates of two opposite corner points of the rectangle.

```

4528 \begin { pgfscope }
4529 \exp_args:Nx \pgfset { \l_@@_tikz_tl }
4530 \@@_pgf_rect_node:nnnnn
4531 { \@@_env: - #1 - #2 - block }
4532 \l_tmpb_dim \l_tmpa_dim \l_tmpd_dim \l_tmpc_dim
4533 \end { pgfscope }

```

We construct the short node.

```

4534 \dim_set_eq:NN \l_tmpb_dim \c_max_dim
4535 \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
4536 {

```

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.

```

4537 \cs_if_exist:cT
4538 { pgf @ sh @ ns @ \@@_env: - ##1 - #2 }
4539 {
4540 \seq_if_in:NnF \g_@@_multicolumn_cells_seq { ##1 - #2 }
4541 {
4542 \pgfpointanchor { \@@_env: - ##1 - #2 } { west }
4543 \dim_set:Nn \l_tmpb_dim { \dim_min:nn \l_tmpb_dim \pgf@x }
4544 }
4545 }
4546 }

```

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.

```

4547 \dim_compare:nNnT \l_tmpb_dim = \c_max_dim
4548 {
4549 \@@_qpoint:n { col - #2 }
4550 \dim_set_eq:NN \l_tmpb_dim \pgf@x
4551 }
4552 \dim_set:Nn \l_tmpd_dim { - \c_max_dim }
4553 \int_step_inline:nnn \l_@@_first_row_int \g_@@_row_total_int
4554 {
4555 \cs_if_exist:cT
4556 { pgf @ sh @ ns @ \@@_env: - ##1 - \int_use:N \l_@@_last_col_int }
4557 {
4558 \seq_if_in:NnF \g_@@_multicolumn_cells_seq { ##1 - #2 }
4559 {
4560 \pgfpointanchor
4561 { \@@_env: - ##1 - \int_use:N \l_@@_last_col_int }
4562 { east }
4563 \dim_set:Nn \l_tmpd_dim { \dim_max:nn \l_tmpd_dim \pgf@x }
4564 }
4565 }
4566 }
4567 \dim_compare:nNnT \l_tmpd_dim = { - \c_max_dim }
4568 {
4569 \@@_qpoint:n { col - \@@_succ:n { \l_@@_last_col_int } }
4570 \dim_set_eq:NN \l_tmpd_dim \pgf@x
4571 }
4572 \@@_pgf_rect_node:nnnnn
4573 { \@@_env: - #1 - #2 - block - short }
4574 \l_tmpb_dim \l_tmpa_dim \l_tmpd_dim \l_tmpc_dim

```

If the creation of the “medium nodes” is required, we create a “medium node” for the block. The function `\@@_pgf_rect_node:nnnnn` takes in as arguments the name of the node and two PGF points.

```

4575 \bool_if:NT \l_@@_medium_nodes_bool
4576 {
4577   \@@_pgf_rect_node:nnn
4578   { \@@_env: - #1 - #2 - block - medium }
4579   { \pgfpointanchor { \@@_env: - #1 - #2 - medium } { north-west } }
4580   {
4581     \pgfpointanchor
4582     { \@@_env:
4583       - \int_use:N \l_@@_last_row_int
4584       - \int_use:N \l_@@_last_col_int - medium
4585     }
4586     { south-east }
4587   }
4588 }

```

Now, we will put the label of the block beginning with the case of a `\Block` of one row.

```

4589 \int_compare:nNnTF { #1 } = { #3 }
4590 {

```

We take into account the case of a block of one row in the “first row” or the “last row”.

```

4591   \int_compare:nNnTF { #1 } = 0
4592   { \l_@@_code_for_first_row_tl }
4593   {
4594     \int_compare:nNnT { #1 } = \l_@@_last_row_int
4595     \l_@@_code_for_last_row_tl
4596   }

```

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`.

```

4597   \pgfextracty \l_tmpa_dim { \@@_qpoint:n { row - #1 - base } }

```

We retrieve (in `\pgf@x`) the x -value of the center of the block.

```

4598   \pgfpointanchor
4599   { \@@_env: - #1 - #2 - block - short }
4600   {
4601     \str_case:Vn \l_@@_pos_of_block_tl
4602     {
4603       c { center }
4604       l { west }
4605       r { east }
4606     }
4607   }

```

We put the label of the block which has been composed in `\l_@@_cell_box`.

```

4608   \pgftransformshift { \pgfpoint \pgf@x \l_tmpa_dim }
4609   \pgfset { inner~sep = \c_zero_dim }
4610   \pgfnode
4611   { rectangle }
4612   {
4613     \str_case:Vn \l_@@_pos_of_block_tl
4614     {
4615       c { base }
4616       l { base-west }
4617       r { base-east }
4618     }
4619   }
4620   { \box_use_drop:N \l_@@_cell_box } { } { }
4621 }

```

If the number of rows is different of 1, we will put the label of the block in using the short node (the label of the block has been composed in `\l_@@_cell_box`).

```

4622 {

```

If we are in the first column, we must put the block as if it was with the key `r`.

```

4623     \int_compare:nNnT \c@jCol = 0
4624     { \tl_set:Nn \l_@@_pos_of_block_tl r }
4625     \int_compare:nNnT \c@jCol = \l_@@_last_col_int
4626     { \tl_set:Nn \l_@@_pos_of_block_tl l }
4627     \pgftransformshift
4628     {
4629         \pgfpointanchor
4630         { \@@_env: - #1 - #2 - block - short }
4631         {
4632             \str_case:Vn \l_@@_pos_of_block_tl
4633             {
4634                 c { center }
4635                 l { west }
4636                 r { east }
4637             }
4638         }
4639     }
4640     \pgfset { inner-sep = \c_zero_dim }
4641     \pgfnode
4642     { rectangle }
4643     {
4644         \str_case:Vn \l_@@_pos_of_block_tl
4645         {
4646             c { center }
4647             l { west }
4648             r { east }
4649         }
4650     }
4651     { \box_use_drop:N \l_@@_cell_box } { } { }
4652 }
4653 \endpgfpicture
4654 }
4655 \group_end:
4656 }

```

How to draw the dotted lines transparently

```

4657 \cs_set_protected:Npn \@@_renew_matrix:
4658 {
4659     \RenewDocumentEnvironment { pmatrix } { } {
4660         { \pNiceMatrix }
4661         { \endpNiceMatrix }
4662     }
4663     \RenewDocumentEnvironment { vmatrix } { } {
4664         { \vNiceMatrix }
4665         { \endvNiceMatrix }
4666     }
4667     \RenewDocumentEnvironment { Vmatrix } { } {
4668         { \VNiceMatrix }
4669         { \endVNiceMatrix }
4670     }
4671     \RenewDocumentEnvironment { bmatrix } { } {
4672         { \bNiceMatrix }
4673         { \endbNiceMatrix }
4674     }
4675     \RenewDocumentEnvironment { Bmatrix } { } {
4676         { \BNiceMatrix }
4677         { \endBNiceMatrix }
4678     }
4679 }

```

Automatic arrays

```

4675 \cs_new_protected:Npn \@@_set_size:n #1-#2 \q_stop
4676 {

```

```

4677 \int_set:Nn \l_@@_nb_rows_int { #1 }
4678 \int_set:Nn \l_@@_nb_cols_int { #2 }
4679 }

4680 \NewDocumentCommand \AutoNiceMatrixWithDelims { m m O { } m O { } m ! O { } }
4681 {
4682   \int_zero_new:N \l_@@_nb_rows_int
4683   \int_zero_new:N \l_@@_nb_cols_int
4684   \@@_set_size:n #4 \q_stop
4685   \begin { NiceArrayWithDelims } { #1 } { #2 }
4686     { * { \l_@@_nb_cols_int } { c } } [ #3 , #5 , #7 ]
4687     \int_compare:nNnT \l_@@_first_row_int = 0
4688     {
4689       \int_compare:nNnT \l_@@_first_col_int = 0 { & }
4690       \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
4691       \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
4692     }
4693     \prg_replicate:nn \l_@@_nb_rows_int
4694     {
4695       \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`).

```

4696     \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { { } #6 & } #6
4697     \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
4698   }
4699   \int_compare:nNnT \l_@@_last_row_int > { -2 }
4700   {
4701     \int_compare:nNnT \l_@@_first_col_int = 0 { & }
4702     \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
4703     \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
4704   }
4705   \end { NiceArrayWithDelims }
4706 }

4707 \cs_set_protected:Npn \@@_define_com:nnn #1 #2 #3
4708 {
4709   \cs_set_protected:cpn { #1 AutoNiceMatrix }
4710   {
4711     \str_gset:Nx \g_@@_name_env_str { #1 AutoNiceMatrix }
4712     \AutoNiceMatrixWithDelims { #2 } { #3 }
4713   }
4714 }

4715 \@@_define_com:nnn p ( )
4716 \@@_define_com:nnn b [ ]
4717 \@@_define_com:nnn v | |
4718 \@@_define_com:nnn V \ | \ |
4719 \@@_define_com:nnn B \{ \}

```

We define also an command `\AutoNiceMatrix` similar to the environment `{NiceMatrix}`.

```

4720 \NewDocumentCommand \AutoNiceMatrix { O { } m O { } m ! O { } }
4721 {
4722   \group_begin:
4723     \bool_set_true:N \l_@@_NiceArray_bool
4724     \AutoNiceMatrixWithDelims . . { #2 } { #4 } [ #1 , #3 , #5 ]
4725   \group_end:
4726 }

```

The redefinition of the command `\dotfill`

```

4727 \cs_set_eq:NN \@@_old_dotfill \dotfill
4728 \cs_new_protected:Npn \@@_dotfill:
4729 {

```


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}`).

```

4730 \@@_old_dotfill
4731 \bool_if:NT \l_@@_NiceTabular_bool
4732 { \group_insert_after:N \@@_dotfill_ii: }
4733 { \group_insert_after:N \@@_dotfill_i: }
4734 }
4735 \cs_new_protected:Npn \@@_dotfill_i: { \group_insert_after:N \@@_dotfill_ii: }
4736 \cs_new_protected:Npn \@@_dotfill_ii: { \group_insert_after:N \@@_dotfill_iii: }

```

Now, if the box is not empty (unfortunately, we can’t actually test whether the box is empty and that’s why we only consider it’s width), we insert `\@@_dotfill` (which is the saved version of `\dotfill`) in the cell of the array, and it will extend, since it is no longer in `\l_@@_cell_box`.

```

4737 \cs_new_protected:Npn \@@_dotfill_iii:
4738 { \dim_compare:nNnT { \box_wd:N \l_@@_cell_box } = \c_zero_dim \@@_old_dotfill }

```

The command `\diagbox`

The command `\diagbox` will be linked to `\diagbox:nn` in the environments of `nicematrix`.

```

4739 \cs_new_protected:Npn \@@_diagbox:nn #1 #2
4740 {
4741   \tl_gput_right:Nx \g_@@_internal_code_after_tl
4742   {
4743     \@@_actually_diagbox:nnnnnn
4744     { \int_use:N \c@iRow }
4745     { \int_use:N \c@jCol }
4746     { \int_use:N \c@iRow }
4747     { \int_use:N \c@jCol }
4748     { \exp_not:n { #1 } }
4749     { \exp_not:n { #2 } }
4750   }

```

We put the cell with `\diagbox` in the sequence `\g_@@_pos_of_blocks_seq` because a cell with `\diagbox` must be considered as non empty by the key `except-corners`.

```

4751 \seq_gput_right:Nx \g_@@_pos_of_blocks_seq
4752 {
4753   { \int_use:N \c@iRow }
4754   { \int_use:N \c@jCol }
4755   { \int_use:N \c@iRow }
4756   { \int_use:N \c@jCol }
4757 }
4758 }

```

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.

```

4759 \cs_new_protected:Npn \@@_actually_diagbox:nnnnnn #1 #2 #3 #4 #5 #6
4760 {
4761   \pgfpicture
4762   \pgf@relevantforpicturesizefalse
4763   \pgfrememberpicturepositiononpagetrue
4764   \@@_qpoint:n { row - #1 }
4765   \dim_set_eq:NN \l_tmpa_dim \pgf@y
4766   \@@_qpoint:n { col - #2 }
4767   \dim_set_eq:NN \l_tmpb_dim \pgf@x
4768   \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
4769   \@@_qpoint:n { row - \@@_succ:n { #3 } }
4770   \dim_set_eq:NN \l_tmpc_dim \pgf@y
4771   \@@_qpoint:n { col - \@@_succ:n { #4 } }
4772   \dim_set_eq:NN \l_tmpd_dim \pgf@x
4773   \pgfpathlineto { \pgfpoint \l_tmpd_dim \l_tmpc_dim }
4774   {

```

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.

```

4775     \CT@arc@
4776     \pgfsetroundcap
4777     \pgfusepathqstroke
4778 }
4779 \pgfset { inner~sep = 1 pt }
4780 \pgfscope
4781 \pgftransformshift { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
4782 \pgfnode { rectangle } { south~west }
4783 { \@@_math_toggle_token: #5 \@@_math_toggle_token: } { } { }
4784 \endpgfscope
4785 \pgftransformshift { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
4786 \pgfnode { rectangle } { north~east }
4787 { \@@_math_toggle_token: #6 \@@_math_toggle_token: } { } { }
4788 \endpgfpicture
4789 }

```

The keyword `\CodeAfter`

In fact, in this subsection, we define the user command `\CodeAfter` for the case of the “normal syntax”. For the case of “light-syntax”, see the definition of the environment `{@@-light-syntax}` on p. 88.

In the environments of `nicematrix`, `\CodeAfter` will be linked to the following command `\@@_CodeAfter:`. That macro must *not* be protected since it begins with `\omit`.

```

4790 \cs_new:Npn \@@_CodeAfter: { \omit \@@_CodeAfter_i:n }

```

However, in each cell of the environment, the command `\CodeAfter` will be linked to the following command `\@@_CodeAfter_i:n` which do *not* begins with `\omit` (and thus, the user will be able to use `\CodeAfter` without error and without the need to prefix by `\omit`).

We have to catch everything until the end of the current environment (of `nicematrix`). First, we go until the next command `\end`.

```

4791 \cs_new_protected:Npn \@@_CodeAfter_i:n #1 \end
4792 {
4793   \tl_gput_right:Nn \g_nicematrix_code_after_tl { #1 }
4794   \@@_CodeAfter_ii:n
4795 }

```

We catch the argument of the command `\end` (in `#1`).

```

4796 \cs_new_protected:Npn \@@_CodeAfter_ii:n #1
4797 {

```

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.

```

4798   \str_if_eq:eeTF \@currenenv { #1 }
4799   { \end { #1 } }

```

If this is not the `\end` we are looking for, we put those tokens in `\g_nicematrix_code_after_tl` and we go on searching for the next command `\end` with a recursive call to the command `\@@_CodeAfter:n`.

```

4800   {
4801     \tl_gput_right:Nn \g_nicematrix_code_after_tl { \end { #1 } }
4802     \@@_CodeAfter_i:n
4803   }
4804 }

```

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.

```
4805 \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.

```
4806 \bool_new:N \c_@@_footnote_bool
```

```
4807 \@@_msg_new:nnn { Unknown-option-for-package }
4808 {
4809   The-option-'\l_keys_key_tl'-is-unknown. \\
4810   If-you-go-on,-it-will-be-ignored. \\
4811   For-a-list-of-the-available-options,~type-H-<return>.
4812 }
4813 {
4814   The-available-options-are~(in-alphabetic-order):~
4815   define-L-C-R,~
4816   footnote,~
4817   footnotehyper,~
4818   renew-dots,~
4819   renew-matrix-and~
4820   transparent.
4821 }
4822 \keys_define:nn { NiceMatrix / Package }
4823 {
4824   define-L-C-R .bool_set:N = \c_@@_define_L_C_R_bool ,
4825   define-L-C-R .default:n = true ,
4826   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
4827   renew-dots .value_forbidden:n = true ,
4828   renew-matrix .code:n = \@@_renew_matrix: ,
4829   renew-matrix .value_forbidden:n = true ,
4830   transparent .meta:n = { renew-dots , renew-matrix } ,
4831   transparent .value_forbidden:n = true ,
4832   footnote .bool_set:N = \c_@@_footnote_bool ,
4833   footnotehyper .bool_set:N = \c_@@_footnotehyper_bool ,
4834   unknown .code:n = \@@_error:n { Unknown-option-for-package }
4835 }
4836 \ProcessKeysOptions { NiceMatrix / Package }
```

```
4837 \@@_msg_new:nn { footnote-with-footnotehyper-package }
4838 {
4839   You-can't-use-the-option-'footnote'~because-the-package~
4840   footnotehyper~has~already~been~loaded.~
4841   If-you-want,~you-can-use-the-option-'footnotehyper'~and-the-footnotes~
4842   within-the-environments-of-nicematrix-will-be-extracted-with-the-tools~
4843   of-the-package-footnotehyper.\\
4844   If-you-go-on,~the-package-footnote-won't-be-loaded.
4845 }
4846 \@@_msg_new:nn { footnotehyper-with-footnote-package }
4847 {
4848   You-can't-use-the-option-'footnotehyper'~because-the-package~
4849   footnote~has~already~been~loaded.~
4850   If-you-want,~you-can-use-the-option-'footnote'~and-the-footnotes~
4851   within-the-environments-of-nicematrix-will-be-extracted-with-the-tools~
4852   of-the-package-footnote.\\
```

```

4853   If-you-go-on,~the-package-footnotehyper~won't~be~loaded.
4854 }

4855 \bool_if:NT \c_@@_footnote_bool
4856 {
4857   \@ifclassloaded { beamer }
4858   { \msg_info:nn { nicematrix } { Option~incompatible~with~Beamer } }
4859   {
4860     \@ifpackageloaded { footnotehyper }
4861     { \@@_error:n { footnote~with~footnotehyper~package } }
4862     { \usepackage { footnote } }
4863   }
4864 }

4865 \bool_if:NT \c_@@_footnotehyper_bool
4866 {
4867   \@ifclassloaded { beamer }
4868   { \@@_info:n { Option~incompatible~with~Beamer } }
4869   {
4870     \@ifpackageloaded { footnote }
4871     { \@@_error:n { footnotehyper~with~footnote~package } }
4872     { \usepackage { footnotehyper } }
4873   }
4874   \bool_set_true:N \c_@@_footnote_bool
4875 }

```

The flag `\c_@@_footnote_bool` is raised and so, we will only have to test `\c_@@_footnote_bool` in order to know if we have to insert an environment `{savenotes}`.

Error messages of the package

The following command converts all the elements of a sequence (which are token lists) into strings.

```

4876 \cs_new_protected:Npn \@@_convert_to_str_seq:N #1
4877 {
4878   \seq_clear:N \l_tmpa_seq
4879   \seq_map_inline:Nn #1
4880   {
4881     \seq_put_left:Nx \l_tmpa_seq { \tl_to_str:n { ##1 } }
4882   }
4883   \seq_set_eq:NN #1 \l_tmpa_seq
4884 }

```

The following command creates a sequence of strings (`str`) from a `clist`.

```

4885 \cs_new_protected:Npn \@@_set_seq_of_str_from_clist:Nn #1 #2
4886 {
4887   \seq_set_from_clist:Nn #1 { #2 }
4888   \@@_convert_to_str_seq:N #1
4889 }

4890 \@@_set_seq_of_str_from_clist:Nn \c_@@_types_of_matrix_seq
4891 {
4892   NiceMatrix ,
4893   pNiceMatrix , bNiceMatrix , vNiceMatrix, BNiceMatrix, VNiceMatrix
4894 }

```

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`.

```

4895 \cs_new_protected:Npn \@@_error_too_much_cols:
4896 {
4897   \seq_if_in:NVTF \c_@@_types_of_matrix_seq \g_@@_name_env_str

```

```

4898     {
4899         \int_compare:nNnTF \l_@@_last_col_int = { -2 }
4900         { \@@_fatal:n { too-much-cols-for-matrix } }
4901         {
4902             \bool_if:NF \l_@@_last_col_without_value_bool
4903             { \@@_fatal:n { too-much-cols-for-matrix-with-last-col } }
4904         }
4905     }
4906     { \@@_fatal:n { too-much-cols-for-array } }
4907 }

```

The following command must *not* be protected since it's used in an error message.

```

4908 \cs_new:Npn \@@_message_hdotsfor:
4909 {
4910     \tl_if_empty:VF \g_@@_HVdotsfor_lines_tl
4911     { ~Maybe-your-use-of~\token_to_str:N \Hdotsfor\ is-incorrect.}
4912 }
4913 \@@_msg_new:nn { too-much-cols-for-matrix-with-last-col }
4914 {
4915     You-try-to-use-more-columns-than-allowed-by-your~
4916     \@@_full_name_env:.\@@_message_hdotsfor:\ The-maximal-number-of~
4917     columns-is~\int_eval:n { \l_@@_last_col_int - 1 }~(plus-the~
4918     exterior-columns).~This-error-is-fatal.
4919 }
4920 \@@_msg_new:nn { too-much-cols-for-matrix }
4921 {
4922     You-try-to-use-more-columns-than-allowed-by-your~
4923     \@@_full_name_env:.\@@_message_hdotsfor:\ Recall-that-the-maximal~
4924     number-of-columns-for-a-matrix-is-fixed-by-the-LaTeX-counter~
4925     'MaxMatrixCols'.~Its-actual-value-is~\int_use:N \c@MaxMatrixCols.~
4926     This-error-is-fatal.
4927 }

```

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`.

```

4928 \@@_msg_new:nn { too-much-cols-for-array }
4929 {
4930     You-try-to-use-more-columns-than-allowed-by-your~
4931     \@@_full_name_env:.\@@_message_hdotsfor:\ The-maximal-number-of-columns-is~
4932     \int_use:N \g_@@_static_num_of_col_int\
4933     ~~~~~(plus-the-potential-exterior-ones).~
4934     This-error-is-fatal.
4935 }
4936 \@@_msg_new:nn { last-col-not-used }
4937 {
4938     The-key~'last-col'~is~in-force-but-you-have-not-used-that-last-column~
4939     in-your~\@@_full_name_env:~.~However,~you-can-go-on.
4940 }
4941 \@@_msg_new:nn { columns-not-used }
4942 {
4943     The-preamble-of-your~\@@_full_name_env:\ announces~\int_use:N
4944     \g_@@_static_num_of_col_int\
4945     columns-but-you-use-only~\int_use:N \c@jCol.\\
4946     However,~you-can-go-on.
4947 }
4948 \@@_msg_new:nn { in-first-col }
4949 {
4950     You-can't-use-the-command~#1 in-the-first-column-(number~0)-of-the-array.\\
4951     If-you-go-on,~this-command-will-be-ignored.
4952 }

```

```

4953 \@@_msg_new:nn { in-last-col }
4954 {
4955     You~can't~use~the~command~#1~in~the~last~column~(exterior)~of~the~array.\\
4956     If~you~go~on,~this~command~will~be~ignored.
4957 }
4958 \@@_msg_new:nn { in-first-row }
4959 {
4960     You~can't~use~the~command~#1~in~the~first~row~(number~0)~of~the~array.\\
4961     If~you~go~on,~this~command~will~be~ignored.
4962 }
4963 \@@_msg_new:nn { in-last-row }
4964 {
4965     You~can't~use~the~command~#1~in~the~last~row~(exterior)~of~the~array.\\
4966     If~you~go~on,~this~command~will~be~ignored.
4967 }
4968 \@@_msg_new:nn { bad-option-for-line-style }
4969 {
4970     Since~you~haven't~loaded~Tikz,~the~only~value~you~can~give~to~'line-style'~
4971     is~'standard'.~If~you~go~on,~this~option~will~be~ignored.
4972 }
4973 \@@_msg_new:nn { Unknown-option-for-xdots }
4974 {
4975     As~for~now~there~is~only~three~options~available~here:~'color',~'line-style'~
4976     and~'shorten'~(and~you~try~to~use~'\l_keys_key_tl').~If~you~go~on,~
4977     this~option~will~be~ignored.
4978 }
4979 \@@_msg_new:nn { Unknown-option-for-rowcolors }
4980 {
4981     As~for~now~there~is~only~one~option~available~here:~'respect-blocks'~
4982     (and~you~try~to~use~'\l_keys_key_tl').~If~you~go~on,~
4983     this~option~will~be~ignored.
4984 }
4985 \@@_msg_new:nn { ampersand-in-light-syntax }
4986 {
4987     You~can't~use~an~ampersand~(\token_to_str &)~to~separate~columns~because
4988     ~you~have~used~the~option~'light-syntax'.~This~error~is~fatal.
4989 }
4990 \@@_msg_new:nn { double-backslash-in-light-syntax }
4991 {
4992     You~can't~use~\token_to_str:N~\\~to~separate~rows~because~you~have~used~
4993     the~option~'light-syntax'.~You~must~use~the~character~'\l_@@_end_of_row_tl'~
4994     (set~by~the~option~'end-of-row').~This~error~is~fatal.
4995 }
4996 \@@_msg_new:nn { standard-cline-in-document }
4997 {
4998     The~key~'standard-cline'~is~available~only~in~the~preamble.\\
4999     If~you~go~on~this~command~will~be~ignored.
5000 }
5001 \@@_msg_new:nn { bad-value-for-baseline }
5002 {
5003     The~value~given~to~'baseline'~(\int_use:N \l_tmpa_int)~is~not~
5004     valid.~The~value~must~be~between~\int_use:N \l_@@_first_row_int\ and~
5005     \int_use:N \g_@@_row_total_int\ or~equal~to~'t',~'c'~or~'b'.\\
5006     If~you~go~on,~a~value~of~1~will~be~used.
5007 }
5008 \@@_msg_new:nn { empty-environment }
5009 { Your~\@@_full_name_env:\ is~empty.~This~error~is~fatal. }
5010 \@@_msg_new:nn { unknown-cell-for-line-in-code-after }
5011 {

```

```

5012 Your~command~\token_to_str:N\line\{#1\}\{#2\}~in~the~'code~after'~
5013 can't~be~executed~because~a~cell~doesn't~exist.\\
5014 If~you~go~on~this~command~will~be~ignored.
5015 }
5016 \@@_msg_new:nn { Hdotsfor~in~col-0 }
5017 {
5018   You~can't~use~\token_to_str:N \Hdotsfor\ in~an~exterior~column~of~
5019   the~array.~If~you~go~on,~the~corresponding~dotted~line~won't~be~drawn.
5020 }
5021 \@@_msg_new:nn { bad~corner }
5022 {
5023   #1~is~an~incorrect~specification~for~a~corner~(in~the~keys~
5024   'except~corners'~and~'hlines~except~corners').~The~available~
5025   values~are:~NW,~SW,~NE~and~SE.\\
5026   If~you~go~on,~this~specification~of~corner~will~be~ignored.
5027 }
5028 \@@_msg_new:nn { last~col~non~empty~for~NiceArray }
5029 {
5030   In~the~\@@_full_name_env:,~you~must~use~the~option~
5031   'last~col'~without~value.\\
5032   However,~you~can~go~on~for~this~time~
5033   (the~value~'\l_keys_value_tl'~will~be~ignored).
5034 }
5035 \@@_msg_new:nn { last~col~non~empty~for~NiceMatrixOptions }
5036 {
5037   In~\NiceMatrixoptions,~you~must~use~the~option~
5038   'last~col'~without~value.\\
5039   However,~you~can~go~on~for~this~time~
5040   (the~value~'\l_keys_value_tl'~will~be~ignored).
5041 }
5042 \@@_msg_new:nn { Block~too~large }
5043 {
5044   You~try~to~draw~a~block~in~the~cell~#1~#2~of~your~matrix~but~the~matrix~is~
5045   too~small~for~that~block. \\
5046 }
5047 \@@_msg_new:nn { unknown~column~type }
5048 {
5049   The~column~type~'#1'~in~your~\@@_full_name_env:\
5050   is~unknown. \\
5051   This~error~is~fatal.
5052 }
5053 \@@_msg_new:nn { tabularnote~forbidden }
5054 {
5055   You~can't~use~the~command~\token_to_str:N\tabularnote\
5056   ~in~a~\@@_full_name_env:~This~command~is~available~only~in~
5057   \{NiceTabular\},~\{NiceArray\}~and~\{NiceMatrix\}. \\
5058   If~you~go~on,~this~command~will~be~ignored.
5059 }
5060 \@@_msg_new:nn { bottomule~without~booktabs }
5061 {
5062   You~can't~use~the~option~'tabular/bottomrule'~because~you~haven't~
5063   loaded~'booktabs'.\\
5064   If~you~go~on,~this~option~will~be~ignored.
5065 }
5066 \@@_msg_new:nn { enumitem~not~loaded }
5067 {
5068   You~can't~use~the~command~\token_to_str:N\tabularnote\
5069   ~because~you~haven't~loaded~'enumitem'.\\
5070   If~you~go~on,~this~command~will~be~ignored.
5071 }

```

```

5072 \@@_msg_new:nn { Wrong-last-row }
5073 {
5074   You~have~used~'last-row=\int_use:N \l_@@_last_row_int'~but~your~
5075   \@@_full_name_env:\ seems~to~have~\int_use:N \c@iRow \ rows.~
5076   If~you~go~on,~the~value~of~\int_use:N \c@iRow \ will~be~used~for~
5077   last-row.~You~can~avoid~this~problem~by~using~'last-row'~
5078   without~value~(more~compilations~might~be~necessary).
5079 }
5080 \@@_msg_new:nn { Yet-in-env }
5081 { Environments~of~nicematrix~can't~be~nested.\\ This~error~is~fatal. }
5082 \@@_msg_new:nn { Outside-math-mode }
5083 {
5084   The~\@@_full_name_env:\ can~be~used~only~in~math-mode~
5085   (and~not~in~\token_to_str:N \vcenter).\\
5086   This~error~is~fatal.
5087 }
5088 \@@_msg_new:nn { Bad-value-for-letter-for-dotted-lines }
5089 {
5090   The~value~of~key~'\l_keys_key_tl'~must~be~of~length~1.\\
5091   If~you~go~on,~it~will~be~ignored.
5092 }
5093 \@@_msg_new:nnn { Unknown-key-for-Block }
5094 {
5095   The~key~'\l_keys_key_tl'~is~unknown~for~the~command~\token_to_str:N
5096   \Block.\\ If~you~go~on,~it~will~be~ignored. \\
5097   For~a~list~of~the~available~keys,~type~H~<return>.
5098 }
5099 {
5100   The~available~options~are~(in~alphabetic~order):~,~c,~
5101   color,~l,~and~r.
5102 }
5103 \@@_msg_new:nnn { Unknown-key-for-notes }
5104 {
5105   The~key~'\l_keys_key_tl'~is~unknown.\\
5106   If~you~go~on,~it~will~be~ignored. \\
5107   For~a~list~of~the~available~keys~about~notes,~type~H~<return>.
5108 }
5109 {
5110   The~available~options~are~(in~alphabetic~order):~
5111   bottomrule,~
5112   code-after,~
5113   code-before,~
5114   enumitem-keys,~
5115   enumitem-keys-para,~
5116   para,~
5117   label-in-list,~
5118   label-in-tabular~and~
5119   style.
5120 }
5121 \@@_msg_new:nnn { Unknown-key-for-NiceMatrixOptions }
5122 {
5123   The~key~'\l_keys_key_tl'~is~unknown~for~the~command~
5124   \token_to_str:N \NiceMatrixOptions. \\
5125   If~you~go~on,~it~will~be~ignored. \\
5126   For~a~list~of~the~*principal*~available~keys,~type~H~<return>.
5127 }
5128 {
5129   The~available~options~are~(in~alphabetic~order):~
5130   allow-duplicate-names,~
5131   cell-space-bottom-limit,~
5132   cell-space-top-limit,~

```



```

5133     code-for-first-col,~
5134     code-for-first-row,~
5135     code-for-last-col,~
5136     code-for-last-row,~
5137     create-extra-nodes,~
5138     create-medium-nodes,~
5139     create-large-nodes,~
5140     end-of-row,~
5141     first-col,~
5142     first-row,~
5143     hlines,~
5144     hvlines,~
5145     hvlines-except-corners,~
5146     last-col,~
5147     last-row,~
5148     left-margin,~
5149     letter-for-dotted-lines,~
5150     light-syntax,~
5151     notes~(several subkeys),~
5152     nullify-dots,~
5153     renew-dots,~
5154     renew-matrix,~
5155     right-margin,~
5156     small,~
5157     transparent,~
5158     vlines,~
5159     xdots/color,~
5160     xdots/shorten~and~
5161     xdots/line-style.
5162 }

5163 \@@_msg_new:nnn { Unknown~option~for~NiceArray }
5164 {
5165     The~option~'\l_keys_key_tl'~is~unknown~for~the~environment~
5166     \{NiceArray\}. \\
5167     If~you~go~on,~it~will~be~ignored. \\
5168     For~a~list~of~the~*principal*~available~options,~type~H~<return>.
5169 }
5170 {
5171     The~available~options~are~(in~alphabetic~order):~
5172     b,~
5173     baseline,~
5174     c,~
5175     cell-space-bottom-limit,~
5176     cell-space-top-limit,~
5177     code-after,~
5178     code-for-first-col,~
5179     code-for-first-row,~
5180     code-for-last-col,~
5181     code-for-last-row,~
5182     colortbl-like,~
5183     columns-width,~
5184     create-extra-nodes,~
5185     create-medium-nodes,~
5186     create-large-nodes,~
5187     extra-left-margin,~
5188     extra-right-margin,~
5189     first-col,~
5190     first-row,~
5191     hlines,~
5192     hvlines,~
5193     hvlines-except-corners,~
5194     last-col,~
5195     last-row,~

```

```

5196 left-margin,~
5197 light-syntax,~
5198 name,~
5199 notes/bottomrule,~
5200 notes/para,~
5201 nullify-dots,~
5202 renew-dots,~
5203 right-margin,~
5204 rules/color,~
5205 rules/width,~
5206 small,~
5207 t,~
5208 vlines,~
5209 xdots/color,~
5210 xdots/shorten~and~
5211 xdots/line-style.
5212 }

```

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`).

```

5213 \@@_msg_new:nnn { Unknown-option-for-NiceMatrix }
5214 {
5215   The~option~'\l_keys_key_tl'~is~unknown~for~the~
5216   \@@_full_name_env:. \\\
5217   If~you~go~on,~it~will~be~ignored. \\\
5218   For~a~list~of~the~*principal*~available~options,~type~H~<return>.
5219 }
5220 {
5221   The~available~options~are~(in~alphabetic~order):~
5222   b,~
5223   baseline,~
5224   c,~
5225   cell-space-bottom-limit,~
5226   cell-space-top-limit,~
5227   code-after,~
5228   code-for-first-col,~
5229   code-for-first-row,~
5230   code-for-last-col,~
5231   code-for-last-row,~
5232   colortbl-like,~
5233   columns-width,~
5234   create-extra-nodes,~
5235   create-medium-nodes,~
5236   create-large-nodes,~
5237   extra-left-margin,~
5238   extra-right-margin,~
5239   first-col,~
5240   first-row,~
5241   hlines,~
5242   hvlines,~
5243   hvlines-except-corners,~
5244   l,~
5245   last-col,~
5246   last-row,~
5247   left-margin,~
5248   light-syntax,~
5249   name,~
5250   nullify-dots,~
5251   r,~
5252   renew-dots,~
5253   right-margin,~
5254   rules/color,~
5255   rules/width,~

```

```

5256     small,~
5257     t,~
5258     vlines,~
5259     xdots/color,~
5260     xdots/shorten~and~
5261     xdots/line-style.
5262 }

5263 \@@_msg_new:nnn { Unknown~option~for~NiceTabular }
5264 {
5265     The~option~'\l_keys_key_tl'~is~unknown~for~the~environment~
5266     \{NiceTabular\}. \\
5267     If~you~go~on,~it~will~be~ignored. \\
5268     For~a~list~of~the~*principal*~available~options,~type~H~<return>.
5269 }
5270 {
5271     The~available~options~are~(in~alphabetic~order):~
5272     b,~
5273     baseline,~
5274     c,~
5275     cell-space-bottom-limit,~
5276     cell-space-top-limit,~
5277     code-after,~
5278     code-for-first-col,~
5279     code-for-first-row,~
5280     code-for-last-col,~
5281     code-for-last-row,~
5282     colortbl-like,~
5283     columns-width,~
5284     create-extra-nodes,~
5285     create-medium-nodes,~
5286     create-large-nodes,~
5287     extra-left-margin,~
5288     extra-right-margin,~
5289     first-col,~
5290     first-row,~
5291     hlines,~
5292     hvlines,~
5293     hvlines-except-corners,~
5294     last-col,~
5295     last-row,~
5296     left-margin,~
5297     light-syntax,~
5298     name,~
5299     notes/bottomrule,~
5300     notes/para,~
5301     nullify-dots,~
5302     renew-dots,~
5303     right-margin,~
5304     rules/color,~
5305     rules/width,~
5306     t,~
5307     vlines,~
5308     xdots/color,~
5309     xdots/shorten~and~
5310     xdots/line-style.
5311 }

5312 \@@_msg_new:nnn { Duplicate-name }
5313 {
5314     The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
5315     the~same~environment~name~twice.~You~can~go~on,~but,~
5316     maybe,~you~will~have~incorrect~results~especially~
5317     if~you~use~'columns-width=auto'.~If~you~don't~want~to~see~this~
5318     message~again,~use~the~option~'allow-duplicate-names'~in~

```

```

5319     '\token_to_str:N \NiceMatrixOptions'.\\
5320     For~a~list~of~the~names~already~used,~type~H~<return>. \\
5321   }
5322   {
5323     The~names~already~defined~in~this~document~are:~
5324     \seq_use:Nnnn \g_@@_names_seq { ,~ } { ,~ } { ~and~ }.
5325   }

5326 \@@_msg_new:nn { Option~auto~for~columns~width }
5327   {
5328     You~can't~give~the~value~'auto'~to~the~option~'columns~width'~here.~
5329     If~you~go~on,~the~option~will~be~ignored.
5330   }

```

18 History

The successive versions of the file `nicematrix.sty` provided by TeXLive are available on the SVN server of TeXLive:

<https://www.tug.org/svn/texlive/trunk/Master/texmf-dist/tex/latex/nicematrix/nicematrix.sty>

Changes between versions 1.0 and 1.1

The dotted lines are no longer drawn with Tikz nodes but with Tikz circles (for efficiency).
Modification of the code which is now twice faster.

Changes between versions 1.1 and 1.2

New environment `{NiceArray}` with column types L, C and R.

Changes between version 1.2 and 1.3

New environment `{pNiceArrayC}` and its variants.

Correction of a bug in the definition of `{BNiceMatrix}`, `{vNiceMatrix}` and `{VNiceMatrix}` (in fact, it was a typo).

Options are now available locally in `{pNiceMatrix}` and its variants.

The names of the options are changed. The old names were names in “camel style”.

Changes between version 1.3 and 1.4

The column types `w` and `W` can now be used in the environments `{NiceArray}`, `{pNiceArrayC}` and its variants with the same meaning as in the package `array`.

New option `columns-width` to fix the same width for all the columns of the array.

Changes between version 1.4 and 2.0

The versions 1.0 to 1.4 of `nicematrix` were focused on the continuous dotted lines whereas the version 2.0 of `nicematrix` provides different features to improve the typesetting of mathematical matrices.

Changes between version 2.0 and 2.1

New implementation of the environment `{pNiceArrayRC}`. With this new implementation, there is no restriction on the width of the columns.

The package `nicematrix` no longer loads `mathtools` but only `amsmath`.

Creation of “medium nodes” and “large nodes”.

Changes between version 2.1 and 2.1.1

Small corrections: for example, the option `code-for-first-row` is now available in the command `\NiceMatrixOptions`.

Following a discussion on TeX StackExchange⁴⁶, Tikz externalization is now deactivated in the environments of the package `nicematrix`.⁴⁷

Changes between version 2.1.2 and 2.1.3

When searching the end of a dotted line from a command like `\Cdots` issued in the “main matrix” (not in the exterior column), the cells in the exterior column are considered as outside the matrix. That means that it’s possible to do the following matrix with only a `\Cdots` command (and a single `\Vdots`).

$$\left(\begin{array}{ccc} & C_j & \\ 0 & \vdots & 0 \\ & a & \cdots \\ 0 & & 0 \end{array} \right) L_i$$

Changes between version 2.1.3 and 2.1.4

Replacement of some options `0 { }` in commands and environments defined with `xparse` by `! 0 { }` (because a recent version of `xparse` introduced the specifier `!` and modified the default behaviour of the last optional arguments).

See www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end

Changes between version 2.1.4 and 2.1.5

Compatibility with the classes `revtex4-1` and `revtex4-2`.

Option `allow-duplicate-names`.

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`.

⁴⁶cf. tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package

⁴⁷Before this version, there was an error when using `nicematrix` with Tikz externalization. In any case, it’s not possible to externalize the Tikz elements constructed by `nicematrix` because they use the options `overlay` and `remember picture`.

Changes between version 2.2.1 and 2.3

Compatibility with the column type `S` of `siunitx`.

Option `hlines`.

Changes between version 2.3 and 3.0

Modification of `\Hdotsfor`. Now `\Hdotsfor` erases the `\vlines` (of “`|`”) as `\hdotsfor` does.

Composition of exterior rows and columns on the four sides of the matrix (and not only on two sides) with the options `first-row`, `last-row`, `first-col` and `last-col`.

Changes between version 3.0 and 3.1

Command `\Block` to draw block matrices.

Error message when the user gives an incorrect value for `last-row`.

A dotted line can no longer cross another dotted line (excepted the dotted lines drawn by `\cdottedline`, the symbol “`:`” (in the preamble of the array) and `\line` in `code-after`).

The starred versions of `\Cdots`, `\Ldots`, etc. are now deprecated because, with the new implementation, they become pointless. These starred versions are no longer documented.

The vertical rules in the matrices (drawn by “`|`”) are now compatible with the color fixed by `colortbl`.

Correction of a bug: it was not possible to use the colon “`:`” in the preamble of an array when `pdflatex` was used with `french-babel` (because `french-babel` activates the colon in the beginning of the document).

Changes between version 3.1 and 3.2 (and 3.2a)

Option `small`.

Changes between version 3.2 and 3.3

The options `first-row`, `last-row`, `first-col` and `last-col` are now available in the environments `{NiceMatrix}`, `{pNiceMatrix}`, `{bNiceMatrix}`, etc.

The option `columns-width=auto` doesn't need any more a second compilation.

The options `renew-dots`, `renew-matrix` and `transparent` are now available as package options (as said in the documentation).

The previous version of `nicematrix` was incompatible with a recent version of `expl3` (released 2019/09/30). This version is compatible.

Changes between version 3.3 and 3.4

Following a discussion on TeX StackExchange⁴⁸, 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.

⁴⁸cf. tex.stackexchange.com/questions/510841/nicematrix-and-tikz-external-optimize

Changes between version 3.5 and 3.6

LaTeX counters `iRow` and `jCol` available in the cells of the array.

Addition of `\normalbaselines` before the construction of the array: in environments like `{align}` of `amsmath` the value of `\baselineskip` is changed and if the options `first-row` and `last-row` were used in an environment of `nicematrix`, the position of the delimiters was wrong.

A warning is written in the `.log` file if an obsolete environment is used.

There is no longer artificial errors `Duplicate~name` in the environments of `amsmath`.

Changes between version 3.6 and 3.7

The four “corners” of the matrix are correctly protected against the four codes: `code-for-first-col`, `code-for-last-col`, `code-for-first-row` and `code-for-last-row`.

New command `\pAutoNiceMatrix` and its variants (suggestion of Christophe Bal).

Changes between version 3.7 and 3.8

New programming for the command `\Block` when the block has only one row. With this programming, the vertical rules drawn by the specifier “|” at the end of the block is actually drawn. In previous versions, they were not because the block of one row was constructed with `\multicolumn`. An error is raised when an obsolete environment is used.

Changes between version 3.8 and 3.9

New commands `\NiceMatrixLastEnv` and `\OnlyMainNiceMatrix`.

New options `create-medium-nodes` and `create-large-nodes`.

Changes between version 3.9 and 3.10

New option `light-syntax` (and `end-of-row`).

New option `dotted-lines-margin` for fine tuning of the dotted lines.

Changes between versions 3.10 and 3.11

Correction of a bug linked to `first-row` and `last-row`.

Changes between versions 3.11 and 3.12

Command `\rotate` in the cells of the array.

Options `vlines`, `hlines` and `hvlines`.

Option `baseline` pour `{NiceArray}` (not for the other environments).

The name of the Tikz nodes created by the command `\Block` has changed: when the command has been issued in the cell $i-j$, the name is $i-j$ -block and, if the creation of the “medium nodes” is required, a node $i-j$ -block-medium is created.

If the user try to use more columns than allowed by its environment, an error is raised by `nicematrix` (instead of a low-level error).

The package must be loaded with the option `obsolete-environments` if we want to use the deprecated environments.

Changes between versions 3.12 and 3.13

The behaviour of the command `\rotate` is improved when used in the “last row”.

The option `dotted-lines-margin` has been renamed in `xdots/shorten` and the options `xdots/color` and `xdots/line-style` have been added for a complete customization of the dotted lines.

In the environments without preamble (`{NiceMatrix}`, `{pNiceMatrix}`, etc.), it’s possible to use the options `l` (`=L`) or `r` (`=R`) to specify the type of the columns.

The starred versions of the commands `\Cdots`, `\Ldots`, `\Vdots`, `\Ddots` and `\Iddots` are deprecated since the version 3.1 of `nicematrix`. Now, one should load `nicematrix` with the option `starred-commands` to avoid an error at the compilation.

The code of `nicematrix` no longer uses Tikz but only PGF. By default, Tikz is *not* loaded by `nicematrix`.

Changes between versions 3.13 and 3.14

Correction of a bug (question 60761504 on [stackoverflow](#)).

Better error messages when the user uses `&` or `\\` when `light-syntax` is in force.

Changes between versions 3.14 and 3.15

It’s possible to put labels on the dotted lines drawn by `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots`, `\Hdotsfor` and the command `\line` in the `code-after` with the tokens `_` and `^`.

The option `baseline` is now available in all the environments of `nicematrix`. Before, it was available only in `{NiceArray}`.

New keyword `\CodeAfter` (in the environments of `nicematrix`).

Changes between versions 3.15 and 4.0

New environment `{NiceTabular}`

Commands to color cells, row and columns with a perfect result in the PDF.

Changes between versions 4.0 and 4.1

New keys `cell-space-top-limit` and `cell-space-bottom-limit`

New command `\diagbox`

The key `hvline` don’t draw rules in the blocks (commands `\Block`) and in the virtual blocks corresponding to the dotted lines.

Changes between versions 4.1 and 4.2

It’s now possible to write `\begin{pNiceMatrix}a&b\\c&d\end{pNiceMatrix}^2` with the expected result.

Changes between versions 4.2 and 4.3

The horizontal centering of the content of a `\Block` is correct even when an instruction such as `!\quad` is used in the preamble of the array.

It’s now possible to use the command `\Block` in the “last row”.

Changes between versions 4.3 and 4.4

New key `hvlines-except-corners`.

Changes between versions 4.4 and 5.0

Use of the standard column types `l`, `c` and `r` instead of `L`, `C` and `R`.

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

Command `\tabularnote`

Changes between versions 5.0 and 5.1

The vertical rules specified by `|` in the preamble are not broken by `\hline\hline` (and other).

Environment `{NiceTabular*}`

Command `\Vdotsfor` similar to `\Hdotsfor`

The variable `\g_nicematrix_code_after_tl` is now public.

Changes between versions 5.1 and 5.2

The vertical rules specified by `|` or `||` in the preamble respect the blocks.

Key `respect-blocks` for `\rowcolors` (with a `s`) in the `code-before`.

The variable `\g_nicematrix_code_before_tl` is now public.

The key `baseline` can take in as value of the form `line-i` to align the `\hline` in the row `i`.

The key `hvlines-except-corners` may take in as value a list of corners (eg: `NW,SE`).

Changes between versions 5.2 and 5.3

Keys `c`, `r` and `l` for the command `\Block`.

It's possible to use the key `draw-first` with `\Ddots` and `\Iddots` to specify which dotted line will be drawn first (the other lines will be drawn parallel to that one if parallelization is activated).

Changes between versions 5.3 and 5.4

Key `tabularnote`.

Different behaviour for the mono-column blocks.

Changes between versions 5.4 and 5.5

The user must never put `\omit` before `\CodeAfter`.

Correction of a bug: the tabular notes `\tabularnotes` were not composed when present in a block (except a mono-column block).

Changes between versions 5.5 and 5.6

Different behaviour for the mono-row blocks.

New command `\NotEmpty`.

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	
<code>@@</code> commands:	<code>\@@_Block_iii:nnnnnn</code> 4474, 4476
<code>\@@_Block:</code> 1064, 4283	<code>\@@_Block_iv:nnnnn</code> 4313, 4417
<code>\@@_Block_i</code> 4285, 4288	<code>\@@_Cdots</code> 989, 1054, 2952
<code>\@@_Block_ii:nnnnn</code> 4288, 4289	<code>\g_@@_Cdots_lines_tl</code> 1081, 2283
<code>\@@_Block_iii:nnnnn</code> 4312, 4316	<code>\@@_Cell:</code> 194, 752, 1447, 1494, 1511, 2071, 3052, 3053, 3054, 3055,

3056, 3057, 3058, 3059, 3060, 3061, 3062, 3063	\l_@@_block_auto_columns_width_bool 1116, 1867, 4043, 4048, 4058, 4068
\@@_CodeAfter: 1068, 4790	\g_@@_block_box_int 285, 1096, 4318, 4332, 4334, 4370, 4382, 4394, 4403, 4413
\@@_CodeAfter_i:n 754, 4790, 4791, 4802	\g_@@_blocks_dp_dim 234, 839, 842, 843, 4397, 4400
\@@_CodeAfter_ii:n 4794, 4796	\g_@@_blocks_ht_dim 233, 845, 848, 849, 4388, 4391
\@@_Ddots 991, 1056, 2982	\g_@@_blocks_seq 277, 1118, 1613, 4407, 4419, 4474
\g_@@_Ddots_lines_tl 1084, 2281	\g_@@_blocks_wd_dim 232, 833, 836, 837, 4376, 4379
\g_@@_HVDotsfor_lines_tl 1086, 2279, 3102, 3178, 4910	\c_@@_booktabs_loaded_bool 24, 30, 1004, 1645
\@@_Hdotsfor: 994, 1061, 3087	\l_@@_cell_box 759, 805, 807, 813, 819, 822, 826, 835, 836, 841, 842, 847, 848, 858, 859, 860, 861, 863, 866, 870, 872, 890, 1006, 1193, 1195, 1510, 1521, 1953, 1977, 1980, 1982, 1998, 2021, 2025, 4515, 4620, 4651, 4738
\@@_Hdotsfor:nmmn 3104, 3116	\l_@@_cell_space_bottom_limit_dim 417, 484, 861
\@@_Hdotsfor_i 3093, 3100	\l_@@_cell_space_top_limit_dim 416, 482, 859
\@@_Hline: 1059, 3817	\l_@@_cell_type_tl 230, 231, 1447, 1512, 4291, 4293
\@@_Hline_i:n 3817, 3818, 3824	\@@_cellcolor 1176, 3380, 3530, 3531
\@@_Hline_ii:nn 3821, 3824	\@@_cellcolor_tabular 998, 3536
\@@_Hline_iii:n 3822, 3825	\g_@@_cells_seq 1789, 1790, 1791, 1793
\@@_Hspace: 1060, 3035	\@@_chessboardcolors 1181, 3523
\@@_Iddots 992, 1057, 3005	\@@_cline 129, 1052
\g_@@_Iddots_lines_tl 1085, 2282	\@@_cline_i:nn 130, 131, 143, 146
\@@_Ldots 988, 993, 1053, 2937	\@@_cline_i:w 131, 132
\g_@@_Ldots_lines_tl 1082, 2284	\l_@@_code_before_bool 266, 547, 574, 940, 1123, 1133, 1809, 1826, 1844, 1875, 1901, 1928, 2150, 2242
\l_@@_Matrix_bool 240, 1286, 1360, 2062	\l_@@_code_before_tl 265, 546, 1124, 1183
\l_@@_NiceArray_bool 237, 334, 1127, 1238, 1298, 1391, 1403, 2038, 3696, 3697, 3813, 3814, 3994, 4001, 4723	\l_@@_code_for_first_col_tl 496, 1964
\g_@@_NiceMatrixBlock_int 228, 4055, 4060, 4063, 4074	\l_@@_code_for_first_row_tl 500, 769, 4592
\l_@@_NiceTabular_bool 150, 238, 760, 925, 1103, 1182, 1184, 1330, 1334, 1392, 1404, 2091, 2100, 4337, 4426, 4731	\l_@@_code_for_last_col_tl 498, 2007
\@@_NotEmpty: 1070, 2085	\l_@@_code_for_last_row_tl 502, 776, 4595
\@@_OnlyMainNiceMatrix:n 1066, 3559	\g_@@_col_total_int 758, 1077, 1279, 1894, 1895, 1931, 1935, 1940, 1941, 1997, 2108, 2111, 2116, 2123, 2167, 2620, 3083, 3084, 3237, 4101, 4111, 4145, 4232, 4512
\@@_OnlyMainNiceMatrix_i:n 3562, 3569, 3572	\l_@@_color_tl 281, 4461, 4488, 4493
\@@_Vdots 990, 1055, 2967	\@@_colortbl_like: 996, 1071
\g_@@_Vdots_lines_tl 1083, 2280	\l_@@_colortbl_like_bool 414, 573, 1071, 1379
\@@_Vdotsfor: 1062, 3176	\c_@@_colortbl_loaded_bool 82, 86, 1021
\@@_Vdotsfor:nmmn 3180, 3191	\@@_columncolor 1180, 3344
\@@_W: 1364, 1433	\@@_columncolor_preamble 1000, 3550
\@@_actually_diagbox:nmmmmn 4323, 4502, 4743, 4759	\c_@@_columncolor_regex 203, 1382
\@@_actually_draw_Cdots: 2530, 2534	\l_@@_columns_width_dim 229, 558, 680, 1863, 1869, 4056, 4062
\@@_actually_draw_Ddots: 2656, 2660	\g_@@_com_or_env_str 251, 254
\@@_actually_draw_Iddots: 2707, 2711	\@@_computations_for_large_nodes: 4172, 4185, 4190
\@@_actually_draw_Ldots: 2488, 2492, 3167	\@@_computations_for_medium_nodes: 4092, 4161, 4171, 4182
\@@_actually_draw_Vdots: 2583, 2587, 3242	\@@_compute_a_corner:nmmmmn 3866, 3868, 3870, 3872, 3877
\@@_adapt_S_column: 164, 179, 1102	\@@_compute_corners: 2197, 3858
\@@_adjust_pos_of_blocks_seq: 2198, 2248	\@@_construct_preamble:n 1251, 1357
\@@_adjust_pos_of_blocks_seq_i:nmmn 2252, 2255	\@@_convert_to_str_seq:N 4876, 4888
\@@_adjust_size_box: 831, 857, 1520, 1974, 2018	\@@_create_col_nodes: 1754, 1782, 1801
\@@_after_array: 1353, 2104	\@@_create_extra_nodes: 1612, 4082
\@@_analyze_end:Nn 1750, 1795	
\l_@@_argspec_tl 2935, 2936, 2937, 2952, 2967, 2982, 3005, 3098, 3099, 3100, 3174, 3175, 3176, 3252, 3253, 3254	
\@@_array: 920, 1751, 1778	
\l_@@_auto_columns_width_bool 439, 557, 1862, 1866, 4050	
\l_@@_baseline_str 428, 429, 550, 551, 552, 553, 933, 1300, 1563, 1575, 1580, 1582, 1587, 1592, 1674, 1675, 1679, 1684, 1686, 1691	
\@@_begin_of_NiceMatrix:nn 2060, 2081	
\@@_begin_of_row: 757, 781, 1952	

\@@_create_large_nodes:	4090, 4166	\c_@@_enumitem_loaded_bool	25, 33, 307, 601, 606, 617, 622
\@@_create_medium_and_large_nodes: . .	4087, 4177	\@@_env:	223, 227, 790, 796, 891, 897, 945, 951, 957, 1147, 1148, 1154, 1155, 1162, 1163, 1173, 1810, 1813, 1815, 1831, 1837, 1840, 1849, 1855, 1858, 1880, 1886, 1889, 1906, 1912, 1918, 1931, 1935, 1941, 2221, 2349, 2417, 2456, 2467, 3130, 3148, 3205, 3223, 3274, 3276, 3295, 3298, 3888, 3907, 3925, 4114, 4116, 4124, 4235, 4244, 4262, 4531, 4538, 4542, 4556, 4561, 4573, 4578, 4579, 4582, 4599, 4630
\@@_create_medium_nodes:	4088, 4156	\g_@@_env_int	222, 223, 225, 1115, 1121, 1125, 1135, 1139, 1142, 1151, 1152, 1159, 1160, 1203, 1206, 1221, 1224, 2115, 2136, 2154, 2157, 2170, 2238, 3459, 3474, 4271
\@@_create_nodes: 4163, 4174, 4184, 4187, 4228		\@@_error:n	12, 310, 335, 459, 469, 627, 668, 679, 688, 693, 711, 718, 726, 732, 737, 748, 750, 1274, 1284, 1289, 1597, 1650, 1696, 3090, 3452, 4469, 4834, 4861, 4871
\@@_create_row_node:	936, 968, 1005	\@@_error:nn	13, 565, 2940, 2943, 2955, 2958, 2970, 2973, 2986, 2987, 2992, 2993, 3009, 3010, 3015, 3016, 3874
\@@_cut_on_hyphen:w	3303, 3325, 3326, 3359, 3360, 3389, 3420, 3431	\@@_error:nnn	14, 3279
\g_@@_ddots_int	2177, 2680, 2681	\@@_error_too_much_cols:	1414, 4895
\@@_def_env:nnn	2044, 2055, 2056, 2057, 2058, 2059	\@@_everycr:	962, 1026, 1029
\@@_define_L_C_R:	216, 1250	\@@_everycr_i:	962, 963
\c_@@_define_L_C_R_bool . . .	215, 1250, 4824	\l_@@_except_corners_clist	435, 533, 537, 3597, 3717, 3861
\@@_define_com:nnn	4707, 4715, 4716, 4717, 4718, 4719	\l_@@_exterior_arraycolsep_bool	430, 676, 1394, 1406
\g_@@_delta_x_one_dim	2179, 2683, 2693	\l_@@_extra_left_margin_dim	445, 525, 1254, 1985
\g_@@_delta_x_two_dim	2181, 2734, 2744	\l_@@_extra_right_margin_dim	446, 526, 1266, 2029, 2623
\g_@@_delta_y_one_dim	2180, 2685, 2693	\@@_extract_coords_values:	4253, 4260
\g_@@_delta_y_two_dim	2182, 2736, 2744	\@@_fatal:n	15, 245, 1106, 1759, 1763, 1765, 1798, 4900, 4903, 4906
\@@_diagbox:nn	1069, 4739	\@@_fatal:nn	16, 1440
\@@_dotfill:	4728	\l_@@_final_i_int	2186, 2296, 2301, 2304, 2329, 2337, 2341, 2350, 2358, 2438, 2468, 2508, 2605, 2672, 2723, 3121, 3149, 3217, 3227, 3229
\@@_dotfill_i:	4733, 4735	\l_@@_final_j_int	2187, 2297, 2302, 2309, 2314, 2320, 2330, 2338, 2342, 2351, 2359, 2439, 2469, 2505, 2545, 2674, 2725, 3142, 3152, 3154, 3196, 3225
\@@_dotfill_ii:	4732, 4735, 4736	\l_@@_final_open_bool	2189, 2303, 2307, 2310, 2317, 2323, 2327, 2343, 2503, 2543, 2552, 2563, 2590, 2603, 2611, 2632, 2670, 2721, 2857, 2872, 2903, 2904, 3119, 3143, 3155, 3194, 3218, 3230, 3271, 3977, 4016
\@@_dotfill_iii:	4736, 4737	\@@_find_extremities_of_line:nnnn	2291, 2478, 2520, 2572, 2652, 2703
\@@_double_int_eval:n	3248, 3262, 3263	\l_@@_first_col_int	117, 130, 288, 289, 492, 730, 757, 1293, 1386, 1804, 1824, 2161, 3316, 3367, 3399, 3428, 3561, 4101, 4111, 4145, 4193, 4232, 4689, 4695, 4701
\g_@@_dp_ante_last_row_dim	784, 1037	\l_@@_first_row_int	286, 287, 493, 734, 1075, 1308, 1594, 1665, 1693, 1704, 2159, 4094, 4108, 4135, 4192, 4230, 4535, 4553, 4687, 5004
\g_@@_dp_last_row_dim	784, 785, 1040, 1041, 1194, 1195, 1317		
\g_@@_dp_row_zero_dim	804, 805, 1031, 1032, 1310, 1668, 1707		
\@@_draw_Cdots:nnn	2516		
\@@_draw_Ddots:nnn	2648		
\@@_draw_Iddots:nnn	2699		
\@@_draw_Ldots:nnn	2474		
\@@_draw_Vdots:nnn	2568		
\@@_draw_blocks:	1613, 4471		
\@@_draw_dotted_lines:	2196, 2268		
\@@_draw_dotted_lines_i:	2271, 2275		
\l_@@_draw_first_bool .	284, 2997, 3020, 3031		
\@@_draw_hlines:	2209, 3810		
\@@_draw_line:	2514, 2566, 2646, 2697, 2748, 2750, 3301, 4011, 4041		
\@@_draw_line_ii:nn	3281, 3285		
\@@_draw_line_iii:nn	3288, 3292		
\@@_draw_non_standard_dotted_line: . .	2756, 2758		
\@@_draw_non_standard_dotted_line: . .	2761, 2764		
\@@_draw_standard_dotted_line: .	2755, 2784		
\@@_draw_standard_dotted_line_i: .	2849, 2853		
\@@_draw_vlines:	2210, 3693		
\g_@@_empty_cell_bool	273, 865, 874, 1987, 2033, 2950, 2965, 2980, 3003, 3026, 3037		
\l_@@_empty_corner_cells_seq	2203, 3631, 3637, 3644, 3750, 3756, 3763, 3860, 3933		
\@@_end_Cell:	196, 852, 1449, 1500, 1516, 2071, 3052, 3053, 3054, 3055, 3056, 3057, 3058, 3059, 3060, 3061, 3062, 3063		
\l_@@_end_of_row_tl	447, 448, 490, 1774, 1775, 4993		
\c_@@_endpgfortikzpicture_tl	39, 43, 2272, 3289, 3980		

<code>\c_@@_footnote_bool</code>	<code>\l_@@_large_nodes_bool</code> 442, 516, 4086, 4090
..... 1092, 1355, 4806, 4832, 4855, 4874	<code>\g_@@_last_col_found_bool</code> 296,
<code>\c_@@_footnotehyper_bool</code> . 4805, 4833, 4865	1080, 1280, 1348, 1893, 1922, 1995, 2107, 2164
<code>\@@_full_name_env:</code>	<code>\l_@@_last_col_int</code>
..... 252, 4916, 4923, 4931, 4939, 294, 295, 669, 704, 706, 719,
4943, 5009, 5030, 5049, 5056, 5075, 5084, 5216	733, 749, 1145, 1217, 1223, 1230, 1283,
<code>\@@_hdottedline:</code> 1058, 3962	1398, 2067, 2069, 2108, 2111, 2163, 2577,
<code>\@@_hdottedline:n</code> 3970, 3974	2618, 2942, 2957, 2993, 3016, 4481, 4486,
<code>\@@_hdottedline_i:</code> 3965, 3967	4487, 4495, 4506, 4512, 4526, 4556, 4561,
<code>\@@_hdottedline_i:n</code> 3979, 3983	4569, 4584, 4625, 4691, 4697, 4703, 4899, 4917
<code>\@@_hline:nn</code> 3700, 3815, 3833	<code>\l_@@_last_col_without_value_bool</code> ...
<code>\@@_hline_i:nn</code> 2207, 3703, 3706 293, 703, 2109, 4902
<code>\@@_hline_i_complete:nn</code> 2207, 3808	<code>\l_@@_last_empty_column_int</code>
<code>\@@_hline_ii:nnnn</code> ... 3726, 3737, 3770, 3809 3898, 3899, 3912, 3918, 3931
<code>\l_@@_hlines_bool</code> 434, 504, 509, 539, 969, 2209	<code>\l_@@_last_empty_row_int</code>
<code>\g_@@_ht_last_row_dim</code> 3880, 3881, 3894, 3915
..... 786, 1038, 1039, 1192, 1193, 1316	<code>\l_@@_last_row_int</code> 290, 291, 494, 774, 820,
<code>\g_@@_ht_row_one_dim</code> .. 812, 813, 1035, 1036	973, 1143, 1188, 1198, 1205, 1212, 1268,
<code>\g_@@_ht_row_zero_dim</code>	1272, 1275, 1292, 1314, 1776, 1777, 1960,
.... 806, 807, 1033, 1034, 1311, 1667, 1706	1961, 2004, 2005, 2130, 2483, 2525, 2972,
<code>\@@_i:</code> 4094, 4096,	2987, 3010, 3567, 3575, 4480, 4483, 4484,
4097, 4098, 4099, 4108, 4114, 4116, 4117,	4495, 4505, 4511, 4524, 4583, 4594, 4699, 5074
4118, 4119, 4124, 4125, 4126, 4127, 4135,	<code>\l_@@_last_row_without_value_bool</code> ...
4138, 4140, 4141, 4142, 4194, 4196, 4199, 292, 1200, 1270, 2128
4200, 4204, 4205, 4230, 4235, 4237, 4239,	<code>\l_@@_left_delim_dim</code>
4243, 4244, 4255, 4262, 4264, 4266, 4270, 4271 1236, 1240, 1245, 1742, 1983
<code>\g_@@_iddots_int</code> 2178, 2731, 2732	<code>\l_@@_left_delim_tl</code> 1094, 4005
<code>\l_@@_in_env_bool</code> 236, 334, 1106, 1107	<code>\l_@@_left_margin_dim</code>
<code>\c_@@_in_preamble_bool</code> . 21, 22, 23, 597, 613 443, 519, 1253, 1984, 3995, 4223
<code>\@@_info:n</code> 4868	<code>\l_@@_letter_for_dotted_lines_str</code> ...
<code>\l_@@_initial_i_int</code> 2184, 687, 695, 696, 1438
2294, 2369, 2372, 2397, 2405, 2409, 2418,	<code>\l_@@_light_syntax_bool</code>
2426, 2436, 2457, 2499, 2554, 2556, 2599, 427, 488, 1256, 1261, 2131
2664, 2715, 3120, 3121, 3131, 3199, 3209, 3211	<code>\@@_light_syntax_i</code> 1767, 1770
<code>\l_@@_initial_j_int</code>	<code>\@@_line</code> 2225, 3254
..... 2185, 2295, 2370, 2377,	<code>\@@_line_i:nn</code> 3261, 3268
2382, 2388, 2398, 2406, 2410, 2419, 2427,	<code>\@@_line_with_light_syntax:n</code> ... 1781, 1785
2437, 2458, 2496, 2538, 2613, 2615, 2620,	<code>\@@_line_with_light_syntax_i:n</code>
2666, 2717, 3124, 3134, 3136, 3195, 3196, 3207 1780, 1786, 1787
<code>\l_@@_initial_open_bool</code>	<code>\@@_math_toggle_token:</code>
.... 2188, 2371, 2375, 2378, 2385, 2391,	149, 854, 1954, 1971, 1999, 2015, 4783, 4787
2395, 2411, 2494, 2536, 2551, 2561, 2590,	<code>\g_@@_max_cell_width_dim</code>
2597, 2609, 2662, 2713, 2855, 2902, 3118, 862, 863, 1117, 1868, 4049, 4075
3125, 3137, 3193, 3200, 3212, 3270, 3976, 4015	<code>\l_@@_max_delimiter_width_bool</code>
<code>\@@_insert_tabularnotes:</code> 1617, 1620 450, 487, 1344
<code>\@@_instruction_of_type:nnn</code>	<code>\c_@@_max_l_dim</code> 2847, 2852
..... 901, 2945, 2960, 2975, 2997, 3020	<code>\l_@@_medium_nodes_bool</code> 441, 515, 4084, 4575
<code>\l_@@_inter_dots_dim</code>	<code>\@@_message_hdotsfor:</code> 4908, 4916, 4923, 4931
. 418, 419, 2193, 2860, 2867, 2878, 2886,	<code>\@@_msg_new:nn</code>
2893, 2898, 2910, 2918, 4006, 4009, 4037, 4039 17, 4837, 4846, 4913, 4920, 4928,
<code>\g_@@_internal_code_after_tl</code> 259, 1483,	4936, 4941, 4948, 4953, 4958, 4963, 4968,
1537, 2211, 2212, 3832, 3969, 4321, 4500, 4741	4973, 4979, 4985, 4990, 4996, 5001, 5008,
<code>\@@_intersect_our_row:nnnn</code> 3512	5010, 5016, 5021, 5028, 5035, 5042, 5047,
<code>\@@_intersect_our_row_p:nnnn</code> 3482	5053, 5060, 5066, 5072, 5080, 5082, 5088, 5326
<code>\@@_j:</code> 4101, 4103,	<code>\@@_msg_new:nnn</code> 18,
4104, 4105, 4106, 4111, 4114, 4116, 4119,	4807, 5093, 5103, 5121, 5163, 5213, 5263, 5312
4121, 4122, 4124, 4127, 4129, 4130, 4145,	<code>\@@_msg_redirect_name:nn</code> 19, 682
4148, 4150, 4151, 4152, 4207, 4209, 4212,	<code>\@@_multicolumn:nnn</code> 1063, 3041
4214, 4218, 4219, 4232, 4235, 4236, 4238,	<code>\g_@@_multicolumn_cells_seq</code>
4243, 4244, 4256, 4262, 4263, 4265, 4270, 4271	... 1073, 3071, 4119, 4127, 4249, 4540, 4558
<code>\l_@@_l_dim</code>	<code>\g_@@_multicolumn_sizes_seq</code> 1074, 3073, 4250
.... 2833, 2834, 2847, 2848, 2860, 2866,	
2877, 2885, 2893, 2898, 2910, 2911, 2918, 2919	

<code>\g_@@_name_env_str</code>	250, 255, 256, 1100, 1101, 1797, 2039, 2040, 2048, 2049, 2078, 2089, 2097, 2244, 4711, 4897
<code>\l_@@_name_str</code>	440, 567, 792, 795, 893, 896, 953, 956, 1201, 1210, 1213, 1219, 1228, 1231, 1814, 1815, 1839, 1840, 1857, 1858, 1888, 1889, 1914, 1917, 1937, 1940, 2118, 2122, 2139, 2143, 4240, 4243, 4267, 4270
<code>\g_@@_names_seq</code>	235, 564, 566, 5324
<code>\l_@@_nb_cols_int</code>	4678, 4683, 4686, 4690, 4696, 4702
<code>\l_@@_nb_rows_int</code>	4677, 4682, 4693
<code>@@_newcolumnmtype</code>	979, 1363, 1364
<code>@@_node_for_multicolumn:nn</code>	4251, 4258
<code>@@_node_for_the_cell:</code> ..	871, 877, 1981, 2030
<code>@@_node_position:</code> ..	1154, 1156, 1162, 1164
<code>\g_@@_not_empty_cell_bool</code>	264, 869, 875, 2086
<code>@@_not_in_exterior:nnnn</code>	3504
<code>@@_not_in_exterior_p:nnnn</code>	3476
<code>\l_@@_notes_above_space_dim</code>	436, 437
<code>\l_@@_notes_bottomrule_bool</code>	585, 722, 743, 1643
<code>\l_@@_notes_code_after_tl</code>	583, 1652
<code>\l_@@_notes_code_before_tl</code>	581, 1624
<code>@@_notes_label_in_list:n</code> ..	303, 322, 330, 593
<code>@@_notes_label_in_tabular:n</code> ..	302, 343, 590
<code>\l_@@_notes_para_bool</code> ..	579, 720, 741, 1628
<code>@@_notes_style:n</code>	301, 304, 322, 330, 346, 351, 587
<code>\l_@@_nullify_dots_bool</code>	438, 514, 2949, 2964, 2979, 3002, 3025
<code>\l_@@_number_of_notes_int</code> ..	300, 337, 347, 357
<code>@@_old_CT@arc@</code>	1108, 2246
<code>@@_old_arraycolsep_dim</code>	274
<code>@@_old_cdots</code>	1046, 2964
<code>@@_old_ddots</code>	1048, 3002
<code>@@_old_dotfill</code>	4727, 4730, 4738
<code>@@_old_dotfill:</code>	1067
<code>\l_@@_old_iRow_int</code>	260, 1008, 2288
<code>@@_old_ialign:</code>	935, 1042, 4473
<code>@@_old_iddots</code>	1049, 3025
<code>\l_@@_old_jCol_int</code>	261, 1011, 2289
<code>@@_old_ldots</code>	1045, 2949
<code>@@_old_multicolumn</code>	3040, 3047
<code>@@_old_pgful@check@rerun</code>	75, 79
<code>@@_old_vdots</code>	1047, 2979
<code>\l_@@_parallelize_diags_bool</code>	431, 432, 511, 2175, 2678, 2729
<code>@@_patch_preamble:n</code>	1376, 1418, 1457, 1486, 1539, 1551
<code>@@_patch_preamble_i:n</code>	1422, 1423, 1424, 1443
<code>@@_patch_preamble_ii:nn</code>	1425, 1426, 1427, 1454
<code>@@_patch_preamble_iii:n</code> ..	1428, 1459, 1467
<code>@@_patch_preamble_iii_i:n</code>	1462, 1464
<code>@@_patch_preamble_iv:nnn</code>	1429, 1430, 1431, 1489
<code>@@_patch_preamble_ix:n</code>	1544, 1554
<code>@@_patch_preamble_v:nnnn</code> ..	1432, 1433, 1505
<code>@@_patch_preamble_vi:n</code>	1434, 1527
<code>@@_patch_preamble_vii:n</code>	1439, 1533
<code>@@_patch_preamble_viii:n</code>	1452, 1503, 1525, 1531, 1541, 1557
<code>@@_pgf_rect_node:nnn</code>	390, 4577
<code>@@_pgf_rect_node:nnnnn</code>	365, 4234, 4261, 4530, 4572
<code>\c_@@_pgfortikzpicture_tl</code>	38, 42, 2270, 3287, 3978
<code>@@_picture_position:</code>	1148, 1156, 1164
<code>\l_@@_pos_of_block_tl</code> ...	282, 283, 4276, 4278, 4280, 4292, 4293, 4344, 4346, 4357, 4360, 4410, 4433, 4435, 4446, 4449, 4463, 4465, 4467, 4601, 4613, 4624, 4626, 4632, 4644
<code>\g_@@_pos_of_blocks_seq</code>	278, 1119, 2171, 2201, 2251, 2253, 3074, 3593, 3713, 3945, 4308, 4751
<code>\g_@@_pos_of_xdots_seq</code>	279, 1120, 2202, 2434, 3595, 3715
<code>@@_pre_array:</code>	1002, 1235
<code>\c_@@_preamble_first_col_tl</code>	1387, 1948
<code>\c_@@_preamble_last_col_tl</code>	1399, 1991
<code>\g_@@_preamble_tl</code>	1361, 1371, 1374, 1384, 1387, 1396, 1399, 1408, 1413, 1445, 1456, 1469, 1491, 1507, 1529, 1535, 1548, 1556, 1751, 1778
<code>@@_pred:n</code>	118, 148, 2069, 3632, 3645, 3751, 3764
<code>@@_provide_pgfsyspdfmark:</code> ..	204, 213, 1091
<code>@@_put_box_in_flow:</code>	1346, 1559, 1744
<code>@@_put_box_in_flow_bis:nn</code>	1345, 1711
<code>@@_put_box_in_flow_i:</code>	1565, 1567
<code>@@_qpoint:n</code>	226, 1570, 1572, 1584, 1600, 1659, 1661, 1677, 1688, 1699, 2496, 2499, 2505, 2508, 2538, 2545, 2554, 2556, 2599, 2605, 2613, 2615, 2664, 2666, 2672, 2674, 2715, 2717, 2723, 2725, 3295, 3298, 3315, 3319, 3332, 3334, 3351, 3353, 3366, 3370, 3390, 3396, 3398, 3402, 3425, 3427, 3436, 3438, 3655, 3657, 3659, 3774, 3776, 3778, 3986, 3990, 3997, 4033, 4036, 4038, 4140, 4150, 4520, 4522, 4524, 4526, 4549, 4569, 4597, 4764, 4766, 4769, 4771
<code>\l_@@_radius_dim</code>	422, 423, 1536, 2192, 2512, 2513, 2927, 3964, 3988, 4034, 4035
<code>\l_@@_real_left_delim_dim</code> ..	1713, 1728, 1743
<code>\l_@@_real_right_delim_dim</code> ..	1714, 1740, 1746
<code>@@_rectanglecolor</code>	1177, 3413
<code>@@_renew_NC@rewrite@S:</code>	185, 187, 1079
<code>@@_renew_dots:</code>	986, 1072
<code>\l_@@_renew_dots_bool</code>	512, 1072, 4826
<code>@@_renew_matrix:</code>	672, 4657, 4828
<code>\l_@@_respect_blocks_bool</code>	3450, 3458
<code>@@_restore_iRow_jCol:</code>	2245, 2286
<code>@@_revtex_array:</code>	912, 923
<code>\c_@@_revtex_bool</code>	46, 48, 51, 922
<code>\l_@@_right_delim_dim</code>	1237, 1241, 1247, 1745, 2027
<code>\l_@@_right_delim_tl</code>	1095, 4008
<code>\l_@@_right_margin_dim</code>	444, 521, 1265, 2028, 2622, 4002, 4226
<code>@@_rotate:</code>	1065, 3247
<code>\g_@@_rotate_bool</code>	241, 829, 856, 1519, 1973, 2017, 3247, 4343, 4356, 4367, 4372, 4432, 4445, 4516
<code>@@_rotate_cell_box:</code>	817, 856, 1519, 1973, 2017, 4516

<code>\g_@@_row_of_col_done_bool</code>	<code>\c_@@_tikz_loaded_bool</code>
263, 966, 1099, 1823	26, 37, 1168, 2213, 4017
<code>\g_@@_row_total_int</code>	<code>\l_@@_tikz_tl</code>
1076, 1291,	4459, 4529
1595, 1694, 2130, 2137, 2144, 2160, 3162,	<code>\@@_true_c:</code>
4094, 4108, 4135, 4230, 4511, 4535, 4553, 5005	195, 1434
<code>\@@_rowcolor</code> 1178, 3308, 3465, 3466, 3487, 3492	<code>\l_@@_type_of_col_tl</code> ..
<code>\@@_rowcolor_tabular</code>	707, 708, 2079, 2081
999, 3541	<code>\c_@@_types_of_matrix_seq</code>
<code>\@@_rowcolors</code>	4890, 4897
1179, 3454	<code>\@@_update_for_first_and_last_row:</code> ..
<code>\@@_rowcolors_ii:nnnn</code>	800, 864, 1190, 1975, 2019
3460, 3471	<code>\@@_use_arraybox_with_notes:</code> ...
<code>\@@_rowcolors_ii:nnnn</code>	1305, 1672
3484, 3499	<code>\@@_use_arraybox_with_notes_b:</code> .
<code>\g_@@_rows_seq</code> .	1302, 1656
1773, 1775, 1777, 1779, 1781	<code>\@@_use_arraybox_with_notes_c:</code>
<code>\l_@@_rules_color_tl</code> ..	1303, 1333, 1608, 1670, 1709
262, 473, 1131, 1132	<code>\@@_vdottedline:n</code>
<code>\@@_set_CT@arc:</code>	1538, 4013
151, 1132	<code>\@@_vdottedline_i:n</code>
<code>\@@_set_CT@arc@i:</code>	4020, 4025, 4029
152, 153	<code>\@@_vline:nn</code>
<code>\@@_set_CT@arc@ii:</code>	1484, 3577, 3698
152, 155	<code>\@@_vline_i:nn</code>
<code>\@@_set_final_coords:</code>	2206, 3582, 3586
2447, 2472	<code>\@@_vline_i_complete:nn</code>
<code>\@@_set_final_coords_from_anchor:n</code> ..	2206, 3691
2463, 2511, 2549, 2593, 2608, 2677, 2728	<code>\@@_vline_ii:nnnn</code> ...
<code>\@@_set_initial_coords:</code>	3607, 3618, 3651, 3692
2442, 2461	<code>\l_@@_vlines_bool</code>
<code>\@@_set_initial_coords_from_anchor:n</code> .	433,
2452, 2502, 2542, 2592, 2602, 2669, 2720	505, 508, 538, 1369, 1393, 1405, 1546, 2210
<code>\@@_set_seq_of_str_from_clist:Nn</code> 4885, 4890	<code>\@@_w:</code>
<code>\@@_set_size:n</code>	1363, 1432
4675, 4684	<code>\g_@@_width_first_col_dim</code>
<code>\c_@@_siunitx_loaded_bool</code> 157, 161, 166, 184	276, 1098, 1296, 1818, 1976, 1977
<code>\l_@@_small_bool</code>	<code>\g_@@_width_last_col_dim</code>
670,	275, 1097, 1350, 1927, 2020, 2021
709, 715, 735, 763, 1014, 1955, 2000, 2190	<code>\l_@@_x_final_dim</code>
<code>\@@_standard_cline</code>	269,
114, 1051	2449, 2506, 2507, 2546, 2547, 2595, 2617,
<code>\@@_standard_cline:w</code>	2625, 2629, 2633, 2635, 2640, 2642, 2675,
114, 115	2684, 2692, 2726, 2735, 2743, 2781, 2795,
<code>\l_@@_standard_cline_bool</code> ..	2804, 2840, 2892, 2908, 3299, 3998, 4009, 4035
415, 480, 1050	<code>\l_@@_x_initial_dim</code>
<code>\c_@@_standard_tl</code> 425, 426, 2754, 4010, 4040	267, 2444, 2497, 2498, 2539, 2540,
<code>\g_@@_static_num_of_col_int</code>	2595, 2616, 2617, 2624, 2629, 2633, 2635,
280, 1288, 1377, 4932, 4944	2637, 2640, 2642, 2667, 2684, 2692, 2718,
<code>\l_@@_stop_loop_bool</code>	2735, 2743, 2772, 2794, 2804, 2840, 2892,
2298, 2299,	2906, 2908, 2926, 2928, 3296, 3991, 4006, 4034
2331, 2344, 2353, 2366, 2367, 2399, 2412, 2421	<code>\l_@@_xdots_color_tl</code> 449, 462, 2487, 2529,
<code>\@@_succ:n</code>	2581, 2582, 2655, 2706, 2762, 3166, 3241, 3258
143, 147, 945, 951, 1484,	<code>\l_@@_xdots_down_tl</code> ...
1572, 1906, 1912, 1917, 1918, 1931, 1935,	466, 2778, 2788, 2823
1940, 1941, 2165, 2505, 2545, 2556, 2605,	<code>\l_@@_xdots_line_style_tl</code>
2615, 2672, 2674, 2717, 2723, 3319, 3332,	424, 426, 458, 2754, 2762, 4010, 4040
3353, 3370, 3396, 3402, 3436, 3438, 3508,	<code>\l_@@_xdots_shorten_dim</code>
3628, 3659, 3697, 3747, 3778, 3814, 3833,	420,
3950, 3952, 3954, 3956, 3997, 4038, 4200,	421, 464, 2194, 2769, 2770, 2866, 2877, 2885
4204, 4214, 4218, 4524, 4526, 4569, 4769, 4771	<code>\l_@@_xdots_up_tl</code>
<code>\l_@@_suffix_tl</code>	467, 2774, 2787, 2813
4162, 4173,	<code>\l_@@_y_final_dim</code>
4183, 4186, 4235, 4243, 4244, 4262, 4270, 4271	270,
<code>\c_@@_table_collect_begin_tl</code> .	2450, 2509, 2513, 2558, 2562, 2564, 2606,
174, 176, 194	2673, 2686, 2689, 2724, 2737, 2740, 2781,
<code>\c_@@_table_print_tl</code>	2795, 2803, 2842, 2897, 2916, 3300, 3989, 4039
177, 178, 196	<code>\l_@@_y_initial_dim</code>
<code>\l_@@_tabular_width_dim</code>	268, 2445, 2500, 2512, 2557, 2558, 2562,
239, 928, 930, 1410, 2098	2564, 2600, 2665, 2686, 2691, 2716, 2737,
<code>\l_@@_tabularnote_tl</code> 299, 724, 745, 1616, 1625	2742, 2772, 2794, 2803, 2842, 2897, 2914,
<code>\g_@@_tabularnotes_seq</code>	2916, 2926, 2929, 3297, 3987, 3988, 3989, 4037
298, 338, 1631, 1637, 1653	<code>\l_@@_y_initial_dim</code>
<code>\@@_test_if_cell_in_a_block:nn</code>	268, 2445, 2500, 2512, 2557, 2558, 2562,
3884, 3902, 3920, 3940	2564, 2600, 2665, 2686, 2691, 2716, 2737,
<code>\@@_test_if_cell_in_block:nnnnnnn</code> ...	2742, 2772, 2794, 2803, 2842, 2897, 2914,
3946, 3948	2916, 2926, 2929, 3297, 3987, 3988, 3989, 4037
<code>\@@_test_if_hline_in_block:nnnn</code>	<code>\l_@@_y_initial_dim</code>
3714, 3716, 3836	268, 2445, 2500, 2512, 2557, 2558, 2562,
<code>\@@_test_if_math_mode:</code>	2564, 2600, 2665, 2686, 2691, 2716, 2737,
242, 1105, 2050	2742, 2772, 2794, 2803, 2842, 2897, 2914,
<code>\@@_test_if_vline_in_block:nnnn</code>	2916, 2926, 2929, 3297, 3987, 3988, 3989, 4037
3594, 3596, 3847	<code>\l_@@_y_initial_dim</code>
<code>\@@_test_in_corner_h:</code>	268, 2445, 2500, 2512, 2557, 2558, 2562,
3717, 3745	2564, 2600, 2665, 2686, 2691, 2716, 2737,
<code>\@@_test_in_corner_v:</code>	2742, 2772, 2794, 2803, 2842, 2897, 2914,
3598, 3626	2916, 2926, 2929, 3297, 3987, 3988, 3989, 4037
<code>\l_@@_the_array_box</code> ..	<code>\l_@@_y_initial_dim</code>
1249, 1252, 1610, 1611	268, 2445, 2500, 2512, 2557, 2558, 2562,

<code>_</code>	4911, 4916, 4923, 4931, 4932, 4943, 4944, 5004, 5005, 5009, 5018, 5049, 5055, 5068, 5075, 5076, 5084
A	
<code>\aboverulesep</code>	1647
<code>\addtocounter</code>	355
<code>\alph</code>	301
<code>\arraycolsep</code>	520, 522, 524, 927, 1017, 1240, 1241, 1332, 1336, 3994, 4001
<code>\arrayrulecolor</code>	89
<code>\arrayrulewidth</code> 122, 127, 139, 475, 791, 944, 946, 952, 974, 1327, 1339, 1372, 1477, 1549, 1664, 1703, 1830, 1832, 1838, 1848, 1850, 1856, 1879, 1881, 1887, 1905, 1907, 1913, 3317, 3318, 3320, 3333, 3335, 3352, 3354, 3368, 3369, 3371, 3395, 3397, 3400, 3401, 3403, 3426, 3429, 3430, 3437, 3439, 3667, 3668, 3670, 3681, 3687, 3787, 3798, 3804, 3829, 4075
<code>\arraystretch</code>	1016, 4340, 4353, 4429, 4442
<code>\AtBeginDocument</code> 23, 27, 67, 83, 158, 182, 305, 599, 615, 2266, 2933, 3096, 3172, 3250, 3283, 3972
<code>\AutoNiceMatrix</code>	4720
<code>\AutoNiceMatrixWithDelims</code>	4680, 4712, 4724
B	
<code>\baselineskip</code>	92, 99
<code>\bggroup</code>	1093
<code>\Block</code>	1064, 5096
<code>\BNiceMatrix</code>	4672
<code>\bNiceMatrix</code>	4669
bool commands:	
<code>\bool_do_until:Nn</code>	2299, 2367
<code>\bool_gset_false:N</code>	829, 874, 875, 1080, 1099, 1987, 2033, 3845, 3856, 4372
<code>\bool_gset_true:N</code>	1823, 1995, 2086, 2950, 2965, 2980, 3003, 3026, 3037, 3247, 3592, 3712
<code>\bool_if:NTF</code>	150, 166, 601, 606, 617, 622, 760, 763, 856, 940, 966, 969, 1004, 1014, 1071, 1072, 1092, 1106, 1116, 1133, 1168, 1182, 1184, 1250, 1270, 1286, 1348, 1355, 1379, 1519, 1546, 1643, 1809, 1826, 1844, 1862, 1875, 1901, 1922, 1928, 1955, 1973, 2000, 2017, 2107, 2109, 2128, 2131, 2150, 2175, 2190, 2209, 2210, 2213, 2561, 2563, 2678, 2729, 2949, 2964, 2979, 3002, 3025, 3893, 3911, 3929, 4058, 4068, 4090, 4343, 4356, 4367, 4432, 4445, 4516, 4575, 4731, 4855, 4865, 4902
<code>\bool_if:nTF</code> 184, 307, 334, 903, 1280, 3272, 3514, 4084
<code>\bool_lazy_all:nTF</code> 1389, 1401, 2199, 3838, 3849
<code>\bool_lazy_and:nnTF</code>	1865, 1956, 2162, 2550, 2786, 3391, 3421, 3457, 3661, 3780
<code>\bool_lazy_or:nnTF</code> 455, 868, 1593, 1614, 1692, 2003, 2590, 2846, 3506, 3885, 3903, 3921, 4309, 4510
<code>\bool_lazy_or_p:nn</code>	1959
<code>\bool_not_p:n</code>	1392, 1393, 1394, 1404, 1405, 1406, 1867, 2164
<code>\bool_set:Nn</code>	2594
<code>\c_false_bool</code>	2945, 2960, 2975
<code>\g_tmpa_bool</code>	3592, 3599, 3633, 3641, 3646, 3712, 3718, 3752, 3760, 3765, 3845, 3856
<code>\l_tmpb_bool</code>	3890, 3904, 3922, 3944, 3957
box commands:	
<code>\box_clear_new:N</code>	1006, 1249
<code>\box_dp:N</code>	785, 805, 842, 861, 1032, 1041, 1195, 1562, 1722, 1735, 4402
<code>\box_gclear_new:N</code>	4331
<code>\box_grotate:Nn</code>	4369
<code>\box_ht:N</code>	786, 807, 813, 825, 848, 859, 1034, 1036, 1039, 1193, 1561, 1722, 1735, 4393
<code>\box_move_up:nn</code>	58, 60, 62, 1605, 1670, 1709
<code>\box_rotate:Nn</code>	819
<code>\box_set_dp:Nn</code>	841, 860, 1562
<code>\box_set_ht:Nn</code>	847, 858, 1561
<code>\box_set_wd:Nn</code>	835
<code>\box_use:N</code>	358, 826, 1497, 1500
<code>\box_use_drop:N</code>	866, 872, 890, 1521, 1564, 1605, 1606, 1611, 1982, 4412, 4620, 4651
<code>\box_wd:N</code> 359, 836, 863, 870, 1245, 1247, 1610, 1729, 1741, 1977, 1980, 2021, 2025, 4381, 4738
<code>\l_tmpa_box</code> 341, 358, 359, 1244, 1245, 1246, 1247, 1320, 1561, 1562, 1564, 1605, 1606, 1722, 1735
<code>\l_tmpb_box</code>	1715, 1729, 1730, 1741
C	
<code>\c</code>	203, 1383
<code>\Cdots</code>	1054, 2955, 2958
<code>\cdots</code>	989, 1046
<code>\cellcolor</code>	998, 1176, 3539
<code>\chessboardcolors</code>	1181
<code>\cline</code>	142, 1051, 1052
clist commands:	
<code>\clist_if_empty:NTF</code>	3597, 3717
<code>\clist_map_inline:Nn</code>	3861
<code>\clist_map_inline:nn</code>	2074, 3321, 3355, 3387
<code>\clist_new:N</code>	435
<code>\clist_set:Nn</code>	537
<code>\CodeAfter</code>	754, 1068, 1767, 1770, 2226
<code>\color</code>	93, 100, 154, 156, 2481, 2484, 2487, 2523, 2526, 2529, 2575, 2578, 2582, 2655, 2706, 3160, 3163, 3166, 3235, 3238, 3241, 3258, 3314, 3350, 3386, 3419
<code>\colorlet</code>	248, 249, 770, 777, 1965, 2008
<code>\columncolor</code>	1000, 1180, 2233, 3555
<code>\cr</code>	126, 144, 1946
<code>\crrcr</code>	1803
cs commands:	
<code>\cs_generate_variant:Nn</code>	146, 4080, 4081
<code>\cs_gset:Npn</code> 93, 100, 2115, 2122, 2136, 2143, 4073
<code>\cs_gset_eq:NN</code>	179, 213, 1025, 1108, 2246
<code>\cs_if_exist:NTF</code>	1007, 1010, 1109, 1112, 1203, 1210, 1221, 1228, 2288, 2289, 2334, 2347, 2402, 2415, 3128, 3146, 3203, 3221, 4060, 4113, 4537, 4555
<code>\cs_if_exist_p:N</code>	456, 3459, 3887, 3906, 3924
<code>\cs_if_free:NTF</code> 209, 2476, 2518, 2570, 2650, 2701
<code>\cs_if_free_p:N</code>	3274, 3276

\hspace 3038
 \hss 1433

I

\ialign 935, 1019, 1042, 4473
 \iddots 1057, 3009, 3010, 3015, 3016
 \iddots 53, 992, 1049
 if commands:
 \if_mode_math: 244
 \ifnum 104, 3817, 3834
 \ifstandalone 1112
 int commands:
 \int_case:nnTF 2984, 2990, 3007, 3013
 \int_compare:nNnTF 117,
 118, 134, 756, 757, 767, 774, 810, 820, 971,
 973, 1143, 1145, 1188, 1198, 1217, 1268,
 1272, 1283, 1288, 1292, 1293, 1626, 1665,
 1704, 1776, 1804, 2001, 2309, 2316, 2320,
 2322, 2377, 2384, 2388, 2390, 2483, 2525,
 2577, 2618, 2620, 3069, 3083, 3162, 3237,
 3330, 3364, 3432, 3434, 3501, 3552, 3566,
 3567, 3574, 3575, 3579, 3950, 3952, 3954,
 3956, 4336, 4374, 4386, 4594, 4623, 4625,
 4687, 4689, 4691, 4695, 4697, 4699, 4701, 4703
 \int_compare_p:n
 3392, 3393, 3422, 3423, 3516, 3518
 \int_do_until:nNnn 3479
 \int_gadd:Nn 3082
 \int_gincr:N .. 755, 783, 1115, 1451, 1502,
 1524, 1530, 1899, 1996, 2680, 2731, 4055, 4318
 \int_if_even:nTF 3529
 \int_incr:N 337, 1461
 \int_step_inline:nn 3525, 3527
 \int_step_inline:nnnn 3882, 3900, 3915, 3918
 \c_zero_int 3507
 iow commands:
 \iow_now:Nn 70,
 206, 2152, 2153, 2155, 2173, 2234, 2235, 2241
 \iow_shipout:Nn 2112, 2113, 2120,
 2126, 2133, 2134, 2141, 2147, 4070, 4071, 4077
 \item 1631, 1637

K

\kern 63
 keys commands:
 \keys_define:nn 451,
 471, 478, 531, 577, 629, 665, 699, 713,
 728, 739, 3029, 3448, 4044, 4274, 4457, 4822
 \l_keys_key_tl 4809, 4976,
 4982, 5090, 5095, 5105, 5123, 5165, 5215, 5265
 \keys_set:nn 486, 691, 698, 1128,
 1129, 2080, 2090, 2099, 2486, 2528, 2580,
 2654, 2705, 3165, 3240, 3257, 3456, 4057, 4479
 \keys_set_known:nn 2996, 3019, 4294
 \l_keys_value_tl 5033, 5040, 5314

L

\Ldots 1053, 2940, 2943
 \ldots 988, 1045
 \leaders 122, 139
 \left 1323, 1718, 1733
 legacy commands:
 \legacy_if:nTF 561
 \line 2225, 5012

M

\makebox 1521
 \mathinner 55
 mode commands:
 \mode_leave_vertical: 1104, 1496
 msg commands:
 \msg_error:nn 12
 \msg_error:nnn 13
 \msg_error:nnnn 14, 4513
 \msg_fatal:nn 15
 \msg_fatal:nnn 16
 \msg_info:nn 4858
 \msg_new:nnn 17
 \msg_new:nnnn 18
 \msg_redirect_name:nnn 20
 \multicolumn 1063, 3040, 3044, 3092, 3113
 \multispan 118, 119, 135, 136
 \myfiledate 6
 \myfileversion 7

N

\newcolumnntype 218, 219, 220
 \newcounter 297
 \NewDocumentCommand
 .. 309, 332, 697, 2937, 2952, 2967, 2982,
 3005, 3100, 3176, 3254, 3308, 3344, 3380,
 3413, 3454, 3523, 3536, 3541, 3550, 4680, 4720
 \NewDocumentEnvironment 1089,
 1748, 1757, 2036, 2046, 2076, 2087, 2095, 4053
 \NewExpandableDocumentCommand 224, 4283
 \newlist 313, 324
 \NiceArray 2092, 2101
 \NiceArrayWithDelims 2041, 2051
 nicematrix commands:
 \g_nicematrix_code_after_tl
 258, 570, 1772, 2227, 2228, 4793, 4801
 \g_nicematrix_code_before_tl
 ... 1087, 2230, 2239, 3538, 3543, 3554, 4490
 \NiceMatrixLastEnv 224
 \NiceMatrixOptions 697, 5124, 5319
 \NiceMatrixoptions 5037
 \noalign .. 92, 99, 104, 127, 962, 1025, 3817, 3964
 \nobreak 327
 \normalbaselines 1013
 \NotEmpty 1070
 \nulldelimiterspace 1729, 1741
 \numexpr 147, 148

O

\omit 117, 1806, 1822, 1898, 4790
 \OnlyMainNiceMatrix 1066, 3558

P

\par 1625, 1633
 peek commands:
 \peek_meaning:NTF 152, 339, 982
 \peek_meaning_ignore_spaces:NTF 1750, 3820
 \peek_meaning_remove_ignore_spaces:NTF 142
 \peek_remove_spaces:n 3067
 \pgfextracty 4597
 \pgfgetlastxy 401
 \pgfpathcircle 2925
 \pgfpathlineto 3678, 3684, 3795, 3801, 4773
 \pgfpathmoveto 3677, 3683, 3794, 3800, 4768

<code>\pgfpathrectanglecorners</code>	3336, 3372, 3404, 3440, 3671, 3788
<code>\pgfpointadd</code>	399
<code>\pgfpointanchor</code>	227, 2454, 2465, 4116, 4124, 4542, 4560, 4579, 4581, 4598, 4629
<code>\pgfpointdiff</code>	400, 1156, 1164
<code>\pgfpointlineattime</code>	2793
<code>\pgfpointorigin</code>	1813, 1936
<code>\pgfpointscale</code>	399
<code>\pgfpointshapeborder</code>	3295, 3298
<code>\pgfrememberpicturepositiononpagetrue</code> ...	788, 881, 950, 1812, 1836, 1854, 1885, 1911, 1934, 2277, 2752, 2830, 3294, 3653, 3772, 3985, 4032, 4159, 4169, 4180, 4518, 4763
<code>\pgfscope</code>	2790, 4780
<code>\pgfset</code>	368, 393, 882, 4529, 4609, 4640, 4779
<code>\pgfsetbaseline</code>	880
<code>\pgfsetlinewidth</code>	3687, 3804
<code>\pgfsetrectcap</code>	3688, 3805
<code>\pgfsetroundcap</code>	4776
<code>\pgfsyspdfmark</code>	209, 210
<code>\pgftransformrotate</code>	2797
<code>\pgftransformshift</code>	374, 399, 2791, 4608, 4627, 4781, 4785
<code>\pgfusepath</code>	2817, 2827
<code>\pgfusepathqfill</code>	2931, 3340, 3376, 3409, 3443, 3674, 3791
<code>\pgfusepathqstroke</code>	3689, 3806, 4777
<code>\phantom</code>	2949, 2964, 2979, 3002, 3025
<code>\pNiceMatrix</code>	4660
prg commands:	
<code>\prg_do_nothing:</code>	170, 179, 185, 213, 468, 1025, 2226
<code>\prg_new_conditional:Nnn</code>	3504, 3512
<code>\prg_replicate:nn</code>	347, 1894, 1895, 3113, 3679, 3796, 4690, 4693, 4696, 4702
<code>\prg_return_false:</code>	3509, 3521
<code>\prg_return_true:</code>	3510, 3520
<code>\ProcessKeysOptions</code>	4836
<code>\ProvideDocumentCommand</code>	53
<code>\ProvidesExplPackage</code>	4
Q	
<code>\quad</code>	328
quark commands:	
<code>\q_stop</code>	114, 115, 131, 132, 153, 155, 1132, 1376, 1435, 1767, 1770, 3248, 3262, 3263, 3303, 3325, 3326, 3359, 3360, 3389, 3420, 3431, 4253, 4260, 4285, 4288, 4675, 4684
R	
<code>\rectanglecolor</code>	1177, 2232, 3545, 4492
<code>\refstepcounter</code>	356
regex commands:	
<code>\regex_const:Nn</code>	203
<code>\regex_replace_all:NnN</code>	1381
<code>\relax</code>	147, 148
<code>\renewcommand</code>	189
<code>\RenewDocumentEnvironment</code>	4659, 4662, 4665, 4668, 4671
<code>\RequirePackage</code>	1, 3, 9, 10, 11
<code>\right</code>	1341, 1725, 1737
<code>\rotate</code>	1065
<code>\rowcolor</code>	999, 1178
<code>\rowcolors</code>	1179
S	
<code>\savenotes</code>	1092
<code>\scriptstyle</code>	763, 1955, 2000, 2774, 2778, 2813, 2823
seq commands:	
<code>\seq_clear:N</code>	4878
<code>\seq_clear_new:N</code>	2154, 3860
<code>\seq_count:N</code>	1777
<code>\seq_gclear:N</code> ..	1118, 1119, 1120, 1653, 2250
<code>\seq_gclear_new:N</code> ...	1073, 1074, 1773, 1789
<code>\seq_gpop_left:Nn</code>	1779, 1791
<code>\seq_gput_left:Nn</code>	566, 3071, 3073, 4308
<code>\seq_gput_right:Nn</code>	338, 2263, 2434, 3074, 4407, 4419, 4751
<code>\seq_gset_eq:Nn</code>	2253
<code>\seq_gset_from_clist:Nn</code>	2157, 2169
<code>\seq_gset_split:Nnn</code>	1775, 1790
<code>\seq_if_empty:NTF</code>	1613
<code>\seq_if_empty_p:N</code>	2201, 2202, 2203
<code>\seq_if_exist:NTF</code>	1135
<code>\seq_if_in:NnTF</code> ..	564, 3630, 3636, 3643, 3749, 3755, 3762, 4119, 4127, 4540, 4558, 4897
<code>\seq_item:Nn</code> ..	1139, 1142, 1151, 1152, 1159, 1160
<code>\seq_map_function:Nn</code>	1781
<code>\seq_map_inline:Nn</code> ..	1631, 1637, 1793, 2251, 3484, 3593, 3595, 3713, 3715, 3945, 4474, 4879
<code>\seq_mapthread_function:NNN</code>	4248
<code>\seq_new:N</code>	235, 277, 278, 279, 298
<code>\seq_put_left:Nn</code>	4881
<code>\seq_put_right:Nn</code>	3932
<code>\seq_set_eq:Nn</code>	3473, 4883
<code>\seq_set_filter:NNn</code>	3475, 3481
<code>\seq_set_from_clist:Nn</code>	4887
<code>\seq_use:Nnnn</code>	2171, 5324
<code>\g_tmpa_seq</code>	2250, 2253, 2263
<code>\l_tmpa_seq</code>	3475, 3481, 4878, 4881, 4883
<code>\l_tmpb_seq</code>	3473, 3475, 3481, 3484
<code>\setlist</code>	314, 325, 602, 607, 618, 623
skip commands:	
<code>\skip_gadd:Nn</code>	1870
<code>\skip_gset:Nn</code>	1861
<code>\skip_gset_eq:Nn</code>	1868, 1869
<code>\skip_horizontal:N</code>	123, 140, 1253, 1254, 1265, 1266, 1295, 1296, 1331, 1332, 1335, 1336, 1350, 1351, 1372, 1536, 1549, 1742, 1743, 1745, 1746, 1817, 1818, 1830, 1832, 1848, 1850, 1872, 1879, 1881, 1900, 1905, 1907, 1926, 1927, 1983, 1984, 1985, 1988, 2022, 2027, 2028, 2029
<code>\skip_horizontal:n</code>	359, 1473
<code>\skip_vertical:N</code>	127, 944, 946, 1326, 1327, 1338, 1339, 1622, 1647, 3964
<code>\skip_vertical:n</code>	825, 3827
<code>\g_tmpa_skip</code> ..	1861, 1868, 1869, 1870, 1872, 1900
<code>\c_zero_skip</code>	1030
<code>\space</code>	255, 256
<code>\stepcounter</code>	345, 350
str commands:	
<code>\c_backslash_str</code>	255
<code>\c_colon_str</code>	696
<code>\str_case:nn</code> ...	3050, 4601, 4613, 4632, 4644
<code>\str_case:nnTF</code>	1300, 1420, 1587, 3863

<code>\str_count:N</code>	1582, 1686
<code>\str_gclear:N</code>	2244
<code>\str_gset:Nn</code>	
...	1101, 2040, 2049, 2078, 2089, 2097, 4711
<code>\str_if_empty:NTF</code>	792, 893, 953,
	1100, 1201, 1219, 1814, 1839, 1857, 1888,
	1914, 1937, 2039, 2048, 2118, 2139, 4240, 4267
<code>\str_if_eq:nnTF</code>	78, 254,
	933, 1438, 1466, 1543, 1563, 1674, 1797, 4798
<code>\str_if_eq_p:nn</code>	457
<code>\str_if_in:NnTF</code>	1575, 1679
<code>\str_new:N</code>	250, 428, 440, 695
<code>\str_range:Nnn</code>	1579, 1683
<code>\str_set:Nn</code>	563, 687
<code>\str_set_eq:NN</code>	567, 696
<code>\l_tmpa_str</code>	563, 564, 566, 567
<code>\strut</code>	1631, 1637

T

<code>\tabcolsep</code>	926, 1331, 1335
<code>\tabskip</code>	1030
<code>\tabularnote</code>	309, 332, 339, 5055, 5068
<code>\tabularnotes</code>	1636
TeX and L ^A T _E X 2 _ε commands:	
<code>\@BTnormal</code>	1005
<code>\@acol</code>	916
<code>\@acoll</code>	914
<code>\@acolr</code>	915
<code>\@array@array</code>	918
<code>\@arrayacol</code>	914, 915, 916
<code>\@arstrutbox</code>	785, 786,
	825, 1032, 1034, 1036, 1039, 1041, 1497, 1500
<code>\@currenvir</code>	4798
<code>\@gobblethree</code>	210
<code>\@halignto</code>	917, 929, 930
<code>\@height</code>	107, 122, 139
<code>\@ifclassloaded</code>	47, 50, 4857, 4867
<code>\@ifnextchar</code>	1078
<code>\@ifpackageloaded</code>	
	29, 32, 35, 69, 85, 160, 4860, 4870
<code>\@mainaux</code>	70, 206, 2112, 2113, 2120,
	2126, 2133, 2134, 2141, 2147, 2152, 2153,
	2155, 2173, 2234, 2235, 2241, 4070, 4071, 4077
<code>\@tabarray</code>	931
<code>\@tempswafalse</code>	1367
<code>\@tempswattrue</code>	1366
<code>\@temptokena</code> ..	169, 172, 191, 193, 1365, 1376
<code>\@whiles</code>	1367
<code>\@width</code>	107
<code>\@xhline</code>	110
<code>\bBigg@</code>	1244, 1246
<code>\c@MaxMatrixCols</code>	2068, 4925
<code>\c@tabularnote</code>	1615, 1626, 1654
<code>\col@sep</code>	926, 927, 1295, 1351, 1817,
	1870, 1926, 1988, 2022, 2498, 2507, 2540, 2547
<code>\CT@arc</code>	89, 90
<code>\CT@arc@</code>	88, 93, 108, 121, 138, 154,
	156, 1108, 1648, 2246, 3686, 3803, 4031, 4775
<code>\CT@drs</code>	96, 97
<code>\CT@drsc@</code> ...	95, 100, 3663, 3666, 3782, 3785
<code>\CT@everycr</code>	1023
<code>\CT@row@color</code>	1025
<code>\if@temp</code>	1367
<code>\NC@</code>	981

<code>\NC@find</code>	170, 198
<code>\NC@list</code>	1367
<code>\NC@rewrite@S</code>	171, 189
<code>\new@ifnextchar</code>	1078
<code>\newcol@</code>	983, 984
<code>\nicematrix@redefine@check@rerun</code> ..	70, 73
<code>\pgf@relevantforpicturesizefalse</code>	
	2278, 2753, 2831, 2922, 3313, 3349, 3385,
	3418, 3654, 3773, 4160, 4170, 4181, 4519, 4762
<code>\pgfsys@getposition</code>	1148, 1154, 1162
<code>\pgfsys@markposition</code>	
	945, 1147, 1810, 1831, 1849, 1880, 1906, 1930
<code>\pgfutil@check@rerun</code>	75, 76
<code>\reserved@a</code>	109
<code>\set@color</code>	4336
<code>\tikz@library@external@loaded</code>	1109
tex commands:	
<code>\tex_mkern:D</code>	57, 59, 61, 64
<code>\tex_the:D</code>	193
<code>\textfont</code>	1601
<code>\textit</code>	301
<code>\textsuperscript</code>	302, 303
<code>\the</code>	147, 148, 1367, 1376
<code>\thetabularnote</code>	304
<code>\tikzexternaldisable</code>	1111
<code>\tikzset</code>	1113, 1170, 2215
tl commands:	
<code>\tl_clear:N</code>	3612, 3623, 3731, 3742
<code>\tl_clear_new:N</code>	3589, 3709
<code>\tl_const:Nn</code> ..	38, 39, 42, 43, 425, 1948, 1991
<code>\tl_gclear:N</code>	1374, 2212, 2228
<code>\tl_gclear_new:N</code>	
	1081, 1082, 1083, 1084, 1085, 1086, 1087
<code>\tl_gput_left:Nn</code>	903, 1387, 1396, 3554
<code>\tl_gput_right:Nn</code>	
	903, 1399, 1408, 1412, 1445, 1456,
	1469, 1483, 1491, 1507, 1529, 1535, 1537,
	1548, 1556, 1772, 3102, 3178, 3538, 3543,
	3832, 3969, 4321, 4490, 4500, 4741, 4793, 4801
<code>\tl_gset:Nn</code> ..	172, 176, 178, 1361, 1371, 2237
<code>\tl_if_blank:nTF</code> ..	3310, 3346, 3382, 3415, 4285
<code>\tl_if_blank_p:n</code>	3663, 3782
<code>\tl_if_empty:NTF</code>	
	1131, 1625, 2230, 3327, 3328, 3361, 3362,
	3601, 3605, 3616, 3720, 3724, 3735, 4291, 4488
<code>\tl_if_empty:nTF</code>	
	544, 667, 701, 717, 731, 747, 1759, 1786,
	2487, 2529, 2581, 2655, 2706, 3166, 3241,
	3258, 3314, 3350, 3386, 3419, 4295, 4298, 4910
<code>\tl_if_empty_p:N</code>	2787, 2788
<code>\tl_if_empty_p:n</code>	1616
<code>\tl_if_eq:NNTF</code>	2754, 4005, 4008
<code>\tl_if_eq:nnTF</code>	556, 678, 1762, 1764
<code>\tl_if_exist:NTF</code>	1121
<code>\tl_if_in:NnTF</code>	3324, 3358
<code>\tl_if_single_token:nTF</code>	686
<code>\tl_item:Nn</code>	175, 176, 178
<code>\tl_lower_case:n</code>	3050
<code>\tl_map_inline:nn</code>	1760
<code>\tl_new:N</code>	174, 177, 230,
	258, 259, 262, 265, 281, 282, 299, 424, 447, 449
<code>\tl_put_left:Nn</code>	1005
<code>\tl_put_right:Nn</code>	546, 1124, 1190

<code>\tl_range:nnn</code>	78	1224, 1231, 1257, 1258, 1262, 1263, 2063, 2083
<code>\tl_set:Nn</code>	175,	<code>\use:n</code>
3329, 3331, 3363, 3365, 3433, 3435, 3588, 4301		3259, 3558
<code>\tl_set_eq:NN</code> 426, 3602, 3721, 4010, 4040, 4293		<code>\usepackage</code>
<code>\tl_set_rescan:Nnn</code>		4862, 4872
1774, 2936, 3099, 3175, 3253		<code>\usepgfmodule</code>
		2
<code>\tl_to_str:n</code>	4881	V
<code>\g_tmpa_tl</code>	172, 175, 178	vbox commands:
<code>\l_tmpb_tl</code>	3306,	<code>\vbox:n</code>
3328, 3329, 3330, 3331, 3332, 3362, 3363,		<code>\vbox_set_top:Nn</code>
3364, 3365, 3370, 3393, 3398, 3399, 3402,		<code>\vbox_to_ht:nn</code>
3423, 3427, 3428, 3434, 3435, 3438, 3588,		<code>\vbox_to_zero:n</code>
3628, 3632, 3638, 3640, 3645, 3710, 3721,		63
3730, 3751, 3757, 3764, 3842, 3843, 3853, 3854		822
<code>\l_tmpc_tl</code>	3589, 3601, 3602,	<code>\vcenter</code>
3605, 3610, 3612, 3616, 3621, 3623, 3709,		<code>\Vdots</code>
3720, 3721, 3724, 3729, 3731, 3735, 3740, 3742		<code>\vdots</code>
token commands:		<code>\vdotsfor</code>
<code>\token_to_str</code>	4987	<code>\Vfill</code>
<code>\token_to_str:N</code>	4911, 4992,	<code>\vline</code>
5012, 5018, 5055, 5068, 5085, 5095, 5124, 5319		<code>\VNiceMatrix</code>
		<code>\vNiceMatrix</code>
U		<code>\vrule</code>
<code>\unskip</code>	1641	<code>\vskip</code>
use commands:		<code>\vtop</code>
<code>\use:N</code>	906, 1206, 1213,	X
		<code>\xglobal</code>
		770, 777, 1965, 2008

Contents

1	The environments of this package	2
2	The vertical space between the rows	2
3	The vertical position of the arrays	3
4	The blocks	4
4.1	General case	4
4.2	The mono-column blocks	5
4.3	The mono-row blocks	5
4.4	The mono-cell blocks	5
4.5	A small remark	6
5	The rules	6
5.1	Some differences with the classical environments	6
5.1.1	The vertical rules	6
5.1.2	The command <code>\cline</code>	7
5.2	The thickness and the color of the rules	7
5.3	The keys <code>hlines</code> and <code>vlines</code>	8
5.4	The key <code>hvlines</code>	8
5.5	The key <code>hvlines-except-corners</code>	8
5.6	The command <code>\diagbox</code>	9
5.7	Dotted rules	9
6	The color of the rows and columns	10
6.1	Use of <code>colortbl</code>	10
6.2	The tools of <code>nicematrix</code> in the code-before	10
6.3	Color tools with the syntax of <code>colortbl</code>	13

7	The width of the columns	14
8	The exterior rows and columns	15
9	The continuous dotted lines	16
9.1	The option nullify-dots	18
9.2	The commands <code>\Hdotsfor</code> and <code>\Vdotsfor</code>	18
9.3	How to generate the continuous dotted lines transparently	19
9.4	The labels of the dotted lines	20
9.5	Customization of the dotted lines	20
9.6	The dotted lines and the rules	21
10	The code-after	21
11	The notes in the tabulars	22
11.1	The footnotes	22
11.2	The notes of tabular	22
11.3	Customisation of the tabular notes	23
11.4	Use of <code>{NiceTabular}</code> with <code>threeparttable</code>	25
12	Other features	26
12.1	Use of the column type <code>S</code> of <code>siunitx</code>	26
12.2	Alignment option in <code>{NiceMatrix}</code>	26
12.3	The command <code>\rotate</code>	26
12.4	The option <code>small</code>	27
12.5	The counters <code>iRow</code> and <code>jCol</code>	27
12.6	The option <code>light-syntax</code>	28
12.7	The environment <code>{NiceArrayWithDelims}</code>	28
13	Use of Tikz with <code>nicematrix</code>	28
13.1	The nodes corresponding to the contents of the cells	28
13.2	The “medium nodes” and the “large nodes”	29
13.3	The “row-nodes” and the “col-nodes”	31
14	API for the developpers	31
15	Technical remarks	32
15.1	Definition of new column types	32
15.2	Diagonal lines	33
15.3	The “empty” cells	33
15.4	The option <code>exterior-arraycolsep</code>	34
15.5	Incompatibilities	34
16	Examples	34
16.1	Notes in the tabulars	34
16.2	Dotted lines	35
16.3	Dotted lines which are no longer dotted	37
16.4	Width of the columns	38
16.5	How to highlight cells of the matrix	38
16.6	Direct use of the Tikz nodes	41
17	Implementation	42
18	History	164
	Index	169