

# The package `nicematrix`\*

F. Pantigny  
 fpantigny@wanadoo.fr

March 23, 2020

## Abstract

The LaTeX package `nicematrix` provides new environments similar to the classical environments `{array}` and `{matrix}` but with some additional features. Among these features are the possibilities to fix the width of the columns and to draw continuous ellipsis dots between the cells of the array.

## 1 Presentation

This package can be used with `xelatex`, `lualatex`, `pdflatex` but also by the classical workflow `latex-dvips-ps2pdf` (or Adobe Distiller). Two or three compilations may be necessary. This package requires and **loads** the packages `expl3`, `l3keys2e`, `xparse`, `array`, `amsmath`, `pgfcore` and the module `shapes` of PGF (`tikz` is *not* loaded). The final user only has to load the extension with `\usepackage{nicematrix}`.

This package provides some new tools to draw mathematical matrices. The main features are the following:

- continuous dotted lines<sup>1</sup>;
- exterior rows and columns for labels;
- a control of the width of the columns.

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

A command `\NiceMatrixOptions` is provided to fix the options (the scope of the options fixed by this command is the current TeX group).

### An example for the continuous dotted lines

For example, consider the following code which uses an environment `{pmatrix}` of `amsmath`.

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

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

This code composes the matrix  $A$  on the right.

Now, if we use the package `nicematrix` with the option **transparent**, the same code will give the result on the right.

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

\*This document corresponds to the version 3.14 of `nicematrix`, at the date of 2020/03/23.

<sup>1</sup>If the class option **draft** is used, these dotted lines will not be drawn for a faster compilation.

## 2 The environments of this extension

The extension `nicematrix` defines the following new environments.

<code>{NiceMatrix}</code>	<code>{NiceArray}</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>		<code>{NiceArrayWithDelims}</code>

By default, the environments `{NiceMatrix}`, `{pNiceMatrix}`, `{bNiceMatrix}`, `{BNiceMatrix}`, `{vNiceMatrix}` and `{VNiceMatrix}` behave almost exactly as the corresponding environments of `amsmath`: `{matrix}`, `{pmatrix}`, `{bmatrix}`, `{Bmatrix}`, `{vmatrix}` and `{Vmatrix}`.

The environment `{NiceArray}` is similar to the environment `{array}` of the package `{array}`. However, for technical reasons, in the preamble of the environment `{NiceArray}`, the user must use the letters `L`, `C` and `R` instead of `l`, `c` and `r`. It's possible to use the constructions `w{...}{...}`, `W{...}{...}`<sup>2</sup>, `l`, `>{...}`, `<{...}`, `@{...}`, `!{...}` and `*{n}{...}` but the letters `p`, `m` and `b` should not be used. See p. 8 the section relating to `{NiceArray}`.

## 3 The continuous dotted lines

Inside the environments of the extension `nicematrix`, new commands are defined: `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, and `\Iddots`. These commands are intended to be used in place of `\dots`, `\cdots`, `\vdots`, `\ddots` and `\iddots`.<sup>3</sup>

Each of them must be used alone in the cell of the array and it draws a dotted line between the first non-empty cells<sup>4</sup> on both sides of the current cell. Of course, for `\Ldots` and `\Cdots`, it's an horizontal line; for `\Vdots`, it's a vertical line and for `\Ddots` and `\Iddots` diagonal ones. It's possible to change the color of these lines with the option `color`.<sup>5</sup>

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

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

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}
```

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

<sup>2</sup>However, for the columns of type `w` and `W`, the cells are composed in math mode (in the environments of `nicematrix`) whereas in `{array}` of `array`, they are composed in text mode.

<sup>3</sup>The command `\iddots`, defined in `nicematrix`, is a variant of `\ddots` with dots going forward. If `mathdots` is loaded, the version of `mathdots` is used. It corresponds to the command `\adots` of `unicode-math`.

<sup>4</sup>The precise definition of a “non-empty cell” is given below (cf. p. 17).

<sup>5</sup>It's also possible to change the color of all theses dotted lines with the option `xdots/color` (`xdots` to remind that it works for `\Cdots`, `\Ldots`, `\Vdots`, etc.): cf. p. 5.

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

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

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

In the first column of this example, there are two instructions `\Vdots` but only one dotted line is drawn (there is no overlapping graphic objects in the resulting PDF<sup>6</sup>).

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

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

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

There are also other means to change the size of the matrix. Someone might want to use the optional argument of the command `\` for the vertical dimension and a command `\hspace*` in a cell for the horizontal dimension.<sup>7</sup>

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

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

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

### 3.1 The option `nullify-dots`

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

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

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

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

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

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

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

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

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

<sup>6</sup>And it's not possible to draw a `\Ldots` and a `\Cdots` line between the same cells.

<sup>7</sup>In `nicematrix`, one should use `\hspace*` and not `\hspace` for such an usage because `nicematrix` loads `array`. One may also remark that it's possible to fix the width of a column by using the environment `{NiceArray}` (or one of its variants) with a column of type `w` or `W`: see p. 11

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

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

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

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

**There must be no space before the opening bracket ( [ ) of the options of the environment.**

### 3.2 The command `\Hdotsfor`

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

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

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

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

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

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

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

The command `\hdotsfor` of `amsmath` takes an optional argument (between square brackets) which is used for fine tuning of the space between two consecutive dots. For homogeneity, `\Hdotsfor` has also an optional argument but this argument is discarded silently.

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

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

The package `nicematrix` provides an option called `transparent` for using existing code transparently in the environments of the `amsmath`: `{matrix}`, `{pmatrix}`, `{bmatrix}`, etc. In fact, this option is an alias for the conjunction of two options: `renew-dots` and `renew-matrix`.<sup>8</sup>

- The option `renew-dots`

With this option, the commands `\ldots`, `\cdots`, `\vdots`, `\ddots`, `\iddots`<sup>3</sup> and `\hdotsfor` are redefined within the environments provided by `nicematrix` and behave like `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots` and `\Hdotsfor`; the command `\dots` (“automatic dots” of `amsmath`) is also redefined to behave like `\Ldots`.

<sup>8</sup>The options `renew-dots`, `renew-matrix` and `transparent` can be fixed with the command `\NiceMatrixOptions` like the other options. However, they can also be fixed as options of the command `\usepackage` (it's an exception for these three specific options.)

- The option `renew-matrix`

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

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

```
\NiceMatrixOptions{transparent}
\begin{pmatrix}
1 & & \cdots & & \cdots & & 1 & & \\
0 & & \ddots & & & & & & \vdots \\
\vdots & & \ddots & & \ddots & & \vdots & & \\
0 & & \cdots & & 0 & & & & 1
\end{pmatrix}
```

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

### 3.4 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 in p. 7) 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. 9.

#### The option `xdots/shorten`

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

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

It should be pointed that, by default, the lines drawn by Tikz with the parameter `dotted` are composed of square dots (and not rounded ones).<sup>9</sup>

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

In order to provide lines with rounded dots in the style of those provided by `\ldots` (at least with the *Computer Modern* fonts), the extension `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`.

<sup>9</sup>The first reason of this behaviour is that the PDF format includes a description for dashed lines. The lines specified with this descriptor are displayed very efficiently by the PDF readers. It’s easy, starting from these dashed lines, to create a line composed by square dots whereas a line of rounded dots needs a specification of each dot in the PDF file.

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

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

## 4 The PGF/Tikz nodes created by nicematrix

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. However, the user may wish to use directly these nodes. It's possible (if Tikz has been loaded<sup>10</sup>). First, the user have to give a name to the array (with the key called `name`). Then, the nodes are accessible through the names “`name-i-j`” where `name` is the name given to the array and `i` and `j` the numbers of the row and the column of the considered cell.

```
\begin{pNiceMatrix}[name=mymatrix]
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{pNiceMatrix}
```

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

```
\tikz[remember picture,overlay]
\draw (mymatrix-2-2) circle (2mm) ;
```

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

In the following example, we have underlined all the nodes of the matrix.

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

In fact, the package `nicematrix` can create “extra nodes”: the “medium nodes” and the “large nodes”. The first ones are created with the option `create-medium-nodes` and the second ones with the option `create-large-nodes`.<sup>11</sup>

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.

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

<sup>10</sup>We remind that, since the version 3.13, `nicematrix` doesn't load Tikz by default by only PGF (Tikz is a layer over PGF).

<sup>11</sup>There 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 “large nodes” are constructed by adding the suffix “-large” to the names of the “normal nodes”. In the following example, we have underlined the “large nodes”. We consider that this example is self-explanatory.<sup>12</sup>

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

The “large nodes” of the first column and last column may appear too small for some usage. That’s why it’s possible to use the options `left-margin` and `right-margin` to add space on both sides of the array and also space in the “large nodes” of the first column and last column. In the following example, we have used the options `left-margin` and `right-margin`.<sup>13</sup>

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

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.

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

In this case, if we want a control over the height of the rows, we can add a `\strut` in each row of the array.

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

We explain below how to fill the nodes created by `nicematrix` (cf. p. 21).

## 5 The code-after

The option `code-after` may be used to give some code that will be excuted after the construction of the matrix (and thus after the construction of all the nodes).

**If Tikz is loaded**<sup>14</sup>, one may access to that nodes with classical Tikz instructions. The nodes should be designed as *i-j* (without the prefix corresponding to the name of the environment).

Moreover, a special command, called `\line`, is available to draw directly dotted lines between nodes.

```
\begin{pNiceMatrix}[code-after = {\line{1-1}{3-3}[color=blue]}]
0 & 0 & 0 \\
0 & & 0 \\
0 & 0 & 0
\end{pNiceMatrix}
```

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

<sup>12</sup>There is no “large nodes” created in the exterior rows and columns (for these rows and columns, cf. p. 9).

<sup>13</sup>The options `left-margin` and `right-margin` take dimensions as values but, if no value is given, the default value is used, which is `\arraycolsep` (by default: 5 pt). There is also an option `margin` to fix both `left-margin` and `right-margin` to the same value.

<sup>14</sup>We remind that, since the version 3.13, `nicematrix` doesn’t load Tikz by default but only PGF (Tikz is a layer over PGF).

## 6 The environment `{NiceArray}`

The environment `{NiceArray}` is similar to the environment `{array}`. As for `{array}`, the mandatory argument is the preamble of the array. However, for technical reasons, in this preamble, the user must use the letters `L`, `C` and `R`<sup>15</sup> instead of `l`, `c` and `r`. It's possible to use the constructions `w{...}{...}`, `W{...}{...}`, `l`, `>{...}`, `<{...}`, `@{...}`, `!{...}` and `*{n}{...}` but the letters `p`, `m` and `b` should not be used.<sup>16</sup>

The environment `{NiceArray}` accepts the options available for `{pNiceMatrix}` and its variants but also a option `baseline` whose value is an integer which indicates the number of the row whose baseline is used as baseline for the environment `{NiceArray}`.

```
$A =
\begin{NiceArray}{CCCC}[hvlines,baseline=2]
1 & 2 & 3 & 4 \\
1 & 2 & 3 & 4 \\
1 & 2 & 3 & 4 \\
\end{NiceArray}$
(The option hvlines is presented further: cf. p. 15.)
```

$$A = \begin{array}{|c|c|c|c|} \hline 1 & 2 & 3 & 4 \\ \hline 1 & 2 & 3 & 4 \\ \hline 1 & 2 & 3 & 4 \\ \hline \end{array}$$

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 environment `{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 `{array}` of `array`, one must use `\firsthline`<sup>17</sup>).

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

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

1. an item
2. 

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

With `{NiceArray}`, it's possible to draw vertical rules:

```
$\left[\begin{NiceArray}{CCCC|C}
a_1 & ? & & \cdots & ? & ? \\
0 & & & \ddots & \vdots & \vdots \\
\vdots & & \ddots & \ddots & ? & \\
0 & & \cdots & 0 & a_n & ? \\
\end{NiceArray}\right]$
```

$$\left[ \begin{array}{cccc|c} a_1 & ? & \cdots & ? & ? \\ 0 & & \ddots & \vdots & \vdots \\ \vdots & & \ddots & \ddots & ? \\ 0 & \cdots & 0 & a_n & ? \end{array} \right]$$

<sup>15</sup>The column types `L`, `C` and `R` are defined locally inside `{NiceArray}` with `\newcolumntype` of `array`. This definition overrides an eventual previous definition. In fact, the column types `w` and `W` are also redefined.

<sup>16</sup>In a command `\multicolumn`, one should also use the letters `L`, `C`, `R`.

<sup>17</sup>It's also possible to use `\firsthline` with `{NiceArray}`.



In fact, there is also variants for the environment `{NiceArray}`: `{pNiceArray}`, `{bNiceArray}`, `{BNiceArray}`, `{vNiceArray}` and `{VNiceArray}`. The key `baseline` is not available for these environments. In the following example, we use an environment `{pNiceArray}` (we don't use `{pNiceMatrix}` because we want to use the types L and R — in `{pNiceMatrix}`, all the columns are of type C).

```


$$\begin{array}{ccccccc} a_{11} & & \cdots & & a_{1n} \\ a_{21} & & & & a_{2n} \\ \vdots & & & & \vdots \\ a_{n-1,1} & & \cdots & & a_{n-1,n} \end{array}$$


```

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.

```


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


```

## 7 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”.

```


$$\begin{array}{ccccccc} & C_1 & & \cdots & & C_4 & \\ L_1 & a_{11} & a_{12} & a_{13} & a_{14} & L_1 \\ \vdots & a_{21} & a_{22} & a_{23} & a_{24} & \vdots \\ & a_{31} & a_{32} & a_{33} & a_{34} & \\ L_4 & a_{41} & a_{42} & a_{43} & a_{44} & L_4 \\ & C_1 & & \cdots & & C_4 & \end{array}$$


```

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

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 `{NiceArray}` and `{pNiceArray}`, the number of columns can obviously be computed from the preamble.
  - When the option `light-syntax` (cf. p. 15) 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}
```

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

#### Remarks

- As shown in the previous example, an horizontal rule (drawn by `\hline`) doesn't extend in the exterior columns and a vertical rule (specified by a `|` in the preamble of the array) doesn't extend in the exterior rows.<sup>18</sup>  
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 16.
- 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. 11) 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).

## 8 The dotted lines to separate rows or columns

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

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

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

<sup>18</sup>The latter is not true when the extension `arydshln` is loaded besides `nicematrix`. In fact, `nicematrix` and `arydshln` are not totally compatible because `arydshln` redefines many internals of `array`. On another hand, if one really wants a vertical rule running in the first and in the last row, he should use `!\vline` instead of `|` in the preamble of the array.

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

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

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

These dotted lines do *not* extend in the potential exterior rows and columns.

```
\begin{pNiceArray}{CCC:C}[
  first-row,last-col,
  code-for-first-row = \color{blue}\scriptstyle,
  code-for-last-col = \color{blue}\scriptstyle ]
C_1 & C_2 & C_3 & C_4 \\
1 & 2 & 3 & 4 & L_1 \\
5 & 6 & 7 & 8 & L_2 \\
9 & 10 & 11 & 12 & L_3 \\
\hdottedline
13 & 14 & 15 & 16 & L_4
\end{pNiceArray}
```

$$\begin{array}{cccc:c} C_1 & C_2 & C_3 & C_4 & \\ \left(\begin{array}{cccc:c} 1 & 2 & 3 & 4 & L_1 \\ 5 & 6 & 7 & 8 & L_2 \\ 9 & 10 & 11 & 12 & L_3 \\ 13 & 14 & 15 & 16 & L_4 \end{array}\right) \end{array}$$

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`. For example, in this document, we have loaded the extension `arydshln` which uses the letter “:” to specify a vertical dashed line. Thus, by using `letter-for-dotted-lines`, we can use the vertical lines of both `arydshln` and `nicematrix`.

```
\NiceMatrixOptions{letter-for-dotted-lines = I}
\arrayrulecolor{blue}
\left(\begin{NiceArray}{C|C:C|C}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12
\end{NiceArray}\right)
\arrayrulecolor{black}
```

$$\left(\begin{array}{c|cc|c} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{array}\right)$$

We have used the command `\arrayrulecolor` (de colortbl) to draw in blue the three rules.

*Remark :* In the extension `array` (on which the extension `nicematrix` relies), horizontal and vertical rules make the array larger or wider by a quantity equal to the width of the rule<sup>19</sup>. In `nicematrix`, the dotted lines drawn by `\hdottedline` and “:” do likewise.

## 9 The width of the columns

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

```
\left(\begin{NiceArray}{wc{1cm}CC}
1 & 12 & -123 \\
12 & 0 & 0 \\
4 & 1 & 2
\end{NiceArray}\right)
```

$$\left(\begin{array}{cc} 1 & 12 & -123 \\ 12 & 0 & 0 \\ 4 & 1 & 2 \end{array}\right)$$

In the environments of `nicematrix`, it's also possible to fix the *minimal* width of all the columns of a matrix directly with the option `columns-width`.

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

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

<sup>19</sup>In fact, this is true only for `\hline` and “|” but not for `\cline`.

Note that the space inserted between two columns (equal to 2 `\arraycolsep`) is not suppressed (of course, it's possible to suppress this space by setting `\arraycolsep` equal to 0 pt).

It's possible to give the special value `auto` to the option `columns-width`: all the columns of the array will have a width equal to the widest cell of the array.<sup>20</sup>

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

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

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

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

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

But it's also possible to fix a zone where all the matrices will have their columns of the same width, equal to the widest cell of all the matrices. This construction uses the environment `{NiceMatrixBlock}` with the option `auto-columns-width`<sup>21</sup>. The environment `{NiceMatrixBlock}` has no direct link with the command `\Block` presented just below (cf. p. 12).

```
\begin{NiceMatrixBlock}[auto-columns-width]
\begin{pNiceMatrix}
a & b \\ c & d \\
\end{pNiceMatrix}
=
\begin{pNiceMatrix}
1 & 1245 \\ 345 & 2 \\
\end{pNiceMatrix}
\end{NiceMatrixBlock}
```

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

Several compilations may be necessary to achieve the job.

## 10 Block matrices

This section has no direct link with the previous one where an environment `{NiceMatrixBlock}` was introduced.

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

The command `\Block` must be used in the upper leftmost cell of the array with two arguments. The first argument is the size of the block with the syntax `i-j` where *i* is the number of rows of the block and *j* its number of columns. The second argument is the content of the block (composed in math mode). A Tikz node corresponding to the merged cells is created with the name “*i-j-block*”. 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`.

In the following examples, we use the command `\arrayrulecolor` of `colortbl`.

<sup>20</sup>The result is achieved with only one compilation (but Tikz will have written informations in the `.aux` file and a message requiring a second compilation will appear).

<sup>21</sup>At this time, this is the only usage of the environment `{NiceMatrixBlock}` but it may have other usages in the future.

```

\arrayrulecolor{cyan}
$\begin{bNiceArray}{CCC|C}[margin]
\Block{3-3}{A} & & 0 \\
& \hspace*{1cm} & \Vdots \\
& & 0 \\
\hline
0 & \Cdots & 0 & 0
\end{bNiceArray}$
\arrayrulecolor{black}

```

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

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

```

\arrayrulecolor{cyan}
$\begin{bNiceArray}{CCC|C}[margin]
\Block{3-3}<\Large>{A} & & 0 \\
& \hspace*{1cm} & \Vdots \\
& & 0 \\
\hline
0 & \Cdots & 0 & 0
\end{bNiceArray}$
\arrayrulecolor{black}

```

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

For technical reasons, you can’t write `\Block{i-j}<>`. But you can write `\Block{i-j}<><>` with the expected result.

## 11 Advanced features

### 11.1 Aligment option in NiceMatrix

The environments without preamble (`{NiceMatrix}`, `{pNiceMatrix}`, `{bNiceMatrix}`, etc.) provide two options `l` and `r` (equivalent at `L` and `R`) which generate all the columns aligned leftwards (or rightwards).<sup>22</sup>

```

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

```

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

### 11.2 The command `\rotate`

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

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

```

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

```

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

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

<sup>22</sup>This is a part of the functionality provided by the environments `{pmatrix*}`, `{bmatrix*}`, etc. of `mathtools`.

```

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

```

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

image of  $e_1$   $e_2$   $e_3$

### 11.3 The option small

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

```

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

```

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

One should note that the environment `{NiceMatrix}` with the option `small` is not composed *exactly* as the environment `{smallmatrix}`. Indeed, all the environments of `nicematrix` are constructed upon `{array}` (of the extension `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.

### 11.4 The counters iRow and jCol

In the cells of the array, it's possible to use the LaTeX counters `iRow` and `jCol` which represent the number of the current row and the number of the current column<sup>23</sup>. Of course, the user must not change the value of these counters which are used internally by `nicematrix`.

In the `code-after` (cf. p. 7), `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).

```

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

```

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

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

<sup>23</sup>We recall that the exterior “first row” (if it exists) has the number 0 and that the exterior “first column” (if it exists) has also the number 0.

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

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

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

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

## 11.5 The options `hlines`, `vlines` and `hvlines`

You can add horizontal rules between rows in the environments of `nicematrix` with the usual command `\hline` and you can use the specifier “|” to add vertical rules. However, by convenience, the extension `nicematrix` also provides the option `hlines` (resp. `vlines`) which will draw all the horizontal (resp. vertical) rules (excepted, of course, the exterior rules corresponding to the exterior rows and columns). The key `hvlines` is an alias for the conjunction for the keys `hlines` et `vlines`.

In the following example, we use the command `\arrayrulecolor` of `colortbl`.

```
\arrayrulecolor{cyan}
$\begin{NiceArray}{CCCC}%
[hvlines,first-row,first-col]
% & e & a & b & c \\
e & e & a & b & c \\
a & a & e & c & b \\
b & b & c & e & a \\
c & c & b & a & e
\end{NiceArray}$
\arrayrulecolor{black}
```

	<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>

However, there is a difference between the key `vlines` and the use of the specifier “|” in the preamble of the environment: the rules drawn by `vlines` completely cross the double-rules drawn by `\hline\hline`.

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

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

For the environments with delimiters (for example `{pNiceArray}` or `{pNiceMatrix}`), the option `vlines` don't draw vertical rules on both sides, where are the delimiters (fortunately).

```
\setlength{\arrayrulewidth}{0.2pt}
$\begin{pNiceMatrix}[vlines]
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 2 & 3 & 4 & 5 & 6 \\
\end{pNiceMatrix}$
```

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

## 11.6 The option `light-syntax`

The option `light-syntax`<sup>24</sup> allows the user to compose the arrays with a lighter syntax, which gives a more readable 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.

<sup>24</sup>This option is inspired by the extension `spalign` of Joseph Rabinoff.

```

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

```

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

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

When the option `light-syntax` is used, it is not possible to put verbatim material (for example with the command `\verb`) in the cells of the array.<sup>25</sup>

## 11.7 Use of the column type S of siunitx

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

```

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

```

$$\begin{matrix} C_1 & \dots & C_n \\ \left( \begin{array}{ccc} 2.3 & 0 & \dots & 0 \\ 12.4 & \vdots & & \vdots \\ 1.45 & \vdots & & \vdots \\ 7.2 & 0 & \dots & 0 \end{array} \right) \end{matrix}$$

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

## 12 Technical remarks

### 12.1 Definition of new column types

The extension `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 eventual exterior row.

For example, one may wish to define a new column type `?` in order to draw a (black) heavy rule of width 1 pt. The following definition will do the job<sup>26</sup>:

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

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

```

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

```

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

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

### 12.2 Intersections of dotted lines

Since the version 3.1 of `nicematrix`, the dotted lines created by `\Cdots`, `\Ldots`, `\Vdots`, etc. can't intersect.<sup>27</sup> That means that a dotted line created by one these commands automatically stops when it arrives on a dotted line already drawn. Therefore, the order in which dotted lines are drawn is important. Here's that order (by design) : `\Hdotsfor`, `\Vdots`, `\Ddots`, `\Iddots`, `\Cdots` and `\Ldots`.

With this structure, it's possible to draw the following matrix.

<sup>25</sup>The reason is that, when the option `light-syntax` is used, the whole content of the environment is loaded as a TeX argument to be analyzed. The environment doesn't behave in that case as a standard environment of LaTeX which only put TeX commands before and after the content.

<sup>26</sup>The command `\vrule` is a TeX (and not LaTeX) command.

<sup>27</sup>On the contrary, dotted lines created by `\hdottedline`, the letter `:` in the preamble of the array and the command `\line` in the `code-after` can have intersections with other dotted lines.



```

 $\begin{pNiceMatrix}[nullify-dots]$ 
1 & 2 & 3 & \Cdots & n \\
1 & 2 & 3 & \Cdots & n \\
\vdots & \vdots & \vdots & \hspace*{15mm} & \vdots \\
& \Cdots & & & \\
& \Cdots & & & \\
& \Cdots & & & \\
\end{pNiceMatrix}

```

$$\begin{pmatrix} 1 & 2 & 3 & \cdots & n \\ 1 & 2 & 3 & \cdots & n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ & \vdots & & & \vdots \\ & \vdots & & & \vdots \end{pmatrix}$$

## 12.3 The names of the PGF nodes created by nicematrix

We have said that, when a name is given to an environment of `nicematrix`, it's possible to access the PGF/Tikz nodes through this name (cf. p. 6).

That's the recommended way to access these nodes. However, we describe now the internal names of these nodes.

The environments created by `nicematrix` are numbered by an internal global counter. The command `\NiceMatrixLastEnv` provides the number of the last environment of `nicematrix` (for LaTeX, it's a “fully expandable” command and not a counter).

For the environment of number  $n$ , the node in row  $i$  and column  $j$  has the name `nm-n-i-j`. The `medium` and `large` nodes have the same name, suffixed by `-medium` and `-large`.

## 12.4 Diagonal lines

By default, all the diagonal lines<sup>28</sup> of a same array are “parallelized”. That means that the first diagonal line is drawn and, then, the other lines are drawn parallel to the first one (by rotation around the left-most extremity of the line). That's why the position of the instructions `\Ddots` in the array can have a marked effect on the final result.

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

Example with parallelization (default):

```

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

```

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

```

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

```

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

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

The same example without parallelization:

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

## 12.5 The “empty” cells

An instruction like `\Ldots`, `\Cdots`, etc. tries to determine the first non-empty cells on both sides. However, an empty cell is not necessarily a cell with no TeX content (that is to say a cell with no token between the two ampersands `&`). Indeed, a cell which only contains `\hspace*{1cm}` may be considered as empty.

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

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

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

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

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

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

## 12.6 The option `exterior-arraycolsep`

The environment `{array}` inserts an horizontal space equal to `\arraycolsep` before and after each column. In particular, there is a space equal to `\arraycolsep` before and after the array. This feature of the environment `{array}` was probably not a good idea<sup>29</sup>. The environment `{matrix}` of `amsmath` and its variants (`{pmatrix}`, `{vmatrix}`, etc.) of `amsmath` prefer to delete these spaces with explicit instructions `\hskip -\arraycolsep`<sup>30</sup>. The extension `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).

## 12.7 The class option `draft`

When the class option `draft` is used, the dotted lines are not drawn, for a faster compilation.

## 12.8 A technical problem with the argument of `\`

For technical, reasons, if you use the optional argument of the command `\`, the vertical space added will also be added to the “normal” node corresponding at the previous node.

```
\begin{pNiceMatrix}
a & \frac{A}{B} \\
b & c
\end{pNiceMatrix}
```

$$\begin{pmatrix} a & \frac{A}{B} \\ b & c \end{pmatrix}$$

There are two solutions to solve this problem. The first solution is to use a TeX command to insert space between the rows.

```
\begin{pNiceMatrix}
a & \frac{A}{B} \\
\noalign{\kern2mm}
b & c
\end{pNiceMatrix}
```

$$\begin{pmatrix} a & \frac{A}{B} \\ b & c \end{pmatrix}$$

The other solution is to use the command `\multicolumn` in the previous cell.

```
\begin{pNiceMatrix}
a & \multicolumn{1}{C}{\frac{A}{B}} \\
b & c
\end{pNiceMatrix}
```

$$\begin{pmatrix} a & \frac{A}{B} \\ b & c \end{pmatrix}$$

<sup>29</sup>In the documentation of `{amsmath}`, we can read: *The extra space of `\arraycolsep` that `array` adds on each side is a waste so we remove it [in `{matrix}`] (perhaps we should instead remove it from `array` in general, but that's a harder task).*

<sup>30</sup>And not by inserting `@{}` on both sides of the preamble of the array. As a consequence, the length of the `\hline` is not modified and may appear too long, in particular when using square brackets

## 12.9 Obsolete environments

The version 3.0 of `nicematrix` has introduced the environment `{pNiceArray}` (and its variants) with the options `first-row`, `last-row`, `first-col` and `last-col`.

Consequently the following environments present in previous versions of `nicematrix` are deprecated:

- `{NiceArrayCwithDelims}` ;
- `{pNiceArrayC}`, `{bNiceArrayC}`, `{BNiceArrayC}`, `{vNiceArrayC}`, `{VNiceArrayC}` ;
- `{NiceArrayRCwithDelims}` ;
- `{pNiceArrayRC}`, `{bNiceArrayRC}`, `{BNiceArrayRC}`, `{vNiceArrayRC}`, `{VNiceArrayRC}`.

Since the version 3.12, the only way to use these environments is loading `nicematrix` with the option `obsolete-environments`.

However, these environments will certainly be completely deleted in a future version of `nicematrix`.

## 13 Examples

### 13.1 Dotted lines

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

```


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


```

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

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

```


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


```

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

An example with `\multicolumn`:

```


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


```

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

An example with `\Hdotsfor`:

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

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

An example for the resultant of two polynomials:

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

An example for a linear system (the vertical rule has been drawn in cyan with the tools of `colortbl`):

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

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

## 13.2 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{code-for-last-col = \color{blue}\scriptstyle}
\setlength{\extrarowheight}{1mm}
\quad $\begin{pNiceArray}{CCCC:C}[last-col]
1&1&1&1&\backslash
2&4&8&16&9&\backslash
3&9&27&81&36&\backslash
4&16&64&256&100&
\end{pNiceArray}$
...
\end{NiceMatrixBlock}
```

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

## 13.3 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 may 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`).

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

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

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` and `vlines` spread the cells (when the package `array` is loaded but, when the package `nicematrix` is loaded, `array` is always loaded).<sup>31</sup>

The package `nicematrix` is constructed upon the environment `{array}` and, therefore, it's possible to use the package `colortbl` in the environments of `nicematrix`. However, it's not always easy to do a fine tuning of `colortbl`. That's why we propose another 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}}

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

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

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

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

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

<sup>31</sup>On the other side, the command `\cline` doesn't spread the rows of the array.

```

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

```

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

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.

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

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.

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

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,
code-after = {\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) ; } ]
A_{11} & A_{12} & A_{13} & A_{14} \\
A_{21} & A_{22} & A_{23} & A_{24} \\
A_{31} & A_{32} & A_{33} & A_{34} \\
A_{41} & A_{42} & A_{43} & A_{44}
\end{pNiceArray}

```

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

### 13.4 Direct use of the Tikz nodes

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

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

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

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

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

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

```
&
```

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

```
\begin{bNiceArray}{C>{\strut}CCCC}[name=B,first-row]
```

```

& & C_j & \\
b_{11} & \Cdots & b_{1j} & \Cdots & b_{1n} \\
\Vdots & & \Vdots & & \Vdots \\
& & b_{kj} & & \\
& & \Vdots & & \\
b_{n1} & \Cdots & b_{nj} & \Cdots & b_{nn}
\end{bNiceArray} \\ \\

```

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}

```

$$L_i \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{i1} & \dots & a_{in} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \quad \begin{bmatrix} & \vdots & \\ & \vdots & \\ \dots & c_{ij} & \dots \end{bmatrix}$$

## 14 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 `\align` to begin the `\halign`.

## Declaration of the package and extensions loaded

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

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}
3 \RequirePackage{expl3}[2020/02/08]

```

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

```

4 \RequirePackage{l3keys2e}
5 \ProvidesExplPackage
6   {nicematrix}
7   {\myfiledate}
8   {\myfileversion}
9   {Mathematical matrices with PGF/TikZ}

```

The version of 2020/02/08 of expl3 has replaced `\l_keys_key_tl` by `\l_keys_key_str`. We have immediately changed in this file. Now, you test the existence of `\l_keys_key_str` in order to detect whether the version of LaTeX used by the final user is up to date.

```

10 \msg_new:nnn { nicematrix } { expl3-too-old }
11 {
12     Your-version-of-LaTeX~(especially-expl3)~is-too-old.~
13     You-can-go-on-but-you-will-probably-have-other-errors~
14     if-you-use-the-functionalities-of-nicematrix.
15 }
16 \cs_if_exist:NF \l_keys_key_str
17 { \msg_error:nn { nicematrix } { expl3-too-old } }

```

We test the class option `draft`. In this case, we raise the flag `\c_@@_draft_bool` because we won't draw the dotted lines if the option `draft` is used.

```

18 \bool_new:N \c_@@_draft_bool
19 \DeclareOption { draft } { \bool_set_true:N \c_@@_draft_bool }
20 \DeclareOption* { }
21 \ProcessOptions \relax

```

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

We load some packages.

```

22 \RequirePackage { array }
23 \RequirePackage { amsmath }
24 \RequirePackage { xparse } [ 2018-07-01 ]

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

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

```

## Technical definitions

```

34 \bool_new:N \c_@@_tikz_loaded_bool
35 \AtBeginDocument
36 {
37     \@ifpackageloaded { tikz }
38     {

```

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}` `{tikzpicture}` can't be done dynamically with a conditional because, when the external Tikz library, the pair `\tikzpicture-\endtikzpicture` (or `\begin{tikzpicture}-\end{tikzpicture}`) must be statically “visible” (even when externalization is not activated).

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

```

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

```

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.

```

48 \bool_new:N \c_@@_revtex_bool
49 \ifclassloaded { revtex4-1 }
50 { \bool_set_true:N \c_@@_revtex_bool }
51 { }
52 \ifclassloaded { revtex4-2 }
53 { \bool_set_true:N \c_@@_revtex_bool }
54 { }

```

The following message must be defined right now because it may be used during the loading of the package.

```

55 \@@_msg_new:nn { Draft-mode }
56 { The~compilation~is~in~draft~mode:~the~dotted~lines~won't~be~drawn. }
57 \bool_if:NT \c_@@_draft_bool { \msg_warning:nn { nicematrix } { Draft-mode } }

```

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

```

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

```

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

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.

```

72 \int_new:N \g_@@_env_int
73 \cs_new:Npn \@@_env: { nm - \int_use:N \g_@@_env_int }
74 \cs_new_protected:Npn \@@_qpoint: #1
75 { \pgfpointanchor { \@@_env: - #1 } { center } }

```

We also define a counter to count the environments `{NiceMatrixBlock}`.

```

76 \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).

```

77 \dim_new:N \l_@@_columns_width_dim

```

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

```

78 \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.

```

79 \bool_new:N \l_@@_in_env_bool

```

If the user uses `{NiceArray}` (and not another environment relying upon `{NiceArrayWithDelims}` like `{pNiceArray}`), we will raise the flag `\l_@@_NiceArray_bool`. We have to know that, because, in `{NiceArray}`, we won't use a structure with `\left` and `\right` and we will use the option of position (`t`, `b` or `c`).

```

80 \bool_new:N \l_@@_NiceArray_bool

81 \cs_new_protected:Npn \@@_test_if_math_mode:
82 {
83   \if_mode_math: \else:
84     \@@_fatal:n { Outside-math-mode }
85   \fi:
86 }

```

We have to know whether `colortbl` is loaded for the redefinition of `\everycr` and `\vline` and for the options `hlines` and `vlines`.

```

87 \bool_new:N \c_@@_colortbl_loaded_bool
88 \AtBeginDocument
89 {
90   \ifpackageloaded { colortbl }
91   {
92     \bool_set_true:N \c_@@_colortbl_loaded_bool
93     \cs_set_protected:Npn \@@_vline_i: { { \CT@arc@ \vline } }
94   }
95   { }
96 }
97 \colorlet { nicematrix-last-col } { . }
98 \colorlet { nicematrix-last-row } { . }

```

The length `\l_@@_inter_dots_dim` is the distance between two dots for the dotted lines. The default value is 0.45 em but it will be changed if the option `small` is used.

```

99 \dim_new:N \l_@@_inter_dots_dim
100 \dim_set:Nn \l_@@_inter_dots_dim { 0.45 em }

```

The length `\l_@@_xdots_shorten_dim` 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).

```

101 \dim_new:N \l_@@_xdots_shorten_dim
102 \dim_set:Nn \l_@@_xdots_shorten_dim { 0.3 em }

```

The length `\l_@@_radius_dim` is the radius of the dots for the dotted lines (for `\hdottedline` and `\dottedline` and for all the other 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 (to 0.37 pt).

```

103 \dim_new:N \l_@@_radius_dim
104 \dim_set:Nn \l_@@_radius_dim { 0.53 pt }

```

The name of the current environment or the current command (despite the name which contains `env`).

```

105 \str_new:N \g_@@_name_env_str

```

The string `\g_@@_com_or_env_str` 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*.

```

106 \str_new:N \g_@@_com_or_env_str
107 \str_set:Nn \g_@@_com_or_env_str { environment }

```

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

```

108 \cs_new:Npn \@@_full_name_env:
109 {
110   \str_if_eq:VnTF \g_@@_com_or_env_str { command }
111     { command \space \c_backslash_str \g_@@_name_env_str }
112     { environment \space \{ \g_@@_name_env_str \} }
113 }

```

```

114 \tl_new:N \g_@@_internal_code_after_tl
115 \tl_new:N \g_@@_code_after_tl

```

The counters `\l_@@_save_iRow_int` and `\l_@@_save_jCol_int` will be used to save the values of the eventual LaTeX counters `iRow` and `jCol`. These LaTeX counters will be restored at the end of the environment.

```
116 \int_new:N \l_@@_save_iRow_int
117 \int_new:N \l_@@_save_jCol_int
```

The TeX counters `\c@iRow` and `\c@jCol` will be created in the beginning of `{\NiceArrayWithDelims}` (if they don't exist previously).

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

119 \tl_new:N \l_@@_initial_suffix_tl
120 \tl_new:N \l_@@_initial_anchor_tl
121 \tl_new:N \l_@@_final_suffix_tl
122 \tl_new:N \l_@@_final_anchor_tl

123 \dim_new:N \l_@@_x_initial_dim
124 \dim_new:N \l_@@_y_initial_dim
125 \dim_new:N \l_@@_x_final_dim
126 \dim_new:N \l_@@_y_final_dim

127 \dim_new:N \l_tmpc_dim
128 \dim_new:N \l_tmpd_dim

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

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.

```
130 \tl_new:N \l_@@_xdots_line_style_tl
131 \tl_const:Nn \c_@@_standard_tl { standard }
132 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
```

## 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. As usual, the global version is for the passage in the `\group_insert_after:N`.

```
133 \int_new:N \l_@@_first_row_int
134 \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.

```
135 \int_new:N \l_@@_first_col_int
136 \int_set:Nn \l_@@_first_col_int 1
```

### • Last row

The counter `\l_@@_last_row_int` is the number of the eventual “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).

```
137 \int_new:N \l_@@_last_row_int
138 \int_set:Nn \l_@@_last_row_int { -2 }
```

If, in an environment like `{\pNiceArray}`, the option `last-row` is used without value, we will globally raise the following flag. It will be used to know if we have, after the construction of the array, to write in the `aux` file the number of the “last row”.<sup>32</sup>

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

<sup>32</sup>We can't use `\l_@@_last_row_int` for this usage because, if `nicematrix` has read its value from the `aux` file, the value of the counter won't be `-1` any longer.

Idem for `\l_@@_last_col_without_value_bool`

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

#### • Last column

For the eventual “last column”, we use an integer. A value of  $-2$  means that there is no last column. A value of  $-1$  means that there is a last column but we don’t know its value because the user has used the option `last-col` without value (it’s possible in an environment without preamble like `{pNiceMatrix}`). A value of  $0$  means that the option `last-col` has been used in an environment with preamble (like `{pNiceArray}`).

```
141 \int_new:N \l_@@_last_col_int
142 \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.

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

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

### 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`.

```
144 \bool_new:N \c_@@_siunitx_loaded_bool
145 \AtBeginDocument
146 {
147   \ifpackageloaded { siunitx }
148     { \bool_set_true:N \c_@@_siunitx_loaded_bool }
149     { }
150 }
```

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} }
    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 `\@temptokena` list, 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 `\@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:`.

```

151 \cs_set_protected:Npn \@@adapt_S_column:
152 {
153   \bool_if:NT \c_@@siunitx_loaded_bool
154   {
155     \group_begin:
156     \@temptokena = { }

```

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

```

157     \cs_set_eq:NN \NC@find \prg_do_nothing:
158     \NC@rewrite@S { }

```

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

```

159     \tl_gset:NV \g_tmpa_tl \@temptokena
160     \group_end:
161     \tl_new:N \c_@@table_collect_begin_tl
162     \tl_set:Nx \l_tmpa_tl { \tl_item:Nn \g_tmpa_tl 2 }
163     \tl_gset:Nx \c_@@table_collect_begin_tl { \tl_item:Nn \l_tmpa_tl 1 }
164     \tl_new:N \c_@@table_print_tl
165     \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).

```

166     \cs_gset_eq:NN \@@adapt_S_column: \prg_do_nothing:
167   }
168 }

```

The command `\@@renew_NC@rewrite@S:` will be used in each environment of `nicematrix` in order to “rewrite” the `S` column in each environment (only if the boolean `\c_@@siunitx_loaded_bool` is raised, of course).

```

169 \cs_new_protected:Npn \@@renew_NC@rewrite@S:
170 {
171   \renewcommand*{\NC@rewrite@S}[1] []
172   {
173     \@temptokena \exp_after:wN
174     {
175       \tex_the:D \@temptokena
176       > { \@@_Cell: \c_@@table_collect_begin_tl S {##1} }
177       c
178       < { \c_@@table_print_tl \@@_end_Cell: }
179     }
180     \NC@find
181   }
182 }

```

The following command is only for efficiency. It must *not* be protected because it will be used (for instance) in names of PGF nodes.

```

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

```

## Command for creation of rectangle nodes

The following command should be used in a `{pgfpicture}`. It creates an rectangular (empty but with a name) when the four corners are given.

**#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.

```
185 \cs_new_protected:Npn \@@_pgf_rect_node:nnnnn #1 #2 #3 #4 #5
186 {
187   \begin { pgfscope }
188   \pgfset
189   {
190     outer~sep = \c_zero_dim ,
191     inner~sep = \c_zero_dim ,
192     minimum~size = \c_zero_dim
193   }
194   \pgftransformshift { \pgfpoint { 0.5 * ( #2 + #4 ) } { 0.5 * ( #3 + #5 ) } }
195   \pgfnode
196   { rectangle }
197   { center }
198   {
199     \vbox_to_ht:nn
200     { \dim_abs:n { #5 - #3 } }
201     {
202       \vfill
203       \hbox_to_wd:nn { \dim_abs:n { #4 - #2 } } { }
204     }
205   }
206   { #1 }
207   { }
208   \end { pgfscope }
209 }
```

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

```
210 \cs_new_protected:Npn \@@_pgf_rect_node:nnn #1 #2 #3
211 {
212   \begin { pgfscope }
213   \pgfset
214   {
215     outer~sep = \c_zero_dim ,
216     inner~sep = \c_zero_dim ,
217     minimum~size = \c_zero_dim
218   }
219   \pgftransformshift { \pgfpoint scale { 0.5 } { \pgfpointadd { #2 } { #3 } } }
220   \pgfpointdiff { #3 } { #2 }
221   \pgfgetlastxy \l_tmpa_dim \l_tmpb_dim
222   \pgfnode
223   { rectangle }
224   { center }
225   {
226     \vbox_to_ht:nn
227     { \dim_abs:n \l_tmpb_dim }
228     { \vfill \hbox_to_wd:nn { \dim_abs:n \l_tmpa_dim } { } }
229   }
230   { #1 }
231   { }
232   \end { pgfscope }
233 }
```

## The options

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



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

The token list `\l_@@_baseline_str` will contain one of the three values `t`, `c` or `b` and will indicate the position of the environment as in the option of the environment `{array}`. For the environment `{pNiceMatrix}`, `{pNiceArray}` and their variants, the value will programmatically be fixed to `c`. For the environment `{NiceArray}`, however, the three values `t`, `c` and `b` are possible.

```
235 \str_new:N \l_@@_baseline_str
236 \str_set:Nn \l_@@_baseline_str c
```

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

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

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

```
238 \bool_new:N \l_@@_parallelize_diags_bool
239 \bool_set_true:N \l_@@_parallelize_diags_bool
```

The flag `\l_@@_hlines_bool` corresponds to the option `\hlines` and the flag `\l_@@_vlines_bool` to the option `\vlines`.

```
240 \bool_new:N \l_@@_hlines_bool
241 \bool_new:N \l_@@_vlines_bool
```

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.).

```
242 \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).

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

The token list `\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.

```
244 \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”.

```
245 \bool_new:N \l_@@_medium_nodes_bool
246 \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`).

```
247 \dim_new:N \l_@@_left_margin_dim
248 \dim_new:N \l_@@_right_margin_dim
```

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

```
249 \dim_new:N \g_@@_width_last_col_dim
250 \dim_new:N \g_@@_width_first_col_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`.

```
251 \dim_new:N \l_@@_extra_left_margin_dim
252 \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.

```
253 \tl_new:N \l_@@_end_of_row_tl
254 \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 “:”.

```
255 \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.

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

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

```
257 \keys_define:nn { NiceMatrix / xdots }
258 {
259   line-style .code:n =
260   {
261     \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.

```
262   { \cs_if_exist_p:N \tikzpicture }
263   { \str_if_eq_p:nn { #1 } { standard } }
264   { \tl_set:Nn \l_@@_xdots_line_style_tl { #1 } }
265   { \@@_error:n { bad-option-for-line-style } }
266   } ,
267   line-style .value_required:n = true ,
268   color .tl_set:N = \l_@@_xdots_color_tl ,
269   color .value_required:n = true ,
270   shorten .dim_set:N = \l_@@_xdots_shorten_dim ,
271   shorten .value_required:n = true ,
272   unknown .code:n = \@@_error:n { Unknown-option-for-xdots }
273 }

274 \keys_define:nn { NiceMatrix / Global }
275 {
276   xdots .code:n = \keys_set:nn { NiceMatrix / xdots } { #1 } ,
277   max-delimiter-width .bool_set:N = \l_@@_max_delimiter_width_bool ,
278   light-syntax .bool_set:N = \l_@@_light_syntax_bool ,
279   light-syntax .default:n = true ,
280   end-of-row .tl_set:N = \l_@@_end_of_row_tl ,
281   end-of-row .value_required:n = true ,
282   code-for-first-col .tl_set:N = \l_@@_code_for_first_col_tl ,
283   code-for-first-col .value_required:n = true ,
284   code-for-last-col .tl_set:N = \l_@@_code_for_last_col_tl ,
285   code-for-last-col .value_required:n = true ,
286   code-for-first-row .tl_set:N = \l_@@_code_for_first_row_tl ,
287   code-for-first-row .value_required:n = true ,
288   code-for-last-row .tl_set:N = \l_@@_code_for_last_row_tl ,
289   code-for-last-row .value_required:n = true ,
290   small .bool_set:N = \l_@@_small_bool ,
291   hlines .bool_set:N = \l_@@_hlines_bool ,
292   vlines .bool_set:N = \l_@@_vlines_bool ,
293   hvlines .meta:n = { hlines , vlines } ,
294   parallelize-diags .bool_set:N = \l_@@_parallelize_diags_bool ,
```

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

```

295   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
296   renew-dots .value_forbidden:n = true ,
297   nullify-dots .bool_set:N = \l_@@_nullify_dots_bool ,

```

In some circumstances, the “medium nodes” are created automatically, for example when a dotted line has an “open” extremity (idem for the “large nodes”).

```

298   create-medium-nodes .bool_set:N = \l_@@_medium_nodes_bool ,
299   create-large-nodes .bool_set:N = \l_@@_large_nodes_bool ,
300   create-extra-nodes .meta:n =
301     { create-medium-nodes , create-large-nodes } ,
302   left-margin .dim_set:N = \l_@@_left_margin_dim ,
303   left-margin .default:n = \arraycolsep ,
304   right-margin .dim_set:N = \l_@@_right_margin_dim ,
305   right-margin .default:n = \arraycolsep ,
306   margin .meta:n = { left-margin = #1 , right-margin = #1 } ,
307   margin .default:n = \arraycolsep ,
308   extra-left-margin .dim_set:N = \l_@@_extra_left_margin_dim ,
309   extra-right-margin .dim_set:N = \l_@@_extra_right_margin_dim ,
310   extra-margin .meta:n =
311     { extra-left-margin = #1 , extra-right-margin = #1 } ,
312   extra-margin .value_required:n = true
313 }

```

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

```

314 \keys_define:nn { NiceMatrix / Env }
315 {
316   columns-width .code:n =
317     \str_if_eq:nnTF { #1 } { auto }
318     { \bool_set_true:N \l_@@_auto_columns_width_bool }
319     { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,
320   columns-width .value_required:n = true ,
321   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.

```

322   \legacy_if:nF { measuring@ }
323   {
324     \str_set:Nn \l_tmpa_str { #1 }
325     \seq_if_in:NVTF \g_@@_names_seq \l_tmpa_str
326     { \@@_error:nn { Duplicate-name } { #1 } }
327     { \seq_gput_left:NV \g_@@_names_seq \l_tmpa_str }
328     \str_set_eq:NN \l_@@_name_str \l_tmpa_str
329   } ,
330   name .value_required:n = true ,
331   code-after .tl_gset:N = \g_@@_code_after_tl ,
332   code-after .value_required:n = true ,
333   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
334   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
335   last-row .int_set:N = \l_@@_last_row_int ,
336   last-row .default:n = -1 ,
337 }

```

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

```

338 \keys_define:nn { NiceMatrix }
339 {
340   NiceMatrixOptions .inherit:n =
341     {
342       NiceMatrix / Global ,
343     } ,
344   NiceMatrixOptions / xdots .inherit:n = NiceMatrix / xdots ,

```

```

345 NiceMatrix .inherit:n =
346 {
347     NiceMatrix / Global ,
348     NiceMatrix / Env ,
349 } ,
350 NiceMatrix / xdots .inherit:n = NiceMatrix / xdots ,
351 NiceArray .inherit:n =
352 {
353     NiceMatrix / Global ,
354     NiceMatrix / Env ,
355 } ,
356 NiceArray / xdots .inherit:n = NiceMatrix / xdots ,
357 pNiceArray .inherit:n =
358 {
359     NiceMatrix / Global ,
360     NiceMatrix / Env ,
361 } ,
362 pNiceArray / xdots .inherit:n = NiceMatrix / xdots
363 }

```

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

```

364 \keys_define:nn { NiceMatrix / NiceMatrixOptions }
365 {

```

With the option `renew-matrix`, the environment `{matrix}` of `amsmath` and its variants are redefined to behave like the environment `{NiceMatrix}` and its variants.

```

366 renew-matrix .code:n = \@@_renew_matrix: ,
367 renew-matrix .value_forbidden:n = true ,
368 transparent .meta:n = { renew-dots , renew-matrix } ,
369 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}`.

```

370 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.

```

371 columns-width .code:n =
372 \str_if_eq:nnTF { #1 } { auto }
373 { \@@_error:n { Option~auto~for~columns~width } }
374 { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,

```

Usually, an error is raised when the user tries to give the same to name 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.

```

375 allow-duplicate-names .code:n =
376 \@@_msg_redirect_name:nn { Duplicate~name } { none } ,
377 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`).

```

378 letter-for-dotted-lines .code:n =
379 {
380     \int_compare:nTF { \tl_count:n { #1 } = 1 }
381     { \str_set:Nx \l_@@_letter_for_dotted_lines_str { #1 } }
382     { \@@_error:n { Bad~value~for~letter~for~dotted~lines } }
383 } ,
384 letter-for-dotted-lines .value_required:n = true ,
385 unknown .code:n = \@@_error:n { Unknown~key~for~NiceMatrixOptions }
386 }

```

```

387 \str_new:N \l_@@_letter_for_dotted_lines_str
388 \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.

```

389 \NewDocumentCommand \NiceMatrixOptions { m }
390 { \keys_set:nn { NiceMatrix / NiceMatrixOptions } { #1 } }

```

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

```

391 \keys_define:nn { NiceMatrix / NiceMatrix }
392 {
393   last-col .code:n = \tl_if_empty:nTF {#1}
394   {
395     \bool_set_true:N \l_@@_last_col_without_value_bool
396     \int_set:Nn \l_@@_last_col_int { -1 }
397   }
398   { \int_set:Nn \l_@@_last_col_int { #1 } } ,
399   l .code:n = \tl_set:Nn \l_@@_type_of_col_tl L ,
400   r .code:n = \tl_set:Nn \l_@@_type_of_col_tl R ,
401   L .code:n = \tl_set:Nn \l_@@_type_of_col_tl L ,
402   R .code:n = \tl_set:Nn \l_@@_type_of_col_tl R ,
403   unknown .code:n = \@@_error:n { Unknown-option-for~NiceMatrix }
404 }

```

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

```

405 \keys_define:nn { NiceMatrix / NiceArray }
406 {

```

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

```

407   c .code:n = \str_set:Nn \l_@@_baseline_str c ,
408   t .code:n = \str_set:Nn \l_@@_baseline_str t ,
409   b .code:n = \str_set:Nn \l_@@_baseline_str b ,
410   baseline .tl_set:N = \l_@@_baseline_str ,
411   baseline .value_required:n = true ,

```

In the environments `{NiceArray}` and its variants, the option `last-col` must be used without value because the number of columns of the array can be read in the preamble of the array.

```

412   last-col .code:n = \tl_if_empty:nF { #1 }
413   { \@@_error:n { last-col-non-empty-for~NiceArray } }
414   \int_zero:N \l_@@_last_col_int ,
415   unknown .code:n = \@@_error:n { Unknown-option-for~NiceArray }
416 }
417 \keys_define:nn { NiceMatrix / pNiceArray }
418 {
419   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
420   last-col .code:n = \tl_if_empty:nF {#1}
421   { \@@_error:n { last-col-non-empty-for~NiceArray } }
422   \int_zero:N \l_@@_last_col_int ,
423   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
424   last-row .int_set:N = \l_@@_last_row_int ,
425   last-row .default:n = -1 ,
426   unknown .code:n = \@@_error:n { Unknown-option-for~NiceMatrix }
427 }

```

## 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}`).

```
428 \cs_new_protected:Npn \@@_Cell:
429 {
```

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

```
430 \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.

```
431 \int_compare:nNnT \c@jCol = 1
432 { \int_compare:nNnT \l_@@_first_col_int = 1 \@@_begin_of_row: }
433 \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 `\c_math_toggle_token` also).

```
434 \hbox_set:Nw \l_@@_cell_box
435 \c_math_toggle_token
436 \bool_if:NT \l_@@_small_bool \scriptstyle
```

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.

```
437 \int_compare:nNnTF \c@iRow = 0
438 {
439   \int_compare:nNnT \c@jCol > 0
440   {
441     \l_@@_code_for_first_row_tl
442     \xglobal \colorlet { nicematrix-first-row } { . }
443   }
444 }
445 {
446   \int_compare:nNnT \c@iRow = \l_@@_last_row_int
447   {
448     \l_@@_code_for_last_row_tl
449     \xglobal \colorlet { nicematrix-last-row } { . }
450   }
451 }
452 }
```

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.

```
453 \cs_new_protected:Npn \@@_begin_of_row:
454 {
455   \int_gincr:N \c@iRow
456   \dim_gset_eq:NN \g_@@_dp_ante_last_row_dim \g_@@_dp_last_row_dim
457   \dim_gset:Nn \g_@@_dp_last_row_dim { \box_dp:N \@arstrutbox }
458   \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
459   \pgfpicture
460   \pgfrememberpicturepositiononpagetrue
461   \pgfcoordinate
462   { \@@_env: - row - \int_use:N \c@iRow - base }
463   \pgfpointorigin
464   \str_if_empty:NF \l_@@_name_str
465   {
466     \pgfnodealias
467     { \@@_env: - row - \int_use:N \c@iRow - base }
468     { \l_@@_name_str - row - \int_use:N \c@iRow - base }
469   }
470   \endpgfpicture
471 }
```

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 will be dynamically added to this command.

```

472 \cs_new_protected:Npn \@@_update_for_first_and_last_row:
473 {
474   \int_compare:nNnTF \c@iRow = 0
475   {
476     \dim_gset:Nn \g_@@_dp_row_zero_dim
477     { \dim_max:nn \g_@@_dp_row_zero_dim { \box_dp:N \l_@@_cell_box } }
478     \dim_gset:Nn \g_@@_ht_row_zero_dim
479     { \dim_max:nn \g_@@_ht_row_zero_dim { \box_ht:N \l_@@_cell_box } }
480   }
481   {
482     \int_compare:nNnT \c@iRow = 1
483     {
484       \dim_gset:Nn \g_@@_ht_row_one_dim
485       { \dim_max:nn \g_@@_ht_row_one_dim { \box_ht:N \l_@@_cell_box } }
486     }
487   }
488 }

489 \cs_new_protected:Npn \@@_end_Cell:
490 {
491   \c_math_toggle_token
492   \hbox_set_end:

```

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”).

```

493   \dim_gset:Nn \g_@@_max_cell_width_dim
494   { \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”.

```

495   \@@_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 use 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.

```

496   \bool_if:NTF \g_@@_empty_cell_bool
497   {
498     \box_use_drop:N \l_@@_cell_box
499     \bool_gset_false:N \g_@@_empty_cell_bool
500   }
501   {
502     \dim_compare:nNnTF { \box_wd:N \l_@@_cell_box } > \c_zero_dim
503     \@@_node_for_the_cell:
504     { \box_use_drop:N \l_@@_cell_box }
505   }
506   \bool_gset_false:N \g_@@_empty_cell_bool
507 }

```

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

```

508 \cs_new_protected:Npn \@@_node_for_the_cell:
509 {
510   \pgfpicture
511   \pgfsetbaseline \c_zero_dim
512   \pgfrememberpicturepositiononpagetrue
513   \pgfset
514   {
515     inner~sep = \c_zero_dim ,
516     minimum~width = \c_zero_dim
517   }
518   \pgfnode
519   { rectangle }
520   { base }
521   { \box_use_drop:N \l_@@_cell_box }
522   { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
523   { }
524   \str_if_empty:NF \l_@@_name_str
525   {
526     \pgfnodealias
527     { \l_@@_name_str - \int_use:N \c@iRow - \int_use:N \c@jCol }
528     { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
529   }
530   \endpgfpicture
531 }

```

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

For example, for the following matrix,

```

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

```

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

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

```

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

```

We begin with a test of the flag `\c_@@_draft_bool` because, if the key `draft` is used, the dotted lines are not drawn.

```

532 \bool_if:NTF \c_@@_draft_bool
533 { \cs_set_protected:Npn \@@_instruction_of_type:nn #1 #2 { } }
534 {
535   \cs_new_protected:Npn \@@_instruction_of_type:nn #1 #2
536   {

```

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).

```

537   \tl_gput_right:cx
538   { g_@@_ #1 _ lines _ tl }
539   {
540     \use:c { @@ _ draw _ #1 : nnn }
541     { \int_use:N \c@iRow }
542     { \int_use:N \c@jCol }

```

Maybe we should prevent the expansion of the list of key-value.

```

543     { #2 }
544   }
545 }
546 }

```



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`.

```

547 \cs_new_protected:Npn \@@_array:
548 {
549   \bool_if:NTF \c_@@_revtex_bool
550   {
551     \cs_set_eq:NN \@acol1 \@arrayacol
552     \cs_set_eq:NN \@acolr \@arrayacol
553     \cs_set_eq:NN \@acol \@arrayacol
554     \cs_set:Npn \@halignto { }
555     \@array@array
556   }
557   \array

```

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

```

558 [ \str_if_eq:VnTF \l_@@_baseline_str c c t ]
559 }

```

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.

```

560 \cs_set_eq:NN \@@_standard_ialign: \ialign

```

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

```

561 \cs_new:Npn \@@_everycr: { \noalign { \@@_everycr_i: } }
562 \cs_new_protected:Npn \@@_everycr_i:
563 {
564   \int_gzero:N \c@jCol

```

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

```

565 \hbox
566 {
567   \pgfpicture
568   \pgfrememberpicturerepositiononpagetrue
569   \pgfcoordinate { \@@_env: - row - \@@_succ:n \c@iRow }
570   \pgfpintorigin
571   \str_if_empty:NF \l_@@_name_str
572   {
573     \pgfnodealias
574     { \@@_env: - row - \int_use:N \c@iRow - row }
575     { \l_@@_name_str - row - \int_use:N \c@iRow - row }
576   }
577   \endpgfpicture
578 }

```

We add the potential horizontal lines specified by the option `hlines`.

```

579 \bool_if:NT \l_@@_hlines_bool
580 {

```

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.

```

581   \int_compare:nNnT \c@iRow > { -1 }
582   {
583     \bool_if:NF \g_@@_row_of_col_done_bool
584     {
585       \int_compare:nNnF \c@iRow = \l_@@_last_row_int
586       {
587         \bool_if:NTF \c_@@_colortbl_loaded_bool
588         { { \CT@arc@ \hrule height \arrayrulewidth } }
589         { \hrule height \arrayrulewidth }
590       }
591     }
592   }

```

```

593     }
594 }

```

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

```

595 \cs_new_protected:Npn \@@_pre_array:
596 {
597   \box_clear_new:N \l_@@_cell_box
598   \cs_if_exist:NT \theiRow
599     { \int_set_eq:NN \l_@@_save_iRow_int \c@iRow }
600   \int_gzero_new:N \c@iRow
601   \cs_if_exist:NT \thejCol
602     { \int_set_eq:NN \l_@@_save_jCol_int \c@jCol }
603   \int_gzero_new:N \c@jCol
604   \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}`).

```

605   \bool_if:NT \l_@@_small_bool
606   {
607     \cs_set:Npn \arraystretch { 0.47 }
608     \dim_set:Nn \arraycolsep { 1.45 pt }
609   }

```

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`.

```

610   \cs_set:Npn \ialign
611   {
612     \bool_if:NTF \c_@@_colortbl_loaded_bool
613     {
614       \CT@everycr
615       {
616         \noalign { \cs_gset_eq:NN \CT@row@color \prg_do_nothing: }
617         \@@_everycr:
618       }
619     }
620     { \everycr { \@@_everycr: } }
621     \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`<sup>33</sup> and `\extrarowheight` (of `array`). That box is inserted (via `\@arstrut`) in the beginning of each row of the array. That's why we use the dimensions of that box to initialize the variables which will be the dimensions of the potential first and last row of the environment. This initialization must be done after the creation of `\@arstrutbox` and that's why we do it in the `\ialign`.

```

622   \dim_gzero_new:N \g_@@_dp_row_zero_dim
623   \dim_gset:Nn \g_@@_dp_row_zero_dim { \box_dp:N \@arstrutbox }
624   \dim_gzero_new:N \g_@@_ht_row_zero_dim
625   \dim_gset:Nn \g_@@_ht_row_zero_dim { \box_ht:N \@arstrutbox }
626   \dim_gzero_new:N \g_@@_ht_row_one_dim
627   \dim_gset:Nn \g_@@_ht_row_one_dim { \box_ht:N \@arstrutbox }
628   \dim_gzero_new:N \g_@@_dp_ante_last_row_dim
629   \dim_gzero_new:N \g_@@_ht_last_row_dim
630   \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
631   \dim_gzero_new:N \g_@@_dp_last_row_dim
632   \dim_gset:Nn \g_@@_dp_last_row_dim { \box_dp:N \@arstrutbox }

```

---

<sup>33</sup>The option `small` of `nicematrix` changes (among other) the value of `\arraystretch`. This is done, of course, before the call of `{array}`.

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`.<sup>34</sup>

```
633     \cs_set_eq:NN \ialign \@@_standard_ialign:
634     \halign
635 }
```

We define the new column types L, C and R that must be used instead of l, c and r in the preamble of `{NiceArray}`.

```
636     \newcolumntype L { > \@@_Cell: l < \@@_end_Cell: }
637     \newcolumntype C { > \@@_Cell: c < \@@_end_Cell: }
638     \newcolumntype R { > \@@_Cell: r < \@@_end_Cell: }
```

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).

```
639     \cs_set_eq:NN \@@_ldots \ldots
640     \cs_set_eq:NN \@@_cdots \cdots
641     \cs_set_eq:NN \@@_vdots \vdots
642     \cs_set_eq:NN \@@_ddots \ddots
643     \cs_set_eq:NN \@@_iddots \iddots
644     \cs_set_eq:NN \firsthline \hline
645     \cs_set_eq:NN \lasthline \hline
646     \cs_set_eq:NN \Ldots \@@_Ldots
647     \cs_set_eq:NN \Cdots \@@_Cdots
648     \cs_set_eq:NN \Vdots \@@_Vdots
649     \cs_set_eq:NN \Ddots \@@_Ddots
650     \cs_set_eq:NN \Iddots \@@_Iddots
651     \cs_set_eq:NN \hdottedline \@@_hdottedline:
652     \cs_set_eq:NN \Hspace \@@_Hspace:
653     \cs_set_eq:NN \Hdotsfor \@@_Hdotsfor:
654     \cs_set_eq:NN \multicolumn \@@_multicolumn:nnn
655     \cs_set_eq:NN \Block \@@_Block:
656     \cs_set_eq:NN \rotate \@@_rotate:
657     \cs_set_eq:NN \OnlyMainNiceMatrix \@@_OnlyMainNiceMatrix:n
658     \bool_if:NT \l_@@_renew_dots_bool
659     {
660         \cs_set_eq:NN \ldots \@@_Ldots
661         \cs_set_eq:NN \cdots \@@_Cdots
662         \cs_set_eq:NN \vdots \@@_Vdots
663         \cs_set_eq:NN \ddots \@@_Ddots
664         \cs_set_eq:NN \iddots \@@_Iddots
665         \cs_set_eq:NN \dots \@@_Ldots
666         \cs_set_eq:NN \hdotsfor \@@_Hdotsfor:
667     }
```

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

```
668     \seq_gclear_new:N \g_@@_multicolumn_cells_seq
669     \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).

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

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

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

```
671     \int_gzero_new:N \g_@@_row_total_int
```

---

<sup>34</sup>The user will probably not employ directly `\ialign` in the array... but more likely environments that utilize `\ialign` internally (e.g.: `{substack}`).

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.

```
672 \int_gzero_new:N \g_@@_col_total_int
673 \cs_set_eq:NN \@ifnextchar \new@ifnextchar
```

We nullify the definitions of the column types `w` and `W` before their redefinition because we want to avoid a warning in the log file for a redefinition of a column type. We must put `\relax` and not `\prg_do_nothing:`.

```
674 \cs_set_eq:NN \NC@find@w \relax
675 \cs_set_eq:NN \NC@find@W \relax
676 \newcolumnntype w [ 2 ]
677 {
678   > {
679     \hbox_set:Nw \l_@@_cell_box
680     \@@_Cell:
681   }
682   c
683   < {
684     \@@_end_Cell:
685     \hbox_set_end:
```

The `\str_lowercase:n` is only for giving the user the ability to write `wC{1cm}` instead of `wc{1cm}` for homogeneity with the letters `L`, `C` and `R` used elsewhere in the preamble instead of `l`, `c` and `r`.

```
686 \makebox [ ##2 ] [ \str_lowercase:n { ##1 } ]
687 { \box_use_drop:N \l_@@_cell_box }
688 }
689 }
690 \newcolumnntype W [ 2 ]
691 {
692   > {
693     \hbox_set:Nw \l_@@_cell_box
694     \@@_Cell:
695   }
696   c
697   < {
698     \@@_end_Cell:
699     \hbox_set_end:
700     \cs_set_eq:NN \hss \hfil
701     \makebox [ ##2 ] [ \str_lowercase:n { ##1 } ]
702     { \box_use_drop:N \l_@@_cell_box }
703   }
704 }
```

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

```
705 \tl_set_rescan:Nno
706 \l_@@_letter_for_dotted_lines_str { } \l_@@_letter_for_dotted_lines_str
707 \exp_args:NV \newcolumnntype \l_@@_letter_for_dotted_lines_str
708 {
709   !
710   {
```

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

```
711 \int_compare:nNnF \c@iRow = 0
712 {
713   \int_compare:nNnF \c@iRow = \l_@@_last_row_int
714   { \skip_horizontal:N 2\l_@@_radius_dim }
715 }
```

Consider the following code:

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

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

```
716 \int_compare:nNnT \c@jCol > \g_@@_last_vdotted_col_int
717 {
718 \int_gset_eq:NN \g_@@_last_vdotted_col_int \c@jCol
719 \tl_gput_right:Nx \g_@@_internal_code_after_tl
```

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

```
720 { \@@_vdottedline:n { \int_use:N \c@jCol } }
721 }
722 }
723 }
724 \int_gzero_new:N \g_@@_last_vdotted_col_int
725 \bool_if:NT \c_@@_siunitx_loaded_bool \@@_renew_NC@rewrite@S:
726 \int_gset:Nn \g_@@_last_vdotted_col_int { -1 }
727 \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.

```
728 \tl_gclear_new:N \g_@@_Cdots_lines_tl
729 \tl_gclear_new:N \g_@@_Ldots_lines_tl
730 \tl_gclear_new:N \g_@@_Vdots_lines_tl
731 \tl_gclear_new:N \g_@@_Ddots_lines_tl
732 \tl_gclear_new:N \g_@@_Iddots_lines_tl
733 \tl_gclear_new:N \g_@@_Hdotsfor_lines_tl
734 }
```

## The environment `{NiceArrayWithDelims}`

```
735 \NewDocumentEnvironment { NiceArrayWithDelims } { m m O { } m ! O { } }
736 {
737 \tl_set:Nn \l_@@_left_delim_tl { #1 }
738 \tl_set:Nn \l_@@_right_delim_tl { #2 }
739 \bool_gset_false:N \g_@@_row_of_col_done_bool
740 \str_if_empty:NT \g_@@_name_env_str
741 { \str_gset:Nn \g_@@_name_env_str { NiceArrayWithDelims } }
742 \@@_adapt_S_column:
743 \@@_test_if_math_mode:
744 \bool_if:NT \l_@@_in_env_bool { \@@_fatal:n { Yet~in~env } }
745 \bool_set_true:N \l_@@_in_env_bool
```

We deactivate Tikz externalization because we will use PGF pictures with the options `overlay` and `remember picture` (or equivalent forms).

```
746 \cs_if_exist:NT \tikz@library@external@loaded
747 {
748 \tikzset { external / export = false }
749 \cs_if_exist:NT \ifstandalone
750 { \tikzset { external / optimize = false } }
751 }
```

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

```

752 \int_gincr:N \g_@@_env_int
753 \bool_if:NF \l_@@_block_auto_columns_width_bool
754 { \dim_gzero_new:N \g_@@_max_cell_width_dim }

```

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

```

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

```

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`.

```

756 \bool_if:NTF \l_@@_NiceArray_bool
757 { \keys_set:nn { NiceMatrix / NiceArray } }
758 { \keys_set:nn { NiceMatrix / pNiceArray } }
759 { #3 , #5 }

```

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).

```

760 \int_compare:nNnT \l_@@_last_row_int > { -2 }
761 {
762   \tl_put_right:Nn \@_update_for_first_and_last_row:
763   {
764     \dim_gset:Nn \g_@@_ht_last_row_dim
765     { \dim_max:nn \g_@@_ht_last_row_dim { \box_ht:N \l_@@_cell_box } }
766     \dim_gset:Nn \g_@@_dp_last_row_dim
767     { \dim_max:nn \g_@@_dp_last_row_dim { \box_dp:N \l_@@_cell_box } }
768   }
769 }
770 \int_compare:nNnT \l_@@_last_row_int = { -1 }
771 {
772   \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.

```

773 \str_if_empty:NTF \l_@@_name_str
774 {
775   \cs_if_exist:cT { @@_last_row_ \int_use:N \g_@@_env_int }
776   {
777     \int_set:Nn \l_@@_last_row_int
778     { \use:c { @@_last_row_ \int_use:N \g_@@_env_int } }
779   }
780 }
781 {
782   \cs_if_exist:cT { @@_last_row_ \l_@@_name_str }
783   {
784     \int_set:Nn \l_@@_last_row_int
785     { \use:c { @@_last_row_ \l_@@_name_str } }
786   }
787 }
788 }

```

A value of `-1` for the counter `\l_@@_last_col_int` means that the user has used the option `last-col` without value (it's possible in an environment without preamble like `{NiceMatrix}` or `{pNiceMatrix}`), 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).

```

789 \int_compare:nNnT \l_@@_last_col_int = { -1 }
790 {
791   \str_if_empty:NTF \l_@@_name_str
792   {
793     \cs_if_exist:cT { @@_last_col_ \int_use:N \g_@@_env_int }
794     {
795       \int_set:Nn \l_@@_last_col_int
796       { \use:c { @@_last_col_ \int_use:N \g_@@_env_int } }
797     }
798   }
799   {
800     \cs_if_exist:cT { @@_last_col_ \l_@@_name_str }
801     {

```

```

802         \int_set:Nn \l_@@_last_col_int
803         { \use:c { @@_last_col_ \l_@@_name_str } }
804     }
805 }
806 }

```

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

```

807 \@@_pre_array:

```

We compute the width of the two delimiters.

```

808 \dim_zero_new:N \l_@@_left_delim_dim
809 \dim_zero_new:N \l_@@_right_delim_dim
810 \bool_if:NTF \l_@@_NiceArray_bool
811 {
812     \dim_gset:Nn \l_@@_left_delim_dim { 2 \arraycolsep }
813     \dim_gset:Nn \l_@@_right_delim_dim { 2 \arraycolsep }
814 }
815 {

```

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

```

816 \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 #1 $ }
817 \dim_set:Nn \l_@@_left_delim_dim { \box_wd:N \l_tmpa_box }
818 \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 #2 $ }
819 \dim_set:Nn \l_@@_right_delim_dim { \box_wd:N \l_tmpa_box }
820 }

```

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

```

821 \box_clear_new:N \l_@@_the_array_box

```

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

```

822 \tl_set:Nn \l_tmpa_tl { #4 }
823 \int_compare:nNnTF \l_@@_first_col_int = 0
824 { \tl_put_left:NV \l_tmpa_tl \c_@@_preamble_first_col_tl }
825 {
826     \bool_lazy_all:nT
827     {
828         \l_@@_NiceArray_bool
829         { \bool_not_p:n \l_@@_vlines_bool }
830         { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
831     }
832     { \tl_put_left:Nn \l_tmpa_tl { @ { } } }
833 }
834 \int_compare:nNnTF \l_@@_last_col_int > { -1 }
835 { \tl_put_right:NV \l_tmpa_tl \c_@@_preamble_last_col_tl }
836 {
837     \bool_lazy_all:nT
838     {
839         \l_@@_NiceArray_bool
840         { \bool_not_p:n \l_@@_vlines_bool }
841         { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
842     }
843     { \tl_put_right:Nn \l_tmpa_tl { @ { } } }
844 }
845 \tl_put_right:Nn \l_tmpa_tl { > { \@@_error_too_much_cols: } 1 }

```

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).

```

846 \hbox_set:Nw \l_@@_the_array_box

```

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

```

847 \bool_if:NT \l_@@_vlines_bool
848 {

```

```

849     \dim_add:Nn \arraycolsep { 0.5 \arrayrulewidth }
850     \skip_horizontal:N 0.5\arrayrulewidth
851   }
852   \skip_horizontal:N \l_@@_left_margin_dim
853   \skip_horizontal:N \l_@@_extra_left_margin_dim
854   \c_math_toggle_token
855   \bool_if:NTF \l_@@_light_syntax_bool
856     { \begin { @@-light-syntax } }
857     { \begin { @@-normal-syntax } }
858 }
859 {
860   \bool_if:NTF \l_@@_light_syntax_bool
861     { \end { @@-light-syntax } }
862     { \end { @@-normal-syntax } }
863   \c_math_toggle_token
864   \skip_horizontal:N \l_@@_right_margin_dim
865   \skip_horizontal:N \l_@@_extra_right_margin_dim

```

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

```

866   \bool_if:NT \l_@@_vlines_bool { \skip_horizontal:N 0.5\arrayrulewidth }
867   \hbox_set_end:

```

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

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

```

868   \int_compare:nNnT \l_@@_last_row_int > { -2 }
869   {
870     \bool_if:NF \l_@@_last_row_without_value_bool
871     {
872       \int_compare:nNnF \l_@@_last_row_int = \c@iRow
873       {
874         \@@_error:n { Wrong~last~row }
875         \int_gset_eq:NN \l_@@_last_row_int \c@iRow
876       }
877     }
878   }

```

Now, the definition of `\c@jCol` and `\g_@@_col_total_int` change: `\c@jCol` will be the number of columns without the “last column”; `\g_@@_col_total_int` will be the number of columns with this “last column”.<sup>35</sup>

```

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

```

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

```

881   \int_gset_eq:NN \g_@@_row_total_int \c@iRow
882   \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. 54).

```

883   \int_compare:nNnT \l_@@_first_col_int = 0
884   {
885     \skip_horizontal:N \arraycolsep
886     \skip_horizontal:N \g_@@_width_first_col_dim
887   }

```

The construction of the real box is different in `{NiceArray}` and in the other environments because, in `{NiceArray}`, we have to take into account the value of `baseline` and we have no delimiter to put. We begin with `{NiceArray}`.

```

888   \bool_if:NTF \l_@@_NiceArray_bool
889   {

```

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`.

---

<sup>35</sup>We remind that the potential “first column” (exterior) has the number 0.



```

890 \str_if_eq:VnTF \l_@@_baseline_str { b }
891 {
892   \pgfpicture
893     \@@_qpoint: { row - 1 }
894     \dim_gset_eq:NN \g_tmpa_dim \pgf@y
895     \@@_qpoint: { row - \int_use:N \c@iRow - base }
896     \dim_gsub:Nn \g_tmpa_dim \pgf@y
897   \endpgfpicture
898   \int_compare:nNnT \l_@@_first_row_int = 0
899   {
900     \dim_gadd:Nn \g_tmpa_dim
901       { \g_@@_ht_row_zero_dim + \g_@@_dp_row_zero_dim }
902   }
903   \box_move_up:nn \g_tmpa_dim { \box_use_drop:N \l_@@_the_array_box }
904 }
905 {
906   \str_if_eq:VnTF \l_@@_baseline_str { c }
907   { \box_use_drop:N \l_@@_the_array_box }
908   {

```

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

```

909 \str_if_eq:VnT \l_@@_baseline_str { t }
910 { \str_set:Nn \l_@@_baseline_str { 1 } }

```

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

```

911 \int_set:Nn \l_tmpa_int \l_@@_baseline_str
912 \bool_if:nT
913 {
914   \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int
915   || \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int
916 }
917 {
918   \@@_error:n { bad-value-for-baseline }
919   \int_set:Nn \l_tmpa_int 1
920 }

```

We use a `{pgfpicture}` to extract coordinates (nothing is drawn).

```

921 \pgfpicture
922 \@@_qpoint: { row - 1 }
923 \dim_gset_eq:NN \g_tmpa_dim \pgf@y
924 \@@_qpoint: { row - \int_use:N \l_tmpa_int - base }
925 \dim_gsub:Nn \g_tmpa_dim \pgf@y
926 \endpgfpicture
927 \int_compare:nNnT \l_@@_first_row_int = 0
928 {
929   \dim_gadd:Nn \g_tmpa_dim
930     { \g_@@_ht_row_zero_dim + \g_@@_dp_row_zero_dim }
931 }
932 \box_move_up:nn \g_tmpa_dim
933 { \box_use_drop:N \l_@@_the_array_box }
934 }
935 }
936 }

```

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).

```

937 {
938   \int_compare:nNnTF \l_@@_first_row_int = 0
939   {
940     \dim_set_eq:NN \l_tmpa_dim \g_@@_dp_row_zero_dim
941     \dim_add:Nn \l_tmpa_dim \g_@@_ht_row_zero_dim
942   }
943   { \dim_zero:N \l_tmpa_dim }

```

We compute `\l_tmpb_dim` which is the total height of the “last row” below the array (when the key `last-row` is used). A value of `-2` for `\l_@@_last_row_int` means that there is no “last row”.<sup>36</sup>

```

944 \int_compare:nNnTF \l_@@_last_row_int > { -2 }
945 {
946     \dim_set_eq:NN \l_tmpb_dim \g_@@_ht_last_row_dim
947     \dim_add:Nn \l_tmpb_dim \g_@@_dp_last_row_dim
948 }
949 { \dim_zero:N \l_tmpb_dim }
950 \hbox_set:Nn \l_tmpa_box
951 {
952     \c_math_toggle_token
953     \left #1
954     \vcenter
955     {

```

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.

```

956         \skip_vertical:N -\l_tmpa_dim
957         \hbox
958         {
959             \skip_horizontal:N -\arraycolsep
960             \box_use_drop:N \l_@@_the_array_box
961             \skip_horizontal:N -\arraycolsep
962         }

```

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

```

963         \skip_vertical:N -\l_tmpb_dim
964     }
965     \right #2
966     \c_math_toggle_token
967 }

```

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.

```

968     \bool_if:NTF \l_@@_max_delimiter_width_bool
969     { \@@_put_box_in_flow_bis:nn { #1 } { #2 } }
970     \@@_put_box_in_flow:
971 }

```

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. 55).

```

972 \bool_if:NT \g_@@_last_col_found_bool
973 {
974     \skip_horizontal:N \g_@@_width_last_col_dim
975     \skip_horizontal:N \arraycolsep
976 }
977 \@@_after_array:
978 }

```

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

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

```

979 \cs_new_protected:Npn \@@_put_box_in_flow:
980 {
981     \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }
982     \box_set_dp:Nn \l_tmpa_box { \box_dp:N \l_tmpa_box + \l_tmpb_dim }
983     \box_use_drop:N \l_tmpa_box
984 }

```

---

<sup>36</sup>A value of `-1` for `\l_@@_last_row_int` means that there is a “last row” but the number of that row is unknown (the user have not set the value with the option `last row`).

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.

```

985 \cs_new_protected:Npn \@@_put_box_in_flow_bis:nn #1 #2
986 {
We will compute the real width of both delimiters used.
987   \dim_zero_new:N \l_@@_real_left_delim_dim
988   \dim_zero_new:N \l_@@_real_right_delim_dim
989   \hbox_set:Nn \l_tmpb_box
990   {
991     \c_math_toggle_token
992     \left #1
993     \vcenter
994     {
995       \vbox_to_ht:nn
996       { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
997       { }
998     }
999     \right .
1000    \c_math_toggle_token
1001  }
1002  \dim_set:Nn \l_@@_real_left_delim_dim
1003  { \box_wd:N \l_tmpb_box - \nulldelimiterspace }
1004  \hbox_set:Nn \l_tmpb_box
1005  {
1006    \c_math_toggle_token
1007    \left .
1008    \vbox_to_ht:nn
1009    { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1010    { }
1011    \right #2
1012    \c_math_toggle_token
1013  }
1014  \dim_set:Nn \l_@@_real_right_delim_dim
1015  { \box_wd:N \l_tmpb_box - \nulldelimiterspace }

```

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

```

1016   \skip_horizontal:N \l_@@_left_delim_dim
1017   \skip_horizontal:N -\l_@@_real_left_delim_dim
1018   \@@_put_box_in_flow:
1019   \skip_horizontal:N \l_@@_right_delim_dim
1020   \skip_horizontal:N -\l_@@_real_right_delim_dim
1021 }

```

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

```

1022 \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).

```

1023 {
1024   \peek_meaning_ignore_spaces:NTF \end
1025   { \@@_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`).

```

1026   { \exp_args:NV \@@_array: \l_tmpa_tl }
1027 }
1028 {
1029   \@@_create_col_nodes:
1030   \endarray

```

```
1031 }
```

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

```
1032 \NewDocumentEnvironment { @@-light-syntax } { b }
1033 {
```

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`.

```
1034 \tl_if_empty:nT { #1 } { \@@_fatal:n { empty-environment } }
1035 \tl_map_inline:nn { #1 }
1036 {
1037   \tl_if_eq:nnT { ##1 } { & }
1038   { \@@_fatal:n { ampersand-in-light-syntax } }
1039   \tl_if_eq:nnT { ##1 } { \ }
1040   { \@@_fatal:n { double-backslash-in-light-syntax } }
1041 }
```

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

```
1042 \seq_gclear_new:N \g_@@_rows_seq
1043 \tl_set_rescan:Nno \l_@@_end_of_row_tl { } \l_@@_end_of_row_tl
1044 \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.

```
1045 \int_compare:nNnT \l_@@_last_row_int = { -1 }
1046 { \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`).

```
1047 \exp_args:NV \@@_array: \l_tmpa_tl
```

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

```
1048 \seq_gpop_left:NN \g_@@_rows_seq \l_tmpa_tl
1049 \exp_args:NV \@@_line_with_light_syntax_i:n \l_tmpa_tl
1050 \seq_map_function:NN \g_@@_rows_seq \@@_line_with_light_syntax:n
1051 \@@_create_col_nodes:
1052 \endarray
1053 }
```

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`.

```
1054 { }
1055 \cs_new_protected:Npn \@@_line_with_light_syntax_i:n #1
1056 {
1057   \seq_gclear_new:N \g_@@_cells_seq
1058   \seq_gset_split:Nnn \g_@@_cells_seq { ~ } { #1 }
1059   \seq_gpop_left:NN \g_@@_cells_seq \l_tmpa_tl
1060   \l_tmpa_tl
1061   \seq_map_function:NN \g_@@_cells_seq \@@_cell_with_light_syntax:n
1062 }
1063 \cs_new_protected:Npn \@@_line_with_light_syntax:n #1
1064 {
1065   \tl_if_empty:nF { #1 }
1066   { \ \ \@@_line_with_light_syntax_i:n { #1 } }
1067 }
1068 \cs_new_protected:Npn \@@_cell_with_light_syntax:n #1 { & #1 }
```

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).

```
1069 \cs_new_protected:Npn \@@_analyze_end:Nn #1 #2
1070 {
1071   \str_if_eq:VnT \g_@@_name_env_str { #2 }
1072   { \@@_fatal:n { empty-environment } }
```

We repute in the stream the `\end{...}` we have extracted and the user will have an error for incorrect nested environments.

```
1073   \end { #2 }
1074 }
```

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).

```
1075 \cs_new:Npn \@@_create_col_nodes:
1076 {
1077   \crrc
1078   \int_compare:nNt \c@iRow = 0 { \@@_fatal:n { Zero~row } }
1079   \int_compare:nNt \l_@@_first_col_int = 0 { \omit & }
1080   \omit
```

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

```
1081   \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`).

```
1082   \pgfpicture
1083   \pgfrememberpicturepositiononpagetrue
1084   \pgfcoordinate { \@@_env: - col - 1 } \pgfpintorigin
1085   \str_if_empty:NF \l_@@_name_str
1086     { \pgfnodealias { \@@_env: - col - 1 } { \l_@@_name_str - col - 1 } }
1087   \endpgfpicture
```

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 erased by a fixed value in the concerned cases.

```
1088   \skip_gset:Nn \g_tmpa_skip { 0 pt+plus 1 fill }
1089   \bool_if:NF \l_@@_auto_columns_width_bool
1090     { \dim_compare:nNt \l_@@_columns_width_dim > \c_zero_dim }
1091     {
1092       \bool_lazy_and:nnTF
1093         \l_@@_auto_columns_width_bool
1094         { \bool_not_p:n \l_@@_block_auto_columns_width_bool }
1095         { \skip_gset_eq:NN \g_tmpa_skip \g_@@_max_cell_width_dim }
1096         { \skip_gset_eq:NN \g_tmpa_skip \l_@@_columns_width_dim }
1097       \skip_gadd:Nn \g_tmpa_skip { 2 \arraycolsep }
1098     }
1099   \skip_horizontal:N \g_tmpa_skip
1100   \hbox
1101   {
1102     \pgfpicture
1103     \pgfrememberpicturepositiononpagetrue
1104     \pgfcoordinate { \@@_env: - col - 2 } \pgfpintorigin
1105     \str_if_empty:NF \l_@@_name_str
1106       { \pgfnodealias { \@@_env: - col - 2 } { \l_@@_name_str - col - 2 } }
1107     \endpgfpicture
1108   }
```

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.

```
1109   \int_gset:Nn \g_tmpa_int 1
1110   \bool_if:NtF \g_@@_last_col_found_bool
1111     { \prg_replicate:nn { \g_@@_col_total_int - 2 } }
1112     { \prg_replicate:nn { \g_@@_col_total_int - 1 } }
1113     {
1114       &
1115       \omit
```

The incrementation of the counter `\g_tmpa_int` must be done after the `\omit` of the cell.

```
1116       \int_gincr:N \g_tmpa_int
1117       \skip_horizontal:N \g_tmpa_skip
```

We create the “col” node on the right of the current column.

```

1118     \pgfpicture
1119     \pgfrememberpicturepositiononpagetrue
1120     \pgfcoordinate { \l_@@_env: - col - \l_@@_succ:n \g_tmpa_int }
1121     \pgfpointorigin
1122     \str_if_empty:NF \l_@@_name_str
1123     {
1124         \pgfnodealias
1125         { \l_@@_env: - col - \l_@@_succ:n \g_tmpa_int }
1126         { \l_@@_name_str - col - \l_@@_succ:n \g_tmpa_int }
1127     }
1128     \endpgfpicture
1129 }
1130 \cr
1131 }

```

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

```

1132 \tl_const:Nn \c_@@_preamble_first_col_tl
1133 {
1134     >
1135     {
1136         \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.

```

1137     \hbox_set:Nw \l_@@_cell_box
1138     \c_math_toggle_token
1139     \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”.

```

1140     \bool_lazy_and:nnT
1141     { \int_compare_p:nNn \c@iRow > 0 }
1142     {
1143         \bool_lazy_or_p:nn
1144         { \int_compare_p:nNn \l_@@_last_row_int < 0 }
1145         { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
1146     }
1147     {
1148         \l_@@_code_for_first_col_tl
1149         \xglobal \colorlet { nicematrix-first-col } { . }
1150     }
1151 }

```

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`.

```

1152     l
1153     <
1154     {
1155         \c_math_toggle_token
1156         \hbox_set_end:
1157         \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.

```

1158     \dim_gset:Nn \g_@@_width_first_col_dim
1159     { \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.

```

1160     \hbox_overlap_left:n
1161     {
1162         \dim_compare:nNnTF { \box_wd:N \l_@@_cell_box } > \c_zero_dim
1163         \l_@@_node_for_the_cell:
1164         { \box_use_drop:N \l_@@_cell_box }
1165         \skip_horizontal:N \l_@@_left_delim_dim
1166         \skip_horizontal:N \l_@@_left_margin_dim
1167         \skip_horizontal:N \l_@@_extra_left_margin_dim

```

```

1168     }
1169     \skip_horizontal:N -2\arraycolsep
1170   }
1171 }

```

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

```

1172 \tl_const:Nn \c_@@_preamble_last_col_tl
1173 {
1174   >
1175   {

```

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

```

1176     \bool_gset_true:N \g_@@_last_col_found_bool
1177     \int_gincr:N \c@jCol
1178     \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.

```

1179     \hbox_set:Nw \l_@@_cell_box
1180     \c_math_toggle_token
1181     \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”.

```

1182     \int_compare:nNnT \c@iRow > 0
1183     {
1184       \bool_lazy_or:nnT
1185       { \int_compare_p:nNn \l_@@_last_row_int < 0 }
1186       { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
1187       {
1188         \l_@@_code_for_last_col_tl
1189         \xglobal \colorlet { nicematrix-last-col } { . }
1190       }
1191     }
1192   }
1193   l
1194   <
1195   {
1196     \c_math_toggle_token
1197     \hbox_set_end:
1198     \@@_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.

```

1199     \dim_gset:Nn \g_@@_width_last_col_dim
1200     { \dim_max:nn \g_@@_width_last_col_dim { \box_wd:N \l_@@_cell_box } }
1201     \skip_horizontal:N -2\arraycolsep

```

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

```

1202     \hbox_overlap_right:n
1203     {
1204       \dim_compare:nNnT { \box_wd:N \l_@@_cell_box } > \c_zero_dim
1205       {
1206         \skip_horizontal:N \l_@@_right_delim_dim
1207         \skip_horizontal:N \l_@@_right_margin_dim
1208         \skip_horizontal:N \l_@@_extra_right_margin_dim
1209         \@@_node_for_the_cell:
1210       }
1211     }
1212   }
1213 }

```

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.

```

1214 \NewDocumentEnvironment { NiceArray } { }
1215 {
1216   \bool_set_true:N \l_@@_NiceArray_bool

```

```

1217 \str_if_empty:NT \g_@@_name_env_str
1218 { \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).

```

1219 \NiceArrayWithDelims . .
1220 }
1221 { \endNiceArrayWithDelims }

```

We create the variants of the environment {NiceArrayWithDelims}.

```

1222 \NewDocumentEnvironment { pNiceArray } { }
1223 {
1224   \str_if_empty:NT \g_@@_name_env_str
1225   { \str_gset:Nn \g_@@_name_env_str { pNiceArray } }
1226   \@@_test_if_math_mode:
1227   \NiceArrayWithDelims ( )
1228 }
1229 { \endNiceArrayWithDelims }

1230 \NewDocumentEnvironment { bNiceArray } { }
1231 {
1232   \str_if_empty:NT \g_@@_name_env_str
1233   { \str_gset:Nn \g_@@_name_env_str { bNiceArray } }
1234   \@@_test_if_math_mode:
1235   \NiceArrayWithDelims [ ]
1236 }
1237 { \endNiceArrayWithDelims }

1238 \NewDocumentEnvironment { BNiceArray } { }
1239 {
1240   \str_if_empty:NT \g_@@_name_env_str
1241   { \str_gset:Nn \g_@@_name_env_str { BNiceArray } }
1242   \@@_test_if_math_mode:
1243   \NiceArrayWithDelims \{ \}
1244 }
1245 { \endNiceArrayWithDelims }

1246 \NewDocumentEnvironment { vNiceArray } { }
1247 {
1248   \str_if_empty:NT \g_@@_name_env_str
1249   { \str_gset:Nn \g_@@_name_env_str { vNiceArray } }
1250   \@@_test_if_math_mode:
1251   \NiceArrayWithDelims | |
1252 }
1253 { \endNiceArrayWithDelims }

1254 \NewDocumentEnvironment { VNiceArray } { }
1255 {
1256   \str_if_empty:NT \g_@@_name_env_str
1257   { \str_gset:Nn \g_@@_name_env_str { VNiceArray } }
1258   \@@_test_if_math_mode:
1259   \NiceArrayWithDelims \| \|
1260 }
1261 { \endNiceArrayWithDelims }

```

## The environment {NiceMatrix} and its variants

```

1262 \cs_new_protected:Npn \@@_define_env:n #1
1263 {
1264   \NewDocumentEnvironment { #1 NiceMatrix } { ! 0 { } }
1265   {
1266     \str_gset:Nn \g_@@_name_env_str { #1 NiceMatrix }
1267     \tl_set:Nn \l_@@_type_of_col_tl C
1268     \keys_set:nn { NiceMatrix / NiceMatrix } { ##1 }
1269     \exp_args:Nnx \@@_begin_of_NiceMatrix:nn { #1 } \l_@@_type_of_col_tl

```



```

1270     }
1271     { \end { #1 NiceArray } }
1272 }
1273 \cs_new_protected:Npn \@@_begin_of_NiceMatrix:nn #1 #2
1274 {
1275     \begin { #1 NiceArray }
1276     {
1277         *
1278         {
1279             \int_compare:nNnTF \l_@@_last_col_int < 0
1280                 \c@MaxMatrixCols
1281                 { \@@_pred:n \l_@@_last_col_int }
1282             }
1283         #2
1284     }
1285 }
1286 \@@_define_env:n { }
1287 \@@_define_env:n p
1288 \@@_define_env:n b
1289 \@@_define_env:n B
1290 \@@_define_env:n v
1291 \@@_define_env:n V

```

## After the construction of the array

```

1292 \cs_new_protected:Npn \@@_after_array:
1293 {
1294     \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.

```

1295     \bool_if:NT \g_@@_last_col_found_bool
1296     { \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`.

```

1297     \bool_if:NT \l_@@_last_col_without_value_bool
1298     {
1299         \dim_set_eq:NN \l_@@_last_col_int \g_@@_col_total_int
1300         \iow_shipout:Nn \@mainaux \ExplSyntaxOn
1301         \iow_shipout:Nx \@mainaux
1302         {
1303             \cs_gset:cpn { @@_last_col_ \int_use:N \g_@@_env_int }
1304             { \int_use:N \g_@@_col_total_int }
1305         }
1306         \str_if_empty:NF \l_@@_name_str
1307         {
1308             \iow_shipout:Nx \@mainaux
1309             {
1310                 \cs_gset:cpn { @@_last_col_ \l_@@_name_str }
1311                 { \int_use:N \g_@@_col_total_int }
1312             }
1313         }
1314         \iow_shipout:Nn \@mainaux \ExplSyntaxOff
1315     }

```

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.

```

1316     \bool_if:NT \l_@@_last_row_without_value_bool
1317     {
1318         \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.

```

1319     \bool_if:NF \l_@@_light_syntax_bool
1320     {
1321         \iow_shipout:Nn \@mainaux \ExplSyntaxOn
1322         \iow_shipout:Nx \@mainaux
1323         {
1324             \cs_gset:cpn { @@_last_row_ \int_use:N \g_@@_env_int }
1325             { \int_use:N \g_@@_row_total_int }
1326         }

```

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.

```

1327         \str_if_empty:NF \l_@@_name_str
1328         {
1329             \iow_shipout:Nx \@mainaux
1330             {
1331                 \cs_gset:cpn { @@_last_row_ \l_@@_name_str }
1332                 { \int_use:N \g_@@_row_total_int }
1333             }
1334         }
1335     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
1336 }
1337 }

```

By default, the diagonal lines will be parallelized<sup>37</sup>. There are two types of diagonals lines: the `\Ddots` diagonals and the `\Iddots` diagonals. We have to count both types in order to know whether a diagonal is the first of its type in the current `{NiceArray}` environment.

```

1338     \bool_if:NT \l_@@_parallelize_diags_bool
1339     {
1340         \int_gzero_new:N \g_@@_ddots_int
1341         \int_gzero_new:N \g_@@_iddots_int

```

The dimensions `\g_@@_delta_x_one_dim` and `\g_@@_delta_y_one_dim` will contain the  $\Delta_x$  and  $\Delta_y$  of the first `\Ddots` diagonal. We have to store these values in order to draw the others `\Ddots` diagonals parallel to the first one. Similarly `\g_@@_delta_x_two_dim` and `\g_@@_delta_y_two_dim` are the  $\Delta_x$  and  $\Delta_y$  of the first `\Iddots` diagonal.

```

1342         \dim_gzero_new:N \g_@@_delta_x_one_dim
1343         \dim_gzero_new:N \g_@@_delta_y_one_dim
1344         \dim_gzero_new:N \g_@@_delta_x_two_dim
1345         \dim_gzero_new:N \g_@@_delta_y_two_dim
1346     }
1347     \bool_if:nTF \l_@@_medium_nodes_bool
1348     {
1349         \bool_if:NTF \l_@@_large_nodes_bool
1350         \@@_create_medium_and_large_nodes:
1351         \@@_create_medium_nodes:
1352     }
1353     { \bool_if:NT \l_@@_large_nodes_bool \@@_create_large_nodes: }
1354     \int_zero_new:N \l_@@_initial_i_int
1355     \int_zero_new:N \l_@@_initial_j_int
1356     \int_zero_new:N \l_@@_final_i_int
1357     \int_zero_new:N \l_@@_final_j_int
1358     \bool_set_false:N \l_@@_initial_open_bool
1359     \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.

```

1360     \bool_if:NT \l_@@_small_bool
1361     {
1362         \dim_set:Nn \l_@@_radius_dim { 0.37 pt }
1363         \dim_set:Nn \l_@@_inter_dots_dim { 0.25 em }

```

<sup>37</sup>It's possible to use the option `parallelize-diags` to disable this parallelization.

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.

```

1364     \dim_set:Nn \l_@@_xdots_shorten_dim { 0.6 \l_@@_xdots_shorten_dim }
1365 }

```

Now, we really draw the lines.

```

1366     \@@_draw_dotted_lines:

```

We draw the vertical rules of the option `vlines` before the `internal-code-after` because the option `white` of a `\Block` may have to erase these vertical rules.

```

1367     \bool_if:NT \l_@@_vlines_bool \@@_draw_vlines:
1368     \g_@@_internal_code_after_tl
1369     \tl_gclear:N \g_@@_internal_code_after_tl
1370     \bool_if:NT \c_@@_tikz_loaded_bool
1371     {
1372         \tikzset
1373         {
1374             every~picture / .style =
1375             {
1376                 overlay ,
1377                 remember~picture ,
1378                 name~prefix = \@@_env: -
1379             }
1380         }
1381     }
1382     \cs_set_eq:NN \line \@@_line
1383     \g_@@_code_after_tl
1384     \tl_gclear:N \g_@@_code_after_tl
1385     \group_end:
1386     \str_gclear:N \g_@@_name_env_str
1387     \@@_restore_iRow_jCol:
1388 }

```

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).

```

1389 \AtBeginDocument
1390 {
1391     \cs_new_protected:Npx \@@_draw_dotted_lines:
1392     {
1393         \c_@@_pgfortikzpicture_tl
1394         \@@_draw_dotted_lines_i:
1395         \c_@@_endpgfortikzpicture_tl
1396     }
1397 }

```

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

```

1398 \cs_new_protected:Npn \@@_draw_dotted_lines_i:
1399 {
1400     \pgfrememberpicturepositiononpagetrue
1401     \pgf@relevantforpicturesizefalse
1402     \g_@@_Hdotsfor_lines_tl
1403     \g_@@_Vdots_lines_tl
1404     \g_@@_Ddots_lines_tl
1405     \g_@@_Iddots_lines_tl
1406     \g_@@_Cdots_lines_tl
1407     \g_@@_Ldots_lines_tl
1408 }

1409 \cs_new_protected:Npn \@@_restore_iRow_jCol:
1410 {
1411     \cs_if_exist:NT \theiRow { \int_gset_eq:NN \c@iRow \l_@@_save_iRow_int }
1412     \cs_if_exist:NT \thejCol { \int_gset_eq:NN \c@jCol \l_@@_save_jCol_int }

```

1413 }

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.

```
1414 \cs_new_protected:Npn \@@_find_extremities_of_line:nnnn #1 #2 #3 #4
1415 {
```

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

```
1416 \cs_set:cpn { @@ _ dotted _ #1 - #2 } { }
```

Initialization of variables.

```
1417 \int_set:Nn \l_@@_initial_i_int { #1 }
1418 \int_set:Nn \l_@@_initial_j_int { #2 }
1419 \int_set:Nn \l_@@_final_i_int { #1 }
1420 \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.

```
1421 \bool_set_false:N \l_@@_stop_loop_bool
1422 \bool_do_until:Nn \l_@@_stop_loop_bool
1423 {
1424   \int_add:Nn \l_@@_final_i_int { #3 }
1425   \int_add:Nn \l_@@_final_j_int { #4 }
```

We test if we are still in the matrix.

```
1426 \bool_set_false:N \l_@@_final_open_bool
1427 \int_compare:nNnTF \l_@@_final_i_int > \c@iRow
1428 {
1429   \int_compare:nNnT { #3 } = 1
1430   { \bool_set_true:N \l_@@_final_open_bool }
1431 }
1432 {
1433   \int_compare:nNnTF \l_@@_final_j_int < 1
1434   {
1435     \int_compare:nNnT { #4 } = { -1 }
1436     { \bool_set_true:N \l_@@_final_open_bool }
1437   }
1438   {
1439     \int_compare:nNnT \l_@@_final_j_int > \c@jCol
1440     {
1441       \int_compare:nNnT { #4 } = 1
1442       { \bool_set_true:N \l_@@_final_open_bool }
1443     }
1444   }
1445 }
1446 \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.

```

1447     {
We do a step backwards.
1448         \int_sub:Nn \l_@@_final_i_int { #3 }
1449         \int_sub:Nn \l_@@_final_j_int { #4 }
1450         \bool_set_true:N \l_@@_stop_loop_bool
1451     }

```

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`.

```

1452     {
1453         \cs_if_exist:cTF
1454         {
1455             @@ _ dotted _
1456             \int_use:N \l_@@_final_i_int -
1457             \int_use:N \l_@@_final_j_int
1458         }
1459         {
1460             \int_sub:Nn \l_@@_final_i_int { #3 }
1461             \int_sub:Nn \l_@@_final_j_int { #4 }
1462             \bool_set_true:N \l_@@_final_open_bool
1463             \bool_set_true:N \l_@@_stop_loop_bool
1464         }
1465         {
1466             \cs_if_exist:cTF
1467             {
1468                 pgf @ sh @ ns @ \@@_env:
1469                 - \int_use:N \l_@@_final_i_int
1470                 - \int_use:N \l_@@_final_j_int
1471             }
1472             { \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 mark 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.

```

1473         {
1474             \cs_set:cpn
1475             {
1476                 @@ _ dotted _
1477                 \int_use:N \l_@@_final_i_int -
1478                 \int_use:N \l_@@_final_j_int
1479             }
1480             { }
1481         }
1482     }
1483 }
1484

```

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

```

1485     \bool_set_false:N \l_@@_stop_loop_bool
1486     \bool_do_until:Nn \l_@@_stop_loop_bool
1487     {
1488         \int_sub:Nn \l_@@_initial_i_int { #3 }
1489         \int_sub:Nn \l_@@_initial_j_int { #4 }
1490         \bool_set_false:N \l_@@_initial_open_bool
1491         \int_compare:nNnTF \l_@@_initial_i_int < 1
1492         {
1493             \int_compare:nNnT { #3 } = 1
1494             { \bool_set_true:N \l_@@_initial_open_bool }
1495         }
1496     }

```

```

1497 \int_compare:nNnTF \l_@@_initial_j_int < 1
1498 {
1499     \int_compare:nNnT { #4 } = 1
1500     { \bool_set_true:N \l_@@_initial_open_bool }
1501 }
1502 {
1503     \int_compare:nNnT \l_@@_initial_j_int > \c@jCol
1504     {
1505         \int_compare:nNnT { #4 } = { -1 }
1506         { \bool_set_true:N \l_@@_initial_open_bool }
1507     }
1508 }
1509 }
1510 \bool_if:NTF \l_@@_initial_open_bool
1511 {
1512     \int_add:Nn \l_@@_initial_i_int { #3 }
1513     \int_add:Nn \l_@@_initial_j_int { #4 }
1514     \bool_set_true:N \l_@@_stop_loop_bool
1515 }
1516 {
1517     \cs_if_exist:cTF
1518     {
1519         @@ _ dotted _
1520         \int_use:N \l_@@_initial_i_int -
1521         \int_use:N \l_@@_initial_j_int
1522     }
1523     {
1524         \int_add:Nn \l_@@_initial_i_int { #3 }
1525         \int_add:Nn \l_@@_initial_j_int { #4 }
1526         \bool_set_true:N \l_@@_initial_open_bool
1527         \bool_set_true:N \l_@@_stop_loop_bool
1528     }
1529     {
1530         \cs_if_exist:cTF
1531         {
1532             pgf @ sh @ ns @ \@@_env:
1533             - \int_use:N \l_@@_initial_i_int
1534             - \int_use:N \l_@@_initial_j_int
1535         }
1536         { \bool_set_true:N \l_@@_stop_loop_bool }
1537         {
1538             \cs_set:cpn
1539             {
1540                 @@ _ dotted _
1541                 \int_use:N \l_@@_initial_i_int -
1542                 \int_use:N \l_@@_initial_j_int
1543             }
1544             { }
1545         }
1546     }
1547 }
1548 }
1549 }
1550 \cs_new:Nn \@@_initial_cell:
1551 { \@@_env: - \int_use:N \l_@@_initial_i_int - \int_use:N \l_@@_initial_j_int }
1552 \cs_new:Nn \@@_final_cell:
1553 { \@@_env: - \int_use:N \l_@@_final_i_int - \int_use:N \l_@@_final_j_int }
1554 \cs_new_protected:Npn \@@_set_initial_coords:
1555 {
1556     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
1557     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
1558 }
1559 \cs_new_protected:Npn \@@_set_final_coords:

```

```

1560 {
1561   \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
1562   \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
1563 }
1564 \cs_new_protected:Npn \@@_set_initial_coords_from_anchor:n #1
1565 {
1566   \pgfpointanchor \@@_initial_cell: { #1 }
1567   \@@_set_initial_coords:
1568 }
1569 \cs_new_protected:Npn \@@_set_final_coords_from_anchor:n #1
1570 {
1571   \pgfpointanchor \@@_final_cell: { #1 }
1572   \@@_set_final_coords:
1573 }

```

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.

```

1574 \cs_new_protected:Npn \@@_draw_Ldots:nnn #1 #2 #3
1575 {
1576   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1577   {
1578     \@@_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.

```

1579   \group_begin:
1580   \int_compare:nNnTF { #1 } = 0
1581   { \color { nicematrix-first-row } }
1582   {

```

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.

```

1583     \int_compare:nNnT { #1 } = \l_@@_last_row_int
1584     { \color { nicematrix-last-row } }
1585   }
1586   \keys_set:nn { NiceMatrix / xdots } { #3 }
1587   \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
1588   \@@_actually_draw_Ldots:
1589   \group_end:
1590 }
1591 }

```

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`.

```

1592 \cs_new_protected:Npn \@@_actually_draw_Ldots:
1593 {
1594   \bool_if:NTF \l_@@_initial_open_bool
1595   {
1596     \@@_qpoint: { col - \int_use:N \l_@@_initial_j_int }
1597     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
1598     \dim_add:Nn \l_@@_x_initial_dim \arraycolsep
1599     \@@_qpoint: { row - \int_use:N \l_@@_initial_i_int - base }
1600     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
1601   }
1602   { \@@_set_initial_coords_from_anchor:n { base-east } }

```

```

1603 \bool_if:NTF \l_@@_final_open_bool
1604 {
1605   \@@_qpoint: { col - \@@_succ:n \l_@@_final_j_int }
1606   \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
1607   \dim_sub:Nn \l_@@_x_final_dim \arraycolsep
1608   \@@_qpoint: { row - \int_use:N \l_@@_final_i_int - base }
1609   \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
1610 }
1611 { \@@_set_final_coords_from_anchor:n { base~west } }

```

We raise the line of a quantity equal to the radius of the dots because we want the dots really “on” the line of texte.

```

1612 \dim_add:Nn \l_@@_y_initial_dim \l_@@_radius_dim
1613 \dim_add:Nn \l_@@_y_final_dim \l_@@_radius_dim
1614 \@@_draw_line:

```

The values of `\l_@@_x_initial_dim`, `\l_@@_y_initial_dim`, `\l_@@_x_final_dim`, `\l_@@_y_final_dim`, `\l_@@_initial_open_bool` and `\l_@@_final_open_bool` are still available after the `\@@_draw_line:`.

```

1615 }

```

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.

```

1616 \cs_new_protected:Npn \@@_draw_Cdots:nnn #1 #2 #3
1617 {
1618   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1619   {
1620     \@@_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.

```

1621 \group_begin:
1622   \int_compare:nNnTF { #1 } = 0
1623     { \color { nicematrix-first-row } }
1624     {

```

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.

```

1625       \int_compare:nNnT { #1 } = \l_@@_last_row_int
1626       { \color { nicematrix-last-row } }
1627     }
1628     \keys_set:nn { NiceMatrix / xdots } { #3 }
1629     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
1630     \@@_actually_draw_Cdots:
1631   \group_end:
1632 }
1633 }

```

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`.

```

1634 \cs_new_protected:Npn \@@_actually_draw_Cdots:
1635 {
1636   \bool_if:NTF \l_@@_initial_open_bool
1637   {
1638     \@@_qpoint: { col - \int_use:N \l_@@_initial_j_int }
1639     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
1640     \dim_add:Nn \l_@@_x_initial_dim \arraycolsep
1641   }

```



```

1642     { \@@_set_initial_coords_from_anchor:n { mid-east } }
1643 \bool_if:NTF \l_@@_final_open_bool
1644 {
1645     \@@_qpoint: { col - \@@_succ:n \l_@@_final_j_int }
1646     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
1647     \dim_sub:Nn \l_@@_x_final_dim \arraycolsep
1648 }
1649 { \@@_set_final_coords_from_anchor:n { mid-west } }
1650 \bool_lazy_and:nnTF
1651   \l_@@_initial_open_bool
1652   \l_@@_final_open_bool
1653 {
1654     \@@_qpoint: { row - \int_use:N \l_@@_initial_i_int }
1655     \dim_set_eq:NN \l_tmpa_dim \pgf@y
1656     \@@_qpoint: { row - \@@_succ:n \l_@@_initial_i_int }
1657     \dim_set:Nn \l_@@_y_initial_dim { ( \l_tmpa_dim + \pgf@y ) / 2 }
1658     \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim
1659 }
1660 {
1661     \bool_if:NT \l_@@_initial_open_bool
1662     { \dim_set_eq:NN \l_@@_y_initial_dim \l_@@_y_final_dim }
1663     \bool_if:NT \l_@@_final_open_bool
1664     { \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim }
1665 }
1666 \@@_draw_line:

```

The values of `\l_@@_x_initial_dim`, `\l_@@_y_initial_dim`, `\l_@@_x_final_dim`, `\l_@@_y_final_dim`, `\l_@@_initial_open_bool` and `\l_@@_final_open_bool` are still available after the `\@@_draw_line:`.

```

1667 }

```

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.

```

1668 \cs_new_protected:Npn \@@_draw_Vdots:nnn #1 #2 #3
1669 {
1670     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
1671     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1672     {
1673         \@@_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.

```

1674     \group_begin:
1675     \int_compare:nNnTF { #2 } = 0
1676     { \color { nicematrix-first-col } }
1677     {
1678         \int_compare:nNnT { #2 } = \l_@@_last_col_int
1679         { \color { nicematrix-last-col } }
1680     }
1681     \keys_set:nn { NiceMatrix / xdots } { #3 }
1682     \@@_actually_draw_Vdots:
1683     \group_end:
1684 }
1685 }

```

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`.

```

1686 \cs_new_protected:Npn \@@_actually_draw_Vdots:
1687 {

```

The boolean `\l_tmpa_bool` indicates whether the column is of type l (L of `{NiceArray}`) or may be considered as if.

```

1688 \bool_set_false:N \l_tmpa_bool
1689 \bool_lazy_or:nnF \l_@@_initial_open_bool \l_@@_final_open_bool
1690 {
1691   \@@_set_initial_coords_from_anchor:n { south~west }
1692   \@@_set_final_coords_from_anchor:n { north~west }
1693   \bool_set:Nn \l_tmpa_bool
1694     { \dim_compare_p:nNn \l_@@_x_initial_dim = \l_@@_x_final_dim }
1695 }

```

Now, we try to determine whether the column is of type c (C of `{NiceArray}`) or may be considered as if.

```

1696 \bool_if:NTF \l_@@_initial_open_bool
1697 {
1698   \@@_qpoint: { row - 1 }
1699   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
1700 }
1701 { \@@_set_initial_coords_from_anchor:n { south } }
1702 \bool_if:NTF \l_@@_final_open_bool
1703 {
1704   \@@_qpoint: { row - \@@_succ:n \c@iRow }
1705   \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
1706 }
1707 { \@@_set_final_coords_from_anchor:n { north } }
1708 \bool_if:NTF \l_@@_initial_open_bool
1709 {
1710   \bool_if:NTF \l_@@_final_open_bool
1711   {
1712     \@@_qpoint: { col - \int_use:N \l_@@_initial_j_int }
1713     \dim_set_eq:NN \l_tmpa_dim \pgf@x
1714     \@@_qpoint: { col - \@@_succ:n \l_@@_initial_j_int }
1715     \dim_set:Nn \l_@@_x_initial_dim { ( \pgf@x + \l_tmpa_dim ) / 2 }
1716     \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim
1717   }
1718   { \dim_set_eq:NN \l_@@_x_initial_dim \l_@@_x_final_dim }
1719 }
1720 {
1721   \bool_if:NTF \l_@@_final_open_bool
1722   { \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim }
1723 }

```

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.

```

1724 \dim_compare:nNnF \l_@@_x_initial_dim = \l_@@_x_final_dim
1725 {
1726   \dim_set:Nn \l_@@_x_initial_dim
1727   {
1728     \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
1729       \l_@@_x_initial_dim \l_@@_x_final_dim
1730   }
1731   \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim
1732 }
1733 }
1734 }
1735 \@@_draw_line:

```

The values of `\l_@@_x_initial_dim`, `\l_@@_y_initial_dim`, `\l_@@_x_final_dim`, `\l_@@_y_final_dim`, `\l_@@_initial_open_bool` and `\l_@@_final_open_bool` are still available after the `\@@_draw_line:`.

```

1736 }

```

For the diagonal lines, the situation is a bit more complicated because, by default, we parallelize the diagonals lines. The first diagonal line is drawn and then, all the other diagonal lines are drawn parallel to the first one.

The first and the second arguments are the coordinates of the cell where the command has been issued. The third argument is the list of the options.

```

1737 \cs_new_protected:Npn \@@_draw_Ddots:nnn #1 #2 #3
1738 {
1739   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1740   {
1741     \@@_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.

```

1742     \group_begin:
1743       \keys_set:nn { NiceMatrix / xdots } { #3 }
1744       \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
1745       \@@_actually_draw_Ddots:
1746     \group_end:
1747   }
1748 }

```

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`.

```

1749 \cs_new_protected:Npn \@@_actually_draw_Ddots:
1750 {
1751   \bool_if:NTF \l_@@_initial_open_bool
1752   {
1753     \@@_qpoint: { row - \int_use:N \l_@@_initial_i_int }
1754     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
1755     \@@_qpoint: { col - \int_use:N \l_@@_initial_j_int }
1756     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
1757   }
1758   { \@@_set_initial_coords_from_anchor:n { south-east } }
1759   \bool_if:NTF \l_@@_final_open_bool
1760   {
1761     \@@_qpoint: { row - \@@_succ:n \l_@@_final_i_int }
1762     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
1763     \@@_qpoint: { col - \@@_succ:n \l_@@_final_j_int }
1764     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
1765   }
1766   { \@@_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.

```

1767   \bool_if:NT \l_@@_parallelize_diags_bool
1768   {
1769     \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).

```

1770     \int_compare:nNnTF \g_@@_ddots_int = 1

```

If the diagonal line is the first one, we have no adjustment of the line to do but we store the  $\Delta_x$  and the  $\Delta_y$  of the line because these values will be used to draw the others diagonal lines parallels to the first one.

```

1771     {
1772       \dim_gset:Nn \g_@@_delta_x_one_dim
1773       { \l_@@_x_final_dim - \l_@@_x_initial_dim }
1774       \dim_gset:Nn \g_@@_delta_y_one_dim
1775       { \l_@@_y_final_dim - \l_@@_y_initial_dim }
1776     }

```

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`.

```

1777     {
1778         \dim_set:Nn \l_@@_y_final_dim
1779         {
1780             \l_@@_y_initial_dim +
1781             ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
1782             \dim_ratio:nn \g_@@_delta_y_one_dim \g_@@_delta_x_one_dim
1783         }
1784     }
1785 }
1786 \@@_draw_line:

```

The values of `\l_@@_x_initial_dim`, `\l_@@_y_initial_dim`, `\l_@@_x_final_dim`, `\l_@@_y_final_dim`, `\l_@@_initial_open_bool` and `\l_@@_final_open_bool` are still available after the `\@@_draw_line:`.

```

1787 }

```

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.

```

1788 \cs_new_protected:Npn \@@_draw_Iddots:nnn #1 #2 #3
1789 {
1790     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1791     {
1792         \@@_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.

```

1793     \group_begin:
1794     \keys_set:nn { NiceMatrix / xdots } { #3 }
1795     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
1796     \@@_actually_draw_Iddots:
1797     \group_end:
1798 }
1799 }

```

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`.

```

1800 \cs_new_protected:Npn \@@_actually_draw_Iddots:
1801 {
1802     \bool_if:NTF \l_@@_initial_open_bool
1803     {
1804         \@@_qpoint: { row - \int_use:N \l_@@_initial_i_int }
1805         \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
1806         \@@_qpoint: { col - \@@_succ:n \l_@@_initial_j_int }
1807         \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
1808     }
1809     { \@@_set_initial_coords_from_anchor:n { south-west } }
1810     \bool_if:NTF \l_@@_final_open_bool
1811     {
1812         \@@_qpoint: { row - \@@_succ:n \l_@@_final_i_int }
1813         \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
1814         \@@_qpoint: { col - \int_use:N \l_@@_final_j_int }
1815         \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
1816     }
1817     { \@@_set_final_coords_from_anchor:n { north-east } }

```

```

1818 \bool_if:NT \l_@@_parallelize_diags_bool
1819 {
1820   \int_gincr:N \g_@@_iddots_int
1821   \int_compare:nNnTF \g_@@_iddots_int = 1
1822   {
1823     \dim_gset:Nn \g_@@_delta_x_two_dim
1824     { \l_@@_x_final_dim - \l_@@_x_initial_dim }
1825     \dim_gset:Nn \g_@@_delta_y_two_dim
1826     { \l_@@_y_final_dim - \l_@@_y_initial_dim }
1827   }
1828   {
1829     \dim_set:Nn \l_@@_y_final_dim
1830     {
1831       \l_@@_y_initial_dim +
1832       ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
1833       \dim_ratio:nn \g_@@_delta_y_two_dim \g_@@_delta_x_two_dim
1834     }
1835   }
1836 }
1837 \@@_draw_line:

```

The values of `\l_@@_x_initial_dim`, `\l_@@_y_initial_dim`, `\l_@@_x_final_dim`, `\l_@@_y_final_dim`, `\l_@@_initial_open_bool` and `\l_@@_final_open_bool` are still available after the `\@@_draw_line:`.

```

1838 }

```

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).

```

1839 \NewExpandableDocumentCommand \NiceMatrixLastEnv { }
1840 { \int_use:N \g_@@_env_int }

```

## 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`

```

1841 \cs_new_protected:Npn \@@_draw_line:
1842 {
1843   \pgfrememberpicturepositiononpagetrue
1844   \pgf@relevantforpicturesizefalse
1845   \tl_if_eq:NNTF \l_@@_xdots_line_style_tl \c_@@_standard_tl
1846     \@@_draw_standard_dotted_line:
1847     \@@_draw_non_standard_dotted_line:
1848 }

```

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.

```

1849 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:
1850 {
1851   \begin { scope }
1852   \exp_args:No \@@_draw_non_standard_dotted_line:n
1853     { \l_@@_xdots_line_style_tl , \l_@@_xdots_color_tl }
1854 }

```

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.

```

1855 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:n #1
1856 {
1857   \draw
1858   [
1859     #1 ,
1860     shorten~> = \l_@@_xdots_shorten_dim ,
1861     shorten~< = \l_@@_xdots_shorten_dim ,
1862   ]
1863     ( \l_@@_x_initial_dim , \l_@@_y_initial_dim )
1864     -- ( \l_@@_x_final_dim , \l_@@_y_final_dim ) ;
1865   \end { scope }
1866 }
```

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

```

1867 \cs_new_protected:Npn \@@_draw_standard_dotted_line:
1868 {
1869   \pgfrememberpicturepositiononpagetrue
1870   \pgf@relevantforpicturesizefalse
1871   \group_begin:
```

The dimension `\l_@@_l_dim` is the length  $\ell$  of the line to draw. We use the floating point reals of `expl3` to compute this length.

```

1872   \dim_zero_new:N \l_@@_l_dim
1873   \dim_set:Nn \l_@@_l_dim
1874   {
1875     \fp_to_dim:n
1876     {
1877       sqrt
1878       (
1879         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) ^ 2
1880         +
1881         ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) ^ 2
1882       )
1883     }
1884   }
```

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.

```

1885   \bool_lazy_or:nnF
1886   { \dim_compare_p:nNn { \dim_abs:n \l_@@_l_dim } > \c_@@_max_l_dim }
1887   { \dim_compare_p:nNn \l_@@_l_dim = \c_zero_dim }
1888   \@@_actually_draw_line:
1889   \group_end:
1890 }
1891 \dim_const:Nn \c_@@_max_l_dim { 50 cm }
1892 \cs_new_protected:Npn \@@_actually_draw_line:
1893 {
```

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

```

1894   \bool_if:NTF \l_@@_initial_open_bool
1895   {
1896     \bool_if:NTF \l_@@_final_open_bool
1897     {
1898       \int_set:Nn \l_tmpa_int
1899       { \dim_ratio:nn \l_@@_l_dim \l_@@_inter_dots_dim }
1900     }
1901     {
1902       \int_set:Nn \l_tmpa_int
```

```

1903         {
1904             \dim_ratio:nn
1905             { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
1906             \l_@@_inter_dots_dim
1907         }
1908     }
1909 }
1910 {
1911     \bool_if:NTF \l_@@_final_open_bool
1912     {
1913         \int_set:Nn \l_tmpa_int
1914         {
1915             \dim_ratio:nn
1916             { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
1917             \l_@@_inter_dots_dim
1918         }
1919     }
1920     {
1921         \int_set:Nn \l_tmpa_int
1922         {
1923             \dim_ratio:nn
1924             { \l_@@_l_dim - 2 \l_@@_xdots_shorten_dim }
1925             \l_@@_inter_dots_dim
1926         }
1927     }
1928 }

```

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

```

1929     \dim_set:Nn \l_tmpa_dim
1930     {
1931         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
1932         \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
1933     }
1934     \dim_set:Nn \l_tmpb_dim
1935     {
1936         ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
1937         \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
1938     }

```

The length  $\ell$  is the length of the dotted line. We note  $\Delta$  the length between two dots and  $n$  the number of intervals between dots. We note  $\delta = \frac{1}{2}(\ell - n\Delta)$ . The distance between the initial extremity of the line and the first dot will be equal to  $k \cdot \delta$  where  $k = 0, 1$  or  $2$ . We first compute this number  $k$  in `\l_tmpb_int`.

```

1939     \int_set:Nn \l_tmpb_int
1940     {
1941         \bool_if:NTF \l_@@_initial_open_bool
1942         { \bool_if:NTF \l_@@_final_open_bool 1 0 }
1943         { \bool_if:NTF \l_@@_final_open_bool 2 1 }
1944     }

```

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.

```

1945     \dim_gadd:Nn \l_@@_x_initial_dim
1946     {
1947         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
1948         \dim_ratio:nn
1949         { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
1950         { 2 \l_@@_l_dim }
1951         * \l_tmpb_int
1952     }
1953     \dim_gadd:Nn \l_@@_y_initial_dim
1954     {
1955         ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
1956         \dim_ratio:nn
1957         { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }

```

```

1958         { 2 \l_@@_l_dim }
1959     * \l_tmpb_int
1960 }
1961 \pgf@relevantforpicturesizefalse
1962 \int_step_inline:nnn 0 \l_tmpa_int
1963 {
1964     \pgfpathcircle
1965     { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
1966     { \l_@@_radius_dim }
1967     \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
1968     \dim_add:Nn \l_@@_y_initial_dim \l_tmpb_dim
1969 }
1970 \pgfusepathqfill
1971 }

```

## 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 for now, they are still available with an error.

```

1972 \NewDocumentCommand \@@_Ldots { s O { } }
1973 {
1974     \bool_if:nTF { #1 }
1975     { \@@_error:n { starred~commands } }
1976     { \@@_instruction_of_type:nn { Ldots } { #2 } }
1977     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_ldots }
1978     \bool_gset_true:N \g_@@_empty_cell_bool
1979 }

```

```

1980 \NewDocumentCommand \@@_Cdots { s O { } }
1981 {
1982     \bool_if:nTF { #1 }
1983     { \@@_error:n { starred~commands } }
1984     { \@@_instruction_of_type:nn { Cdots } { #2 } }
1985     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_cdots }
1986     \bool_gset_true:N \g_@@_empty_cell_bool
1987 }

```

```

1988 \NewDocumentCommand \@@_Vdots { s O { } }
1989 {
1990     \bool_if:nTF { #1 }
1991     { \@@_error:n { starred~commands } }
1992     { \@@_instruction_of_type:nn { Vdots } { #2 } }
1993     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_vdots }
1994     \bool_gset_true:N \g_@@_empty_cell_bool
1995 }

```

```

1996 \NewDocumentCommand \@@_Ddots { s O { } }
1997 {
1998     \bool_if:nTF { #1 }
1999     { \@@_error:n { starred~commands } }
2000     { \@@_instruction_of_type:nn { Ddots } { #2 } }
2001     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_ddots }
2002     \bool_gset_true:N \g_@@_empty_cell_bool
2003 }

```



```

2004 \NewDocumentCommand \@@_Iddots { s O { } }
2005 {
2006   \bool_if:nTF { #1 }
2007     { \@@_error:n { starred~commands } }
2008     { \@@_instruction_of_type:nn { Iddots } { #2 } }
2009   \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_iddots }
2010   \bool_gset_true:N \g_@@_empty_cell_bool
2011 }

```

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

```

2012 \cs_new_protected:Npn \@@_Hspace:
2013 {
2014   \bool_gset_true:N \g_@@_empty_cell_bool
2015   \hspace
2016 }

```

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

```

2017 \cs_set_eq:NN \@@_old_multicolumn \multicolumn
2018 \cs_new:Npn \@@_multicolumn:nnn #1 #2 #3
2019 {
2020   \@@_old_multicolumn { #1 } { #2 } { #3 }
2021   \int_compare:nNnT #1 > 1
2022     {
2023       \seq_gput_left:Nx \g_@@_multicolumn_cells_seq
2024         { \int_eval:n \c@iRow - \int_use:N \c@jCol }
2025       \seq_gput_left:Nn \g_@@_multicolumn_sizes_seq { #1 }
2026     }
2027   \int_gadd:Nn \c@jCol { #1 - 1 }
2028 }

```

The command `\@@_Hdotsfor` will be linked to `\Hdotsfor` in `{NiceArrayWithDelims}`. This command uses an optional argument (as does `\hdotsfor`) but this argument is discarded (in `\hdotsfor`, this argument is used for fine tuning of the space between two consecutive dots). Tikz nodes are created also the implicit cells of the `\Hdotsfor` (maybe we should modify that point).

This command must *not* be protected since it begins with `\multicolumn`.

```

2029 \cs_new:Npn \@@_Hdotsfor:
2030 {
2031   \multicolumn { 1 } { C } { }
2032   \@@_Hdotsfor_i
2033 }

```

The command `\@@_Hdotsfor_i` is defined with the tools of `xparse` because it has an optional argument. Note that such a command defined by `\NewDocumentCommand` is protected and that's why we have put the `\multicolumn` before (in the definition of `\@@_Hdotsfor:`).

```

2034 \bool_if:NTF \c_@@_draft_bool
2035 {

```

We don't put `!` before the last optionnal argument for homogeneity with `\Cdots`, etc. which have only one optional argument.

```

2036   \NewDocumentCommand \@@_Hdotsfor_i { O { } m O { } }
2037     { \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { C } { } } }
2038 }
2039 {
2040   \NewDocumentCommand \@@_Hdotsfor_i { O { } m O { } }
2041     {
2042       \tl_gput_right:Nx \g_@@_Hdotsfor_lines_tl
2043       {
2044         \@@_Hdotsfor:nnnn
2045         { \int_use:N \c@iRow }
2046         { \int_use:N \c@jCol }
2047         { #2 }

```

```

2048         { #3 }
2049     }
2050     \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { C } { } }
2051 }
2052 }

2053 \cs_new_protected:Npn \@@_Hdotsfor:nnnn #1 #2 #3 #4
2054 {
2055     \bool_set_false:N \l_@@_initial_open_bool
2056     \bool_set_false:N \l_@@_final_open_bool

For the row, it's easy.
2057     \int_set:Nn \l_@@_initial_i_int { #1 }
2058     \int_set_eq:NN \l_@@_final_i_int \l_@@_initial_i_int
For the column, it's a bit more complicated.
2059     \int_compare:nNnTF #2 = 1
2060     {
2061         \int_set:Nn \l_@@_initial_j_int 1
2062         \bool_set_true:N \l_@@_initial_open_bool
2063     }
2064     {
2065         \cs_if_exist:cTF
2066         {
2067             pgf @ sh @ ns @ \@@_env:
2068             - \int_use:N \l_@@_initial_i_int
2069             - \int_eval:n { #2 - 1 }
2070         }
2071         { \int_set:Nn \l_@@_initial_j_int { #2 - 1 } }
2072         {
2073             \int_set:Nn \l_@@_initial_j_int { #2 }
2074             \bool_set_true:N \l_@@_initial_open_bool
2075         }
2076     }
2077     \int_compare:nNnTF { #2 + #3 - 1 } = \c@jCol
2078     {
2079         \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
2080         \bool_set_true:N \l_@@_final_open_bool
2081     }
2082     {
2083         \cs_if_exist:cTF
2084         {
2085             pgf @ sh @ ns @ \@@_env:
2086             - \int_use:N \l_@@_final_i_int
2087             - \int_eval:n { #2 + #3 }
2088         }
2089         { \int_set:Nn \l_@@_final_j_int { #2 + #3 } }
2090         {
2091             \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
2092             \bool_set_true:N \l_@@_final_open_bool
2093         }
2094     }

2095     \group_begin:
2096     \int_compare:nNnTF { #1 } = 0
2097     { \color { nicematrix-first-row } }
2098     {
2099         \int_compare:nNnT { #1 } = \g_@@_row_total_int
2100         { \color { nicematrix-last-row } }
2101     }
2102     \keys_set:nn { NiceMatrix / xdots } { #4 }
2103     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2104     \@@_actually_draw_Ldots:
2105     \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).

```

2106 \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
2107 { \cs_set:cpn { @@ _ dotted _ #1 - ##1 } { } }
2108 }

```

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

The command will exit three levels of groups in order to execute the command

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

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

```

2109 \cs_new_protected:Npn \@@_rotate: { \group_insert_after:N \@@_rotate_i: }
2110 \cs_new_protected:Npn \@@_rotate_i: { \group_insert_after:N \@@_rotate_ii: }
2111 \cs_new_protected:Npn \@@_rotate_ii: { \group_insert_after:N \@@_rotate_iii: }
2112 \cs_new_protected:Npn \@@_rotate_iii:
2113 {
2114 \box_rotate:Nn \l_@@_cell_box { 90 }

```

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

```

2115 \int_compare:nNnT \c@iRow = \l_@@_last_row_int
2116 {
2117 \vbox_set_top:Nn \l_@@_cell_box
2118 {
2119 \vbox_to_zero:n { }

```

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

```

2120 \skip_vertical:n { - \box_ht:N \@arstrutbox + 0.8 ex }
2121 \box_use:N \l_@@_cell_box
2122 }
2123 }
2124 }

```

## 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).<sup>38</sup>

```

2125 \cs_new:Npn \@@_double_int_eval:n #1-#2 \q_stop
2126 { \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).

```

2127 \NewDocumentCommand \@@_line { 0 { } m m ! 0 { } }
2128 {
2129 \group_begin:
2130 \keys_set:nn { NiceMatrix / xdots } { #1 , #4 }
2131 \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2132 \use:x
2133 {
2134 \@@_line_i:nn
2135 { \@@_double_int_eval:n #2 \q_stop }
2136 { \@@_double_int_eval:n #3 \q_stop }
2137 }
2138 \group_end:
2139 }

```

---

<sup>38</sup>Indeed, we want that the user may use the command `\line` in `code-after` with LaTeX counters in the arguments — with the command `\value`.

```

2140 \bool_if:NTF \c_@@_draft_bool
2141 { \cs_new_protected:Npn \@@_line_i:nn #1 #2 { } }
2142 {
2143   \cs_new_protected:Npn \@@_line_i:nn #1 #2
2144   {
2145     \bool_set_false:N \l_@@_initial_open_bool
2146     \bool_set_false:N \l_@@_final_open_bool
2147     \bool_if:NTF
2148     {
2149       \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #1 }
2150       ||
2151       \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #2 }
2152     }
2153     {
2154       \@@_error:nnn { unknown~cell~for~line~in~code~after } { #1 } { #2 }
2155     }
2156     { \@@_draw_line_ii:nn { #1 } { #2 } }
2157   }
2158 }
2159 \AtBeginDocument
2160 {
2161   \cs_new_protected:Npx \@@_draw_line_ii:nn #1 #2
2162   {

```

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:`.

```

2163   \c_@@_pgfortikzpicture_tl
2164   \@@_draw_line_iii:nn { #1 } { #2 }
2165   \c_@@_endpgfortikzpicture_tl
2166 }
2167 }

```

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

```

2168 \cs_new_protected:Npn \@@_draw_line_iii:nn #1 #2
2169 {
2170   \pgfrememberpicturepositiononpagetrue
2171   \pgfpointshapeborder { \@@_env: - #1 } { \@@_qpoint: { #2 } }
2172   \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2173   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2174   \pgfpointshapeborder { \@@_env: - #2 } { \@@_qpoint: { #1 } }
2175   \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2176   \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2177   \@@_draw_line:
2178 }

```

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).

## 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`.

```

2179 \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`.

```

2180 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix:n #1
2181 {
2182   \int_compare:nNnTF \l_@@_first_col_int = 0
2183     { \@@_OnlyMainNiceMatrix_i:n { #1 } }
2184     {
2185       \int_compare:nNnTF \c@jCol = 0
2186         {
2187           \int_compare:nNnF \c@iRow = { -1 }
2188             { \int_compare:nNnF \c@iRow = { \l_@@_last_row_int - 1 } { #1 } }
2189         }
2190       { \@@_OnlyMainNiceMatrix_i:n { #1 } }
2191     }
2192 }

```

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.

```

2193 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix_i:n #1
2194 {
2195   \int_compare:nNnF \c@iRow = 0
2196     { \int_compare:nNnF \c@iRow = \l_@@_last_row_int { #1 } }
2197 }

```

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

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

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

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

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

That's why we have done a redefinition of the macro `\@arrayrule` of `array` and this redefinition will add `\@@_vline`: instead of `\vline` to the preamble (that definition is in the beginning of `{NiceArrayWithDelims}`). Here is the definition of `\@@_vline`:. This definition *must* be protected because you don't want that macro expanded during the construction of the preamble (the tests in `\@@_OnlyMainNiceMatrix:n` must be effective in each row and not once for all when the preamble is constructed).

```
2198 \cs_new_protected:Npn \@@_vline: { \@@_OnlyMainNiceMatrix:n { \@@_vline_i: } }
```

If `colortbl` is loaded, the following macro will be redefined (in a `\AtBeginDocument`) to take into account the color fixed by `\arrayrulecolor` of `colortbl`.

```
2199 \cs_set_eq:NN \@@_vline_i: \vline
```

The command `\@@_draw_vlines` will be executed when the user uses the option `vlines` (which draws all the vlines of the array).

```

2200 \cs_new_protected:Npn \@@_draw_vlines:
2201 {
2202   \group_begin:

```

The command `\CT@arc@` is a command of color from `colortbl`.

```

2203   \bool_if:NT \c_@@_colortbl_loaded_bool \CT@arc@
2204   \pgfpicture
2205   \pgfrememberpicturepositiononpagetrue
2206   \pgf@relevantforpicturesizefalse
2207   \pgfsetlinewidth \arrayrulewidth

```

First, we compute in `\l_tmpa_dim` the height of the rules we have to draw.

```

2208 \@@_qpoint: {row - 1 }
2209 \dim_set_eq:NN \l_tmpa_dim \pgf@y
2210 \pgfusepathqfill
2211 \@@_qpoint: { row - \@@_succ:n \c@iRow }
2212 \dim_sub:Nn \l_tmpa_dim \pgf@y
2213 \pgfusepathqfill

```

We translate vertically to take into account the potential “last row”.

```

2214 \dim_zero:N \l_tmpb_dim
2215 \int_compare:nNnT \l_@@_last_row_int > { -1 }
2216 {
2217   \dim_set_eq:NN \l_tmpb_dim \g_@@_dp_last_row_dim
2218   \dim_add:Nn \l_tmpb_dim \g_@@_ht_last_row_dim

```

We adjust the value of `\l_tmpa_dim` by the width of the horizontal rule just before the “last row”.

```

2219 \@@_qpoint: { row - \@@_succ:n \c@iRow }
2220 \dim_add:Nn \l_tmpa_dim \pgf@y
2221 \@@_qpoint: { row - \@@_succ:n \g_@@_row_total_int }
2222 \dim_sub:Nn \l_tmpa_dim \pgf@y
2223 \dim_sub:Nn \l_tmpa_dim \l_tmpb_dim
2224 }

```

Now, we can draw the lines with a loop.

```

2225 \int_step_inline:nnn
2226 { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
2227 { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@jCol } \c@jCol }
2228 {
2229   \pgfpathmoveto
2230   {
2231     \pgfpointadd
2232     { \@@_qpoint: { col - ##1 } }
2233     {
2234       \pgfpoint
2235       {
2236         -0.5 \arrayrulewidth
2237         \int_compare:nNnT { ##1 } = 1
2238         {
2239           \int_compare:nNnT \l_@@_first_col_int = 1
2240           { + \arrayrulewidth }
2241         }
2242       }
2243       { \l_tmpb_dim }
2244     }
2245   }
2246   \pgfpathlineto
2247   {
2248     \pgfpointadd
2249     { \@@_qpoint: { col - ##1 } }
2250     {
2251       \pgfpoint
2252       {
2253         -0.5 \arrayrulewidth
2254         \int_compare:nNnT { ##1 } = 1
2255         {
2256           \int_compare:nNnT \l_@@_first_col_int = 1
2257           { + \arrayrulewidth }
2258         }
2259       }
2260       { \l_tmpb_dim + \l_tmpa_dim }
2261     }
2262   }
2263 }
2264 \pgfusepathqstroke
2265 \endpgfpicture

```

```

2266   \group_end:
2267 }

```

## 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`.

```

2268 \bool_if:NTF \c_@@_draft_bool
2269 { \cs_new:Npn \@@_hdottedline: { } }
2270 {
2271   \cs_new:Npn \@@_hdottedline:
2272   {
2273     \noalign { \skip_vertical:N 2\l_@@_radius_dim }
2274     \@@_hdottedline_i:
2275   }
2276 }

```

On the other side, the following command should be protected.

```

2277 \cs_new_protected:Npn \@@_hdottedline_i:
2278 {

```

We write in the code-after the instruction that will eventually 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).

```

2279   \tl_gput_right:Nx \g_@@_internal_code_after_tl
2280   { \@@_hdottedline:n { \int_use:N \c@iRow } }
2281 }

```

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.

```

2282 \AtBeginDocument
2283 {

```

We recall that, when externalization is used, `\tikzpicture` and `\endtikzpicture` (or `\pgfpicture` and `\endpgfpicture`) must be directly “visible”.

```

2284   \cs_new_protected:Npx \@@_hdottedline:n #1
2285   {
2286     \bool_set_true:N \exp_not:N \l_@@_initial_open_bool
2287     \bool_set_true:N \exp_not:N \l_@@_final_open_bool
2288     \c_@@_pgfortikzpicture_tl
2289     \@@_hdottedline_i:n { #1 }
2290     \c_@@_endpgfortikzpicture_tl
2291   }
2292 }

```

The following command *must* be protected since it is used in the construction of `\@@_hdottedline:n`.

```

2293 \cs_new_protected:Npn \@@_hdottedline_i:n #1
2294 {
2295   \pgfrememberpicturepositiononpagetrue
2296   \@@_qpoint: { 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).

```

2297   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2298   \dim_sub:Nn \l_@@_y_initial_dim \l_@@_radius_dim
2299   \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}
```

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

```
2300 \@@_qpoint: { col - 1 }
2301 \dim_set:Nn \l_@@_x_initial_dim
2302 { \pgf@x + \arraycolsep - \l_@@_left_margin_dim }
2303 \@@_qpoint: { col - \@@_succ:n \c@jCol }
2304 \dim_set:Nn \l_@@_x_final_dim
2305 { \pgf@x - \arraycolsep + \l_@@_right_margin_dim }
```

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.

```
2306 \tl_set:Nn \l_tmpa_tl { ( }
2307 \tl_if_eq:NNF \l_@@_left_delim_tl \l_tmpa_tl
2308 { \dim_gadd:Nn \l_@@_x_initial_dim { 0.5 \l_@@_inter_dots_dim } }
2309 \tl_set:Nn \l_tmpa_tl { ) }
2310 \tl_if_eq:NNF \l_@@_right_delim_tl \l_tmpa_tl
2311 { \dim_gsub:Nn \l_@@_x_final_dim { 0.5 \l_@@_inter_dots_dim } }
```

As for now, we have no option to control the style of the lines drawn by `\hdottedline` and the specifier “:” in the preamble. That’s why we impose the style `standard`.

```
2312 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
2313 \@@_draw_line:
2314 }
```

## Vertical dotted lines

```
2315 \bool_if:nTF \c_@@_draft_bool
2316 { \cs_new_protected:Npn \@@_vdottedline:n #1 { } }
2317 {
2318 \cs_new_protected:Npn \@@_vdottedline:n #1
2319 {
2320 \bool_set_true:N \l_@@_initial_open_bool
2321 \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”.

```
2322 \bool_if:NTF \c_@@_tikz_loaded_bool
2323 {
2324 \tikzpicture
2325 \@@_vdottedline_i:n { #1 }
2326 \endtikzpicture
2327 }
2328 {
2329 \pgfpicture
2330 \@@_vdottedline_i:n { #1 }
2331 \endpgfpicture
2332 }
2333 }
2334 }
```



```

2335 \cs_new_protected:Npn \@@_vdottedline_i:n #1
2336 {

```

The command `\CT@arc@` is a command of color from `colortbl`.

```

2337 \bool_if:NT \c_@@_colortbl_loaded_bool \CT@arc@
2338 \pgfrememberpicturepositiononpagetrue
2339 \@@_qpoint: { 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).

```

2340 \dim_set:Nn \l_@@_x_initial_dim { \pgf@x - \l_@@_radius_dim }
2341 \dim_set:Nn \l_@@_x_final_dim { \pgf@x - \l_@@_radius_dim }
2342 \@@_qpoint: { 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.

```

2343 \dim_set:Nn \l_@@_y_initial_dim { \pgf@y - 0.5 \l_@@_inter_dots_dim }
2344 \@@_qpoint: { row - \@@_succ:n \c@iRow }
2345 \dim_set:Nn \l_@@_y_final_dim { \pgf@y + 0.5 \l_@@_inter_dots_dim }

```

As for now, we have no option to control the style of the lines drawn by `\hdottedline` and the specifier “:” in the preamble. That’s why we impose the style `standard`.

```

2346 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
2347 \@@_draw_line:
2348 }

```

## 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.

```

2349 \bool_new:N \l_@@_block_auto_columns_width_bool

```

As of now, there is only one option available for the environment `{NiceMatrixBlock}`.

```

2350 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
2351 {
2352   auto-columns-width .code:n =
2353   {
2354     \bool_set_true:N \l_@@_block_auto_columns_width_bool
2355     \dim_gzero_new:N \g_@@_max_cell_width_dim
2356     \bool_set_true:N \l_@@_auto_columns_width_bool
2357   }
2358 }

2359 \NewDocumentEnvironment { NiceMatrixBlock } { ! 0 { } }
2360 {
2361   \int_gincr:N \g_@@_NiceMatrixBlock_int
2362   \dim_zero:N \l_@@_columns_width_dim
2363   \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
2364   \bool_if:NT \l_@@_block_auto_columns_width_bool
2365   {
2366     \cs_if_exist:cT { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
2367     {
2368       \exp_args:NNc \dim_set:Nn \l_@@_columns_width_dim
2369       { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
2370     }
2371   }
2372 }

```

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`).

```

2373 {
2374   \bool_if:NT \l_@@_block_auto_columns_width_bool
2375   {
2376     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
2377     \iow_shipout:Nx \@mainaux
2378     {
2379       \cs_gset:cpn
2380       { @@ _ max _ cell _ width _ \int_use:N \g_@@_NiceMatrixBlock_int }

```

For technical reasons, we have to include the width of an eventual rule on the right side of the cells.

```

2381       { \dim_eval:n { \g_@@_max_cell_width_dim + \arrayrulewidth } }
2382     }
2383     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
2384   }
2385 }

```

## The extra nodes

First, two variants of the functions `\dim_min:nn` and `\dim_max:nn`.

```

2386 \cs_generate_variant:Nn \dim_min:nn { v n }
2387 \cs_generate_variant:Nn \dim_max:nn { v n }

```

We have three macros of creation of nodes: `\@@_create_medium_nodes:`, `\@@_create_large_nodes:` and `\@@_create_medium_and_large_nodes:`.

We have to compute the mathematical coordinates of the “medium nodes”. These mathematical coordinates are also used to compute the mathematical coordinates of the “large nodes”. That's why we write a command `\@@_computations_for_medium_nodes:` to do these computations.

The command `\@@_computations_for_medium_nodes:` must be used in a `{pgfpicture}`.

For each row  $i$ , we compute two dimensions `l_@@_row_i_min_dim` and `l_@@_row_i_max_dim`. The dimension `l_@@_row_i_min_dim` is the minimal  $y$ -value of all the cells of the row  $i$ . The dimension `l_@@_row_i_max_dim` is the maximal  $y$ -value of all the cells of the row  $i$ .

Similarly, for each column  $j$ , we compute two dimensions `l_@@_column_j_min_dim` and `l_@@_column_j_max_dim`. The dimension `l_@@_column_j_min_dim` is the minimal  $x$ -value of all the cells of the column  $j$ . The dimension `l_@@_column_j_max_dim` is the maximal  $x$ -value of all the cells of the column  $j$ .

Since these dimensions will be computed as maximum or minimum, we initialize them to `\c_max_dim` or `-\c_max_dim`.

```

2388 \cs_new_protected:Npn \@@_computations_for_medium_nodes:
2389 {
2390   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
2391   {
2392     \dim_zero_new:c { l_@@_row_\@@_i: _min_dim }
2393     \dim_set_eq:cN { l_@@_row_\@@_i: _min_dim } \c_max_dim
2394     \dim_zero_new:c { l_@@_row_\@@_i: _max_dim }
2395     \dim_set:cn { l_@@_row_\@@_i: _max_dim } { - \c_max_dim }
2396   }
2397   \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
2398   {
2399     \dim_zero_new:c { l_@@_column_\@@_j: _min_dim }
2400     \dim_set_eq:cN { l_@@_column_\@@_j: _min_dim } \c_max_dim
2401     \dim_zero_new:c { l_@@_column_\@@_j: _max_dim }
2402     \dim_set:cn { l_@@_column_\@@_j: _max_dim } { - \c_max_dim }
2403   }

```

We begin the two nested loops over the rows and the columns of the array.

```

2404 \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
2405 {
2406   \int_step_variable:nnNn
2407   \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.

```

2408 {
2409   \cs_if_exist:cT
2410   { 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`.

```

2411 {
2412   \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { south-west }
2413   \dim_set:cn { l_@@_row_ \@@_i: _min_dim }
2414   { \dim_min:vn { l_@@_row _ \@@_i: _min_dim } \pgf@y }
2415   \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
2416   {
2417     \dim_set:cn { l_@@_column _ \@@_j: _min_dim }
2418     { \dim_min:vn { l_@@_column _ \@@_j: _min_dim } \pgf@x }
2419   }

```

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`.

```

2420   \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { north-east }
2421   \dim_set:cn { l_@@_row _ \@@_i: _max_dim }
2422   { \dim_max:vn { l_@@_row _ \@@_i: _max_dim } \pgf@y }
2423   \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
2424   {
2425     \dim_set:cn { l_@@_column _ \@@_j: _max_dim }
2426     { \dim_max:vn { l_@@_column _ \@@_j: _max_dim } \pgf@x }
2427   }
2428 }
2429 }
2430 }

```

Now, we have to deal with empty rows or empty columns since we don't have created nodes in such rows and columns.

```

2431 \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
2432 {
2433   \dim_compare:nNnT
2434   { \dim_use:c { l_@@_row _ \@@_i: _min _ dim } } = \c_max_dim
2435   {
2436     \@@_qpoint: { row - \@@_i: - base }
2437     \dim_set:cn { l_@@_row _ \@@_i: _max _ dim } \pgf@y
2438     \dim_set:cn { l_@@_row _ \@@_i: _min _ dim } \pgf@y
2439   }
2440 }
2441 \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
2442 {
2443   \dim_compare:nNnT
2444   { \dim_use:c { l_@@_column _ \@@_j: _min _ dim } } = \c_max_dim
2445   {
2446     \@@_qpoint: { col - \@@_j: }
2447     \dim_set:cn { l_@@_column _ \@@_j: _max _ dim } \pgf@y
2448     \dim_set:cn { l_@@_column _ \@@_j: _min _ dim } \pgf@y
2449   }
2450 }
2451 }

```

Here is the command `\@@_create_medium_nodes:.` When this command is used, the “medium nodes” are created.

```

2452 \cs_new_protected:Npn \@@_create_medium_nodes:
2453 {
2454   \pgfpicture
2455     \pgfrememberpicturepositiononpagetrue
2456     \pgf@relevantforpicturesizefalse
2457     \@@_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”.

```

2458     \tl_set:Nn \l_@@_suffix_tl { -medium }
2459     \@@_create_nodes:
2460     \endpgfpicture
2461   }

```

The command `\@@_create_large_nodes:` must be used when we want to create only the “large nodes” and not the medium ones (if we want to create both, we have to use `\@@_create_medium_and_large_nodes:`). 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:`.

```

2462 \cs_new_protected:Npn \@@_create_large_nodes:
2463 {
2464   \pgfpicture
2465     \pgfrememberpicturepositiononpagetrue
2466     \pgf@relevantforpicturesizefalse
2467     \@@_computations_for_medium_nodes:
2468     \@@_computations_for_large_nodes:
2469     \tl_set:Nn \l_@@_suffix_tl { - large }
2470     \@@_create_nodes:
2471   \endpgfpicture
2472 }
2473 \cs_new_protected:Npn \@@_create_medium_and_large_nodes:
2474 {
2475   \pgfpicture
2476     \pgfrememberpicturepositiononpagetrue
2477     \pgf@relevantforpicturesizefalse
2478     \@@_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”.

```

2479     \tl_set:Nn \l_@@_suffix_tl { - medium }
2480     \@@_create_nodes:
2481     \@@_computations_for_large_nodes:
2482     \tl_set:Nn \l_@@_suffix_tl { - large }
2483     \@@_create_nodes:
2484   \endpgfpicture
2485 }

```

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.

```

2486 \cs_new_protected:Npn \@@_computations_for_large_nodes:
2487 {
2488   \int_set:Nn \l_@@_first_row_int 1
2489   \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`.

```

2490   \int_step_variable:nNn { \c@iRow - 1 } \@@_i:
2491   {
2492     \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim }
2493     {
2494       (
2495         \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } +
2496         \dim_use:c { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
2497       )

```

```

2498         / 2
2499     }
2500     \dim_set_eq:cc { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
2501     { l_@@_row _ \@@_i: _ min _ dim }
2502 }
2503 \int_step_variable:nNn { \c@jCol - 1 } \@@_j:
2504 {
2505     \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim }
2506     {
2507         (
2508             \dim_use:c { l_@@_column _ \@@_j: _ max _ dim } +
2509             \dim_use:c
2510             { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
2511         )
2512         / 2
2513     }
2514     \dim_set_eq:cc { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
2515     { l_@@_column _ \@@_j: _ max _ dim }
2516 }
2517 % \end{macrocode}
2518 % Here, we have to use |\dim_sub:cn| because of the number 1 in the name.
2519 % \begin{macrocode}
2520 \dim_sub:cn
2521 { l_@@_column _ 1 _ min _ dim }
2522 \l_@@_left_margin_dim
2523 \dim_add:cn
2524 { l_@@_column _ \int_use:N \c@jCol _ max _ dim }
2525 \l_@@_right_margin_dim
2526 }

```

The control sequence `\@@_create_nodes:` is used twice: for the construction of the “medium nodes” and for the construction of the “large nodes”. The nodes are constructed with the value of all the dimensions `l_@@_row_i_min_dim`, `l_@@_row_i_max_dim`, `l_@@_column_j_min_dim` and `l_@@_column_j_max_dim`. Between the construction of the “medium nodes” and the “large nodes”, the values of these dimensions are changed. The function also uses `\l_@@_suffix_tl` (-medium or -large).

```

2527 \cs_new_protected:Npn \@@_create_nodes:
2528 {
2529     \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
2530     {
2531         \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
2532         {

```

We draw the rectangular node for the cell  $(\@@_i-\@@_j)$ .

```

2533     \@@_pgf_rect_node:nnnnn
2534     { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
2535     { \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } }
2536     { \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } }
2537     { \dim_use:c { l_@@_column _ \@@_j: _ max _ dim } }
2538     { \dim_use:c { l_@@_row _ \@@_i: _ max _ dim } }
2539     \str_if_empty:NF \l_@@_name_str
2540     {
2541         \pgfnodealias
2542         { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
2543         { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
2544     }
2545 }
2546 }

```

Now, we create the nodes for the cells of the `\multicolumn`. We recall that we have stored in `\g_@@_multicolumn_cells_seq` the list of the cells where a `\multicolumn{n}{...}{...}` with  $n > 1$  was issued and in `\g_@@_multicolumn_sizes_seq` the correspondent values of  $n$ .

```

2547 \seq_mapthread_function:NNN
2548 \g_@@_multicolumn_cells_seq
2549 \g_@@_multicolumn_sizes_seq

```

```

2550 \@@_node_for_multicolumn:nn
2551 }

2552 \cs_new_protected:Npn \@@_extract_coords_values: #1 - #2 \q_stop
2553 {
2554   \cs_set:Npn \@@_i: { #1 }
2555   \cs_set:Npn \@@_j: { #2 }
2556 }

```

The command `\@@_node_for_multicolumn:nn` takes two arguments. The first is the position of the cell where the command `\multicolumn{n}{...}{...}` was issued in the format  $i$ - $j$  and the second is the value of  $n$  (the length of the “multi-cell”).

```

2557 \cs_new_protected:Npn \@@_node_for_multicolumn:nn #1 #2
2558 {
2559   \@@_extract_coords_values: #1 \q_stop
2560   \@@_pgf_rect_node:nnnnn
2561   { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
2562   { \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } }
2563   { \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } }
2564   { \dim_use:c { l_@@_column _ \int_eval:n { \@@_j: +#2-1 } _ max _ dim } }
2565   { \dim_use:c { l_@@_row _ \@@_i: _ max _ dim } }
2566   \str_if_empty:NF \l_@@_name_str
2567   {
2568     \pgfnodealias
2569     { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
2570     { \int_use:N \g_@@_env_int - \@@_i: - \@@_j: \l_@@_suffix_tl }
2571   }
2572 }

```

## Block matrices

The code in this section is for the construction of *block matrices*. It has no direct link with the environment `{NiceMatrixBlock}`.

The following command will be linked to `\Block` in the environments of `nicematrix`. We define it with `\NewDocumentCommand` of `xparse` because it has an optional argument between `<` and `>` (for TeX instructions put before the math mode of the label)

```

2573 \NewDocumentCommand \@@_Block: { 0 { } m D < > { } m }
2574 { \@@_Block_i #2 \q_stop { #1 } { #3 } { #4 } }

```

The first mandatory argument of `\@@_Block:` has a special syntax. It must be of the form  $i$ - $j$  where  $i$  and  $j$  are the size (in rows and columns) of the block.

```

2575 \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.

```

2576 \cs_new_protected:Npn \@@_Block_ii:nnnnn #1 #2 #3 #4 #5
2577 {

```

We write an instruction in the `code-after`. We write the instruction in the beginning of the `code-after` (the `left` in `\tl_gput_left:Nx`) because we want the Tikz nodes corresponding of the block created *before* potential instructions written by the user in the `code-after` (these instructions may use the Tikz node of the created block).

```

2578 \tl_gput_left:Nx \g_@@_internal_code_after_tl
2579 {
2580   \@@_Block_iii:nnnnnn
2581   { \int_use:N \c@iRow }
2582   { \int_use:N \c@jCol }
2583   { \int_eval:n { \c@iRow + #1 - 1 } }
2584   { \int_eval:n { \c@jCol + #2 - 1 } }
2585   { #3 }
2586   \exp_not:n { { #4 $ #5 $ } }
2587 }

```

It's not allowed to use the command `\Block` twice in the same cell of the array. That's why, at the first use, we link the command `\Block` to a special version. The scope of this link is the cell of the array.

```

2588   \cs_set_eq:NN \Block \@@_Block_error:nn
2589 }

2590 \cs_new:Npn \@@_Block_error:nn #1 #2
2591 {
2592   \@@_error:n { Second~Block }
2593   \cs_set_eq:NN \Block \use:nn
2594 }

2595 \keys_define:nn { NiceMatrix / Block }
2596 {
2597   tikz .tl_set:N = \l_@@_tikz_tl ,
2598   tikz .value_required:n = true ,
2599   white .bool_set:N = \l_@@_white_bool ,
2600   white .default:n = true ,
2601   white .value_forbidden:n = true ,
2602 }

```

The following command `\@@_Block_iii:nnnnnn` will be used in the code-after.

```

2603 \cs_new_protected:Npn \@@_Block_iii:nnnnnn #1 #2 #3 #4 #5 #6
2604 {

```

The group is for the keys.

```

2605   \group_begin:
2606   \keys_set:nn { NiceMatrix / Block } { #5 }

2607   \bool_if:nTF
2608   {
2609     \int_compare_p:nNn { #3 } > \c@iRow
2610     || \int_compare_p:nNn { #4 } > \c@jCol
2611   }
2612   { \msg_error:nnnn { nicematrix } { Block~too~large } { #1 } { #2 } }
2613   {

```

We put the contents of the cell in the box `\l_@@_cell_box` because we want the command `\rotate` used in the content to be able to rotate the box.

```

2614   \hbox_set:Nn \l_@@_cell_box { #6 }

```

The construction of the node corresponding to the merged cells.

```

2615   \pgfpicture
2616   \pgfrememberpicturepositiononpagetrue
2617   \pgf@relevantforpicturesizefalse
2618   \@@_qpoint: { row - #1 }
2619   \dim_set_eq:NN \l_tmpa_dim \pgf@y
2620   \@@_qpoint: { col - #2 }
2621   \dim_set_eq:NN \l_tmpb_dim \pgf@x
2622   \@@_qpoint: { row - \@@_succ:n { #3 } }
2623   \dim_set_eq:NN \l_tmpc_dim \pgf@y
2624   \@@_qpoint: { col - \@@_succ:n { #4 } }
2625   \dim_set_eq:NN \l_tmpd_dim \pgf@x

```

The following code doesn't work for the first vertical rule. You should allow the option `white` if and only if the option `vlines` and `hlines` has been used.

```

2626   \bool_if:NT \l_@@_white_bool
2627   {
2628     \begin { pgfscope }
2629     \pgfsetfillcolor { white }

```

Usually, the vertical rules are *before* the col-nodes. But there is an exception: if there is no “first col”, the first vertical rule is after the col node.<sup>39</sup>

<sup>39</sup>That's true for the vertical rules drawn by “|” due to the conception of `{array}` (of `array`) and we have managed to have the same behaviour with `vlines`.

Since we don't want the white rectangle to erase a part of this first rule, we have to do an adjustment in this case. *after* the “col node”.

```

2630         \int_compare:nNnT { #2 } = 1
2631         {
2632             \int_compare:nNnT \l_@@_first_col_int = 1
2633             { \dim_add:Nn \l_tmpb_dim \arrayrulewidth }
2634         }
2635         \pgfpathrectanglecorners
2636         { \pgfpoint \l_tmpb_dim { \l_tmpa_dim - \arrayrulewidth } }
2637         { \pgfpoint { \l_tmpd_dim - \arrayrulewidth } \l_tmpc_dim }
2638         \pgfusepathqfill
2639         \end { pgfscope }
2640     }

```

We construct the node for the block with the name (#1-#2-block). The function \@@\_pgf\_rect\_node:nnnnn takes as arguments the name of the node and the four coordinates of two opposite corner points of the rectangle.

```

2641     \begin { pgfscope }
2642     \exp_args:Nx \pgfset { \l_@@_tikz_tl }
2643     \@@_pgf_rect_node:nnnnn
2644     { \@@_env: - #1 - #2 - block }
2645     \l_tmpb_dim \l_tmpa_dim \l_tmpd_dim \l_tmpc_dim
2646     \end { pgfscope }

```

If the creation of the “medium nodes” is required, we create a “medium node” for the block. The function \@@\_pgf\_rect\_node:nnnnn takes as arguments the name of the node and two PGF points.

```

2647     \bool_if:NT \l_@@_medium_nodes_bool
2648     {
2649         \@@_pgf_rect_node:nnn
2650         { \@@_env: - #1 - #2 - block - medium }
2651         { \pgfpointanchor { \@@_env: - #1 - #2 - medium } { north-west } }
2652         { \pgfpointanchor { \@@_env: - #3 - #4 - medium } { south-east } }
2653     }

```

Now, we will put the label of the block.

```

2654     \int_compare:nNnTF { #1 } = { #3 }
2655     {

```

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.

```

2656         \pgfextracty \l_tmpa_dim { \@@_qpoint: { row - #1 - base } }

```

We retrieve (in \pgf@x) the *x*-value of the center of the block.

```

2657         \@@_qpoint: { #1 - #2 - block }

```

We put the label of the block which has been composed in \l\_@@\_cell\_box.

```

2658         \pgftransformshift { \pgfpoint \pgf@x \l_tmpa_dim }
2659         \pgfnode { rectangle } { base }
2660         { \box_use_drop:N \l_@@_cell_box } { } { }
2661     }

```

If the number of rows is different of 1, we put the label of the block in the center of the node (the label of the block has been composed in \l\_@@\_cell\_box).

```

2662     {
2663         \pgftransformshift { \@@_qpoint: { #1 - #2 - block } }
2664         \pgfnode { rectangle } { center }
2665         { \box_use_drop:N \l_@@_cell_box } { } { }
2666     }
2667     \endpgfpicture
2668 }
2669 \group_end:
2670 }

```



## How to draw the dotted lines transparently

```

2671 \cs_set_protected:Npn \@@_renew_matrix:
2672 {
2673   \RenewDocumentEnvironment { pmatrix } { } {
2674     { \pNiceMatrix }
2675     { \endpNiceMatrix }
2676   \RenewDocumentEnvironment { vmatrix } { } {
2677     { \vNiceMatrix }
2678     { \endvNiceMatrix }
2679   \RenewDocumentEnvironment { Vmatrix } { } {
2680     { \VNiceMatrix }
2681     { \endVNiceMatrix }
2682   \RenewDocumentEnvironment { bmatrix } { } {
2683     { \bNiceMatrix }
2684     { \endbNiceMatrix }
2685   \RenewDocumentEnvironment { Bmatrix } { } {
2686     { \BNiceMatrix }
2687     { \endBNiceMatrix }
2688   }

```

## Automatic arrays

```

2689 \cs_new_protected:Npn \@@_set_size:n #1-#2 \q_stop
2690 {
2691   \int_set:Nn \l_@@_nb_rows_int { #1 }
2692   \int_set:Nn \l_@@_nb_cols_int { #2 }
2693 }
2694 \NewDocumentCommand \AutoNiceMatrixWithDelims { m m O { } m O { } m ! O { } }
2695 {
2696   \int_zero_new:N \l_@@_nb_rows_int
2697   \int_zero_new:N \l_@@_nb_cols_int
2698   \@@_set_size:n #4 \q_stop
2699   \begin { NiceArrayWithDelims } { #1 } { #2 }
2700     { * { \l_@@_nb_cols_int } { C } } [ #3 , #5 , #7 ]
2701     \int_compare:nNnT \l_@@_first_row_int = 0
2702     {
2703       \int_compare:nNnT \l_@@_first_col_int = 0 { & }
2704       \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
2705       \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
2706     }
2707     \prg_replicate:nn \l_@@_nb_rows_int
2708     {
2709       \int_compare:nNnT \l_@@_first_col_int = 0 { & }

```

You put { } before #6 to avoid a hasty expansion of an eventual `\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`).

```

2710       \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { { } #6 & } #6
2711       \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
2712     }
2713     \int_compare:nNnT \l_@@_last_row_int > { -2 }
2714     {
2715       \int_compare:nNnT \l_@@_first_col_int = 0 { & }
2716       \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
2717       \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
2718     }
2719     \end { NiceArrayWithDelims }
2720 }
2721 \cs_set_protected:Npn \@@_define_com:nnn #1 #2 #3
2722 {
2723   \cs_set_protected:cpn { #1 AutoNiceMatrix }
2724   {

```

```

2725     \str_gset:Nx \g_@@_name_env_str { #1 AutoNiceMatrix }
2726     \AutoNiceMatrixWithDelims { #2 } { #3 }
2727   }
2728 }

2729 \@@_define_com:nnn p ( )
2730 \@@_define_com:nnn b [ ]
2731 \@@_define_com:nnn v | |
2732 \@@_define_com:nnn V \| \|
2733 \@@_define_com:nnn B \{ \}

```

## We process the options

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.

```

2734 \bool_new:N \c_@@_obsolete_environments_bool
2735 \keys_define:nn { NiceMatrix / Package }
2736 {
2737   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
2738   renew-dots .value_forbidden:n = true ,
2739   renew-matrix .code:n = \@@_renew_matrix: ,
2740   renew-matrix .value_forbidden:n = true ,
2741   transparent .meta:n = { renew-dots , renew-matrix } ,
2742   transparent .value_forbidden:n = true ,
2743   obsolete-environments .bool_set:N = \c_@@_obsolete_environments_bool ,
2744   obsolete-environments .value_forbidden:n = true ,
2745   obsolete-environments .default:n = true ,
2746   starred-commands .code:n =
2747     \@@_msg_redirect_name:nn { starred~commands } { none } ,
2748   starred-commands .value_forbidden:n = true ,
2749 }
2750 }
2751 \ProcessKeysOptions { NiceMatrix / Package }

```

## Error messages of the package

The following command converts all the elements of a sequence (which are token lists) into strings.

```

2752 \cs_new_protected:Npn \@@_convert_to_str_seq:N #1
2753 {
2754   \seq_clear:N \l_tmpa_seq
2755   \seq_map_inline:Nn #1
2756   {
2757     \seq_put_left:Nx \l_tmpa_seq { \tl_to_str:n { ##1 } }
2758   }
2759   \seq_set_eq:NN #1 \l_tmpa_seq
2760 }

```

The following command creates a sequence of strings (`str`) from a `clist`.

```

2761 \cs_new_protected:Npn \@@_set_seq_of_str_from_clist:Nn #1 #2
2762 {
2763   \seq_set_from_clist:Nn #1 { #2 }
2764   \@@_convert_to_str_seq:N #1
2765 }

2766 \@@_set_seq_of_str_from_clist:Nn \c_@@_types_of_matrix_seq
2767 {
2768   NiceMatrix ,
2769   pNiceMatrix , bNiceMatrix , vNiceMatrix, BNiceMatrix, VNiceMatrix
2770 }

```

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`.

```

2771 \cs_new_protected:Npn \@@_error_too_much_cols:
2772 {
2773   \seq_if_in:NVTf \c_@@_types_of_matrix_seq \g_@@_name_env_str
2774   {
2775     \int_compare:nNnTF \l_@@_last_col_int = { -1 }
2776     { \@@_fatal:n { too~much~cols~for~matrix } }
2777     { \@@_fatal:n { too~much~cols~for~matrix~with~last~col } }
2778   }
2779   { \@@_fatal:n { too~much~cols~for~array } }
2780 }

```

The following command must *not* be protected since it's used in an error message.

```

2781 \cs_new:Npn \@@_message_hdotsfor:
2782 {
2783   \tl_if_empty:VF \g_@@_Hdotsfor_lines_tl
2784   { ~Maybe~your~use~of~\token_to_str:N \Hdotsfor\ is~incorrect.}
2785 }
2786 \@@_msg_new:nn { too~much~cols~for~matrix~with~last~col }
2787 {
2788   You~try~to~use~more~columns~than~allowed~by~your~
2789   \@@_full_name_env:.\@@_message_hdotsfor:\ The~maximal~number~of~
2790   columns~is~\int_eval:n { \l_@@_last_col_int - 1 }~(plus~the~potential~
2791   exterior~ones).~This~error~is~fatal.
2792 }
2793 \@@_msg_new:nn { too~much~cols~for~matrix }
2794 {
2795   You~try~to~use~more~columns~than~allowed~by~your~
2796   \@@_full_name_env:.\@@_message_hdotsfor:\ Recall~that~the~maximal~
2797   number~of~columns~for~a~matrix~is~fixed~by~the~LaTeX~counter~
2798   'MaxMatrixCols'.~Its~actual~value~is~\int_use:N \c@MaxMatrixCols.~
2799   This~error~is~fatal.
2800 }

```

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`.

```

2801 \@@_msg_new:nn { too~much~cols~for~array }
2802 {
2803   You~try~to~use~more~columns~than~allowed~by~your~
2804   \@@_full_name_env:.\@@_message_hdotsfor:\ The~maximal~number~of~columns~is~
2805   \int_eval:n { \c@jCol - 1 }~(plus~the~potential~exterior~ones).~
2806   This~error~is~fatal.
2807 }
2808 \@@_msg_new:nn { bad~option~for~line~style }
2809 {
2810   Since~you~haven't~loaded~Tikz,~the~only~value~you~can~give~to~'line~style'~
2811   is~'standard'.~If~you~go~on,~this~option~will~be~ignored.
2812 }
2813 \@@_msg_new:nn { Unknown~option~for~xdots }
2814 {
2815   As~for~now~there~is~only~three~options~available~here:~'color',~'line~style'~
2816   and~'shorten'~(and~you~try~to~use~'\l_keys_key_str').~If~you~go~on,~
2817   this~option~will~be~ignored.
2818 }
2819 \@@_msg_new:nn { ampersand~in~light~syntax }
2820 {
2821   You~can't~use~an~ampersand~(\token_to_str &)~to~separate~columns~because
2822   ~you~have~used~the~option~'light~syntax'.~This~error~is~fatal.
2823 }

```

```

2824 \@@_msg_new:nn { double-backslash-in-light-syntax }
2825 {
2826   You~can't~use~\token_to_str:N \~to~separate~rows~because~you~have~used~
2827   the~option~'light-syntax'.~You~must~use~the~character~'\l_@@_end_of_row_tl'~
2828   (set~by~the~option~'end-of-row').~This~error~is~fatal.
2829 }
2830 \@@_msg_new:nn { starred-commands }
2831 {
2832   The~starred~versions~of~\token_to_str:N \Cdots,~\token_to_str:N \Ldots,~
2833   \token_to_str:N \Vdots,~\token_to_str:N \Ddots\ and~\token_to_str:N \Iddots\
2834   are~deprecated.~However,~you~can~go~on~for~this~time.~If~you~don't~want~to~
2835   see~this~error~we~should~load~'nicematrix'~with~the~option~
2836   'starred-commands'.
2837 }
2838 \@@_msg_new:nn { bad-value-for-baseline }
2839 {
2840   The~value~given~to~'baseline'~(\int_use:N \l_tmpa_int)~is~not~
2841   valid.~The~value~must~be~between~\int_use:N \l_@@_first_row_int\ and~
2842   \int_use:N \g_@@_row_total_int\ or~equal~to~'t',~'c'~or~'b'.\\
2843   If~you~go~on,~a~value~of~1~will~be~used.
2844 }
2845 \@@_msg_new:nn { Second-Block }
2846 {
2847   You~can't~use~\token_to_str:N \Block\ twice~in~the~same~cell~of~the~array.\\
2848   If~you~go~on,~this~command~(and~the~other)~will~be~ignored.
2849 }
2850 \@@_msg_new:nn { empty-environment }
2851 { Your~\@@_full_name_env:\ is~empty.~This~error~is~fatal. }
2852 \@@_msg_new:nn { unknown-cell-for-line-in-code-after }
2853 {
2854   Your~command~\token_to_str:N \line\{#1\}\{#2\}~in~the~'code-after'~
2855   can't~be~executed~because~a~cell~doesn't~exist.\\
2856   If~you~go~on~this~command~will~be~ignored.
2857 }
2858 \@@_msg_new:nn { last-col-non-empty-for-NiceArray }
2859 {
2860   In~the~\@@_full_name_env:,~you~must~use~the~option~
2861   'last-col'~without~value.\\
2862   However,~you~can~go~on~for~this~time~
2863   (the~value~'\l_keys_value_tl'~will~be~ignored).
2864 }
2865 \@@_msg_new:nn { Block-too-large }
2866 {
2867   You~try~to~draw~a~block~in~the~cell~#1-#2~of~your~matrix~but~the~matrix~is~
2868   too~small~for~that~block. \\
2869   If~you~go~on,~this~command~will~be~ignored.
2870 }
2871 \@@_msg_new:nn { Wrong-last-row }
2872 {
2873   You~have~used~'last-row=\int_use:N \l_@@_last_row_int'~but~your~
2874   \@@_full_name_env:\ seems~to~have~\int_use:N \c@iRow \ rows.~
2875   If~you~go~on,~the~value~of~\int_use:N \c@iRow \ will~be~used~for~
2876   last~row.~You~can~avoid~this~problem~by~using~'last-row'~
2877   without~value~(more~compilations~might~be~necessary).
2878 }
2879 \@@_msg_new:nn { Yet-in-env }
2880 {
2881   Environments~\{NiceArray\}~(or~\{NiceMatrix\},~etc.)~can't~be~nested.\\
2882   This~error~is~fatal.
2883 }

```

```

2884 \@@_msg_new:nn { Outside-math-mode }
2885 {
2886   The~\@@_full_name_env:\ can~be~used~only~in~math~mode~
2887   (and~not~in~\token_to_str:N \vcenter).\
2888   This~error~is~fatal.
2889 }
2890 \@@_msg_new:nn { Bad~value~for~letter~for~dotted~lines }
2891 {
2892   The~value~of~key~'\tl_use:N\l_keys_key_str'~must~be~of~length~1.\
2893   If~you~go~on,~it~will~be~ignored.
2894 }
2895 \@@_msg_new:nnn { Unknown~key~for~NiceMatrixOptions }
2896 {
2897   The~key~'\tl_use:N\l_keys_key_str'~is~unknown~for~the~command~
2898   \token_to_str:N \NiceMatrixOptions. \
2899   If~you~go~on,~it~will~be~ignored. \
2900   For~a~list~of~the~available~keys,~type~H~<return>.
2901 }
2902 {
2903   The~available~options~are~(in~alphabetic~order):~
2904   allow~duplicate~names,~
2905   code~for~first~col,~
2906   code~for~first~row,~
2907   code~for~last~col,~
2908   code~for~last~row,~
2909   create~extra~nodes,~
2910   create~medium~nodes,~
2911   create~large~nodes,~
2912   end~of~row,~
2913   exterior~arraycolsep,~
2914   hlines,~
2915   hvlines,~
2916   left~margin,~
2917   letter~for~dotted~lines,~
2918   light~syntax,~
2919   nullify~dots,~
2920   parallelize~diags,~
2921   renew~dots,~
2922   renew~matrix,~
2923   right~margin,~
2924   small,~
2925   transparent,~
2926   vlines,~
2927   xdots/color,~
2928   xdots/shorten~and~
2929   xdots/line~style.
2930 }
2931 \@@_msg_new:nnn { Unknown~option~for~NiceArray }
2932 {
2933   The~option~'\tl_use:N\l_keys_key_str'~is~unknown~for~the~environment~
2934   \{NiceArray\}. \
2935   If~you~go~on,~it~will~be~ignored. \
2936   For~a~list~of~the~available~options,~type~H~<return>.
2937 }
2938 {
2939   The~available~options~are~(in~alphabetic~order):~
2940   b,~
2941   baseline,~
2942   c,~
2943   code~after,~
2944   code~for~first~col,~
2945   code~for~first~row,~
2946   code~for~last~col,~

```

```

2947 code-for-last-row,~
2948 columns-width,~
2949 create-extra-nodes,~
2950 create-medium-nodes,~
2951 create-large-nodes,~
2952 end-of-row,~
2953 extra-left-margin,~
2954 extra-right-margin,~
2955 first-col,~
2956 first-row,~
2957 hlines,~
2958 hvlines,~
2959 last-col,~
2960 last-row,~
2961 left-margin,~
2962 light-syntax,~
2963 name,~
2964 nullify-dots,~
2965 parallelize-diags,~
2966 renew-dots,~
2967 right-margin,~
2968 small,~
2969 t,~
2970 vlines,~
2971 xdots/color,~
2972 xdots/shorten-and~
2973 xdots/line-style.
2974 }

```

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`).

```

2975 \@@_msg_new:nnn { Unknown~option~for~NiceMatrix }
2976 {
2977   The~option~'\tl_use:N\l_keys_key_str'~is~unknown~for~the~
2978   \@@_full_name_env:. \\\
2979   If~you~go~on,~it~will~be~ignored. \\\
2980   For~a~list~of~the~available~options,~type~H~<return>.
2981 }
2982 {
2983   The~available~options~are~(in~alphabetic~order):~
2984   code-after,~
2985   code-for-first-col,~
2986   code-for-first-row,~
2987   code-for-last-col,~
2988   code-for-last-row,~
2989   columns-width,~
2990   create-extra-nodes,~
2991   create-medium-nodes,~
2992   create-large-nodes,~
2993   end-of-row,~
2994   extra-left-margin,~
2995   extra-right-margin,~
2996   first-col,~
2997   first-row,~
2998   hlines,~
2999   hvlines,~
3000   l~(=L),~
3001   last-col,~
3002   last-row,~
3003   left-margin,~
3004   light-syntax,~
3005   name,~
3006   nullify-dots,~
3007   parallelize-diags,~

```

```

3008   r~(=R),~
3009   renew-dots,~
3010   right-margin,~
3011   small,~
3012   vl原因,~
3013   xdots/color,~
3014   xdots/shorten-and~
3015   xdots/line-style.
3016 }

3017 \@@_msg_new:nnn { Duplicate-name }
3018 {
3019   The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
3020   the~same~environment~name~twice.~You~can~go~on,~but,~
3021   maybe,~you~will~have~incorrect~results~especially~
3022   if~you~use~'columns-width=auto'.~If~you~don't~want~to~see~this~
3023   message~again,~use~the~option~'allow-duplicate-names'.\\
3024   For~a~list~of~the~names~already~used,~type~H~<return>. \\
3025 }
3026 {
3027   The~names~already~defined~in~this~document~are:~
3028   \seq_use:Nnnn \g_@@_names_seq { ,~ } { ,~ } { ~and~ }.
3029 }

3030 \@@_msg_new:nn { Option-auto-for-columns-width }
3031 {
3032   You~can't~give~the~value~'auto'~to~the~option~'columns-width'~here.~
3033   If~you~go~on,~the~option~will~be~ignored.
3034 }

3035 \@@_msg_new:nn { Zero~row }
3036 {
3037   There~is~a~problem.~Maybe~you~have~used~l,~c~and~r~instead~of~L,~C~
3038   and~R~in~the~preamble~of~your~environment. \\
3039   This~error~is~fatal.
3040 }

```

## Obsolete environments

The following environments are loaded only when the package `nicematrix` has been loaded with the option `obsolete-environments`. However, they will be completely deleted in a future version.

```

3041 \bool_if:NT \c_@@_obsolete_environments_bool
3042 {
3043   \NewDocumentEnvironment { pNiceArrayC } { }
3044   {
3045     \int_zero:N \l_@@_last_col_int
3046     \pNiceArray
3047   }
3048   { \endpNiceArray }
3049   \NewDocumentEnvironment { bNiceArrayC } { }
3050   {
3051     \int_zero:N \l_@@_last_col_int
3052     \bNiceArray
3053   }
3054   { \endbNiceArray }
3055   \NewDocumentEnvironment { BNiceArrayC } { }
3056   {
3057     \int_zero:N \l_@@_last_col_int
3058     \BNiceArray
3059   }
3060   { \endBNiceArray }
3061   \NewDocumentEnvironment { vNiceArrayC } { }
3062   {

```

```

3063     \int_zero:N \l_@@_last_col_int
3064     \vNiceArray
3065   }
3066   { \endvNiceArray }
3067 \NewDocumentEnvironment { VNiceArrayC } { }
3068 {
3069     \int_zero:N \l_@@_last_col_int
3070     \VNiceArray
3071   }
3072   { \endVNiceArray }
3073 \NewDocumentEnvironment { pNiceArrayRC } { }
3074 {
3075     \int_zero:N \l_@@_last_col_int
3076     \int_zero:N \l_@@_first_row_int
3077     \pNiceArray
3078   }
3079   { \endpNiceArray }
3080 \NewDocumentEnvironment { bNiceArrayRC } { }
3081 {
3082     \int_zero:N \l_@@_last_col_int
3083     \int_zero:N \l_@@_first_row_int
3084     \bNiceArray
3085   }
3086   { \endbNiceArray }
3087 \NewDocumentEnvironment { BNiceArrayRC } { }
3088 {
3089     \int_zero:N \l_@@_last_col_int
3090     \int_zero:N \l_@@_first_row_int
3091     \BNiceArray
3092   }
3093   { \endBNiceArray }
3094 \NewDocumentEnvironment { vNiceArrayRC } { }
3095 {
3096     \int_zero:N \l_@@_last_col_int
3097     \int_zero:N \l_@@_first_row_int
3098     \vNiceArray
3099   }
3100   { \endvNiceArray }
3101 \NewDocumentEnvironment { VNiceArrayRC } { }
3102 {
3103     \int_zero:N \l_@@_last_col_int
3104     \int_zero:N \l_@@_first_row_int
3105     \VNiceArray
3106   }
3107   { \endVNiceArray }
3108 \NewDocumentEnvironment { NiceArrayCwithDelims } { }
3109 {
3110     \int_zero:N \l_@@_last_col_int
3111     \NiceArrayWithDelims
3112   }
3113   { \endNiceArrayWithDelims }
3114 \NewDocumentEnvironment { NiceArrayRCwithDelims } { }
3115 {
3116     \int_zero:N \l_@@_last_col_int
3117     \int_zero:N \l_@@_first_row_int
3118     \NiceArrayWithDelims
3119   }
3120   { \endNiceArrayWithDelims }
3121 }

```



## 15 History

### Changes between versions 1.0 and 1.1

The dotted lines are no longer drawn with Tikz nodes but with Tikz circles (for efficiency).  
Modification of the code which is now twice faster.

### Changes between versions 1.1 and 1.2

New environment `{NiceArray}` with column types `L`, `C` and `R`.

### Changes between version 1.2 and 1.3

New environment `{pNiceArrayC}` and its variants.  
Correction of a bug in the definition of `{BNiceMatrix}`, `{vNiceMatrix}` and `{VNiceMatrix}` (in fact, it was a typo).  
Options are now available locally in `{pNiceMatrix}` and its variants.  
The names of the options are changed. The old names were names in “camel style”.

### Changes between version 1.3 and 1.4

The column types `w` and `W` can now be used in the environments `{NiceArray}`, `{pNiceArrayC}` and its variants with the same meaning as in the package `array`.  
New option `columns-width` to fix the same width for all the columns of the array.

### Changes between version 1.4 and 2.0

The versions 1.0 to 1.4 of `nicematrix` were focused on the continuous dotted lines whereas the version 2.0 of `nicematrix` provides different features to improve the typesetting of mathematical matrices.

### Changes between version 2.0 and 2.1

New implementation of the environment `{pNiceArrayRC}`. With this new implementation, there is no restriction on the width of the columns.  
The package `nicematrix` no longer loads `mathtools` but only `amsmath`.  
Creation of “medium nodes” and “large nodes”.

### Changes between version 2.1 and 2.1.1

Small corrections: for example, the option `code-for-first-row` is now available in the command `\NiceMatrixOptions`.  
Following a discussion on TeX StackExchange<sup>40</sup>, Tikz externalization is now deactivated in the environments of the extension `nicematrix`.<sup>41</sup>

### Changes between version 2.1 and 2.1.2

Option `draft`: with this option, the dotted lines are not drawn (quicker).

---

<sup>40</sup>cf. [tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package](https://tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package)

<sup>41</sup>Before this version, there was an error when using `nicematrix` with Tikz externalization. In any case, it's not possible to externalize the Tikz elements constructed by `nicematrix` because they use the options `overlay` and `remember picture`.

## Changes between version 2.1.2 and 2.1.3

When searching the end of a dotted line from a command like `\Cdots` issued in the “main matrix” (not in the exterior column), the cells in the exterior column are considered as outside the matrix. That means that it’s possible to do the following matrix with only a `\Cdots` command (and a single `\Vdots`).

$$\begin{pmatrix} & C_j & \\ 0 & \vdots & 0 \\ & a & \cdots \\ 0 & & 0 \end{pmatrix}^{L_i}$$

## Changes between version 2.1.3 and 2.1.4

Replacement of some options `0 { }` in commands and environments defined with `xparse` by `! 0 { }` (because a recent version of `xparse` introduced the specifier `!` and modified the default behaviour of the last optional arguments).

See [www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end](http://www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end)

## Changes between version 2.1.4 and 2.1.5

Compatibility with the classes `revtex4-1` and `revtex4-2`.

Option `allow-duplicate-names`.

## Changes between version 2.1.5 and 2.2

Possibility to draw horizontal dotted lines to separate rows with the command `\hdottedline` (similar to the classical command `\hline` and the command `\hdashline` of `arydshln`).

Possibility to draw vertical dotted lines to separate columns with the specifier “:” in the preamble (similar to the classical specifier “|” and the specifier “:” of `arydshln`).

## Changes between version 2.2 and 2.2.1

Improvement of the vertical dotted lines drawn by the specifier “:” in the preamble.

Modification of the position of the dotted lines drawn by `\hdottedline`.

## Changes between version 2.2.1 and 2.3

Compatibility with the column type `S` of `siunitx`.

Option `hlines`.

A warning is issued when the `draft` mode is used. In this case, the dotted lines are not drawn.

## Changes between version 2.3 and 3.0

Modification of `\Hdotsfor`. Now `\Hdotsfor` erases the `\vlines` (of “|”) as `\hdotsfor` does.

Composition of exterior rows and columns on the four sides of the matrix (and not only on two sides) with the options `first-row`, `last-row`, `first-col` and `last-col`.

## Changes between version 3.0 and 3.1

Command `\Block` to draw block matrices.

Error message when the user gives an incorrect value for `last-row`.

A dotted line can no longer cross another dotted line (excepted the dotted lines drawn by `\cdottedline`, the symbol “:” (in the preamble of the array) and `\line in code-after`).

The starred versions of `\Cdots`, `\Ldots`, etc. are now deprecated because, with the new implementation, they become pointless. These starred versions are no longer documented.

The vertical rules in the matrices (drawn by “|”) are now compatible with the color fixed by `colortbl`.

Correction of a bug: it was not possible to use the colon “:” in the preamble of an array when `pdflatex` was used with `french-babel` (because `french-babel` activates the colon in the beginning of the document).

## Changes between version 3.1 and 3.2 (and 3.2a)

Option `small`.

## Changes between version 3.2 and 3.3

The options `first-row`, `last-row`, `first-col` and `last-col` are now available in the environments `{NiceMatrix}`, `{pNiceMatrix}`, `{bNiceMatrix}`, etc.

The option `columns-width=auto` doesn't need any more a second compilation.

The options `renew-dots`, `renew-matrix` and `transparent` are now available as package options (as said in the documentation).

The previous version of `nicematrix` was incompatible with a recent version of `expl3` (released 2019/09/30). This version is compatible.

## Changes between version 3.3 and 3.4

Following a discussion on TeX StackExchange<sup>42</sup>, optimization of Tikz externalization is disabled in the environments of `nicematrix` when the class `standalone` or the package `standalone` is used.

## Changes between version 3.4 and 3.5

Correction on a bug on the two previous versions where the `code-after` was not executed.

## Changes between version 3.5 and 3.6

LaTeX counters `iRow` and `jCol` available in the cells of the array.

Addition of `\normalbaselines` before the construction of the array: in environments like `{align}` of `amsmath` the value of `\baselineskip` is changed and if the options `first-row` and `last-row` were used in an environment of `nicematrix`, the position of the delimiters was wrong.

A warning is written in the `.log` file if an obsolete environment is used.

There is no longer artificial errors `Duplicate-name` in the environments of `amsmath`.

## Changes between version 3.6 and 3.7

The four “corners” of the matrix are correctly protected against the four codes: `code-for-first-col`, `code-for-last-col`, `code-for-first-row` and `code-for-last-row`.

New command `\pAutoNiceMatrix` and its variants (suggestion of Christophe Bal).

## 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.

---

<sup>42</sup>cf. [tex.stackexchange.com/questions/510841/nicematrix-and-tikz-external-optimize](https://tex.stackexchange.com/questions/510841/nicematrix-and-tikz-external-optimize)

## Changes between 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 3.12 and 3.13

Correction of a bug (question 60761504 on `stackoverflow`).

Best error messages when the user uses `&` or `\\` when `light-syntax` is in force.

# 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:</code> . . . . .	655, 2573
<code>\@@_Block_error:nn</code> . . . . .	2588, 2590
<code>\@@_Block_i</code> . . . . .	2574, 2575
<code>\@@_Block_ii:nnnnn</code> . . . . .	2575, 2576
<code>\@@_Block_iii:nnnnnn</code> . . . . .	2580, 2603
<code>\@@_Cdots</code> . . . . .	647, 661, 1980
<code>\g_@@_Cdots_lines_tl</code> . . . . .	728, 1406
<code>\@@_Cell:</code> . . . . .	176, 428, 636, 637, 638, 680, 694
<code>\@@_Ddots</code> . . . . .	649, 663, 1996
<code>\g_@@_Ddots_lines_tl</code> . . . . .	731, 1404
<code>\@@_Hdotsfor:</code> . . . . .	653, 666, 2029
<code>\@@_Hdotsfor:nnnn</code> . . . . .	2044, 2053
<code>\@@_Hdotsfor_i</code> . . . . .	2032, 2036, 2040
<code>\g_@@_Hdotsfor_lines_tl</code> . . . . .	733, 1402, 2042, 2783
<code>\@@_Hspace:</code> . . . . .	652, 2012
<code>\@@_Iddots</code> . . . . .	650, 664, 2004
<code>\g_@@_Iddots_lines_tl</code> . . . . .	732, 1405
<code>\@@_Ldots</code> . . . . .	646, 660, 665, 1972
<code>\g_@@_Ldots_lines_tl</code> . . . . .	729, 1407
<code>\l_@@_NiceArray_bool</code> . . . . .	80, 756, 810, 828, 839, 888, 1216, 2226, 2227
<code>\g_@@_NiceMatrixBlock_int</code> . . . . .	76, 2361, 2366, 2369, 2380
<code>\@@_OnlyMainNiceMatrix:n</code> . . . . .	657, 2180, 2198
<code>\@@_OnlyMainNiceMatrix_i:n</code> . . . . .	2183, 2190, 2193
<code>\@@_Vdots</code> . . . . .	648, 662, 1988
<code>\g_@@_Vdots_lines_tl</code> . . . . .	730, 1403
<code>\@@_actually_draw_Cdots:</code> . . . . .	1630, 1634
<code>\@@_actually_draw_Ddots:</code> . . . . .	1745, 1749
<code>\@@_actually_draw_Iddots:</code> . . . . .	1796, 1800
<code>\@@_actually_draw_Ldots:</code> . . . . .	1588, 1592, 2104
<code>\@@_actually_draw_Vdots:</code> . . . . .	1682, 1686
<code>\@@_actually_draw_line:</code> . . . . .	1888, 1892
<code>\@@_adapt_S_column:</code> . . . . .	151, 166, 742
<code>\@@_after_array:</code> . . . . .	977, 1292
<code>\@@_analyze_end:Nn</code> . . . . .	1025, 1069
<code>\@@_array:</code> . . . . .	547, 1026, 1047
<code>\l_@@_auto_columns_width_bool</code> . . . . .	243, 318, 1089, 1093, 2356

<code>\l_@@_baseline_str</code> .....	235, 236, 407, 408, 409, 410, 558, 890, 906, 909, 910, 911
<code>\@@_begin_of_NiceMatrix:nn</code> ....	1269, 1273
<code>\@@_begin_of_row:</code> .....	432, 453, 1136
<code>\l_@@_block_auto_columns_width_bool</code> ..	..... 753, 1094, 2349, 2354, 2364, 2374
<code>\@@_cdots</code> .....	640, 1985
<code>\l_@@_cell_box</code> ..	434, 477, 479, 485, 494, 498, 502, 504, 521, 597, 679, 687, 693, 702, 765, 767, 1137, 1159, 1162, 1164, 1179, 1200, 1204, 2114, 2117, 2121, 2614, 2660, 2665
<code>\@@_cell_with_light_syntax:n</code> ...	1061, 1068
<code>\g_@@_cells_seq</code> ....	1057, 1058, 1059, 1061
<code>\g_@@_code_after_tl</code> ...	115, 331, 1383, 1384
<code>\l_@@_code_for_first_col_tl</code> ....	282, 1148
<code>\l_@@_code_for_first_row_tl</code> ....	286, 441
<code>\l_@@_code_for_last_col_tl</code> ....	284, 1188
<code>\l_@@_code_for_last_row_tl</code> ....	288, 448
<code>\g_@@_col_total_int</code> .....	..... 433, 672, 879, 1111, 1112, 1178, 1296, 1299, 1304, 1311, 2397, 2407, 2441, 2531
<code>\c_@@_colortbl_loaded_bool</code> .....	..... 87, 92, 587, 612, 2203, 2337
<code>\l_@@_columns_width_dim</code> .....	..... 77, 319, 374, 1090, 1096, 2362, 2368
<code>\g_@@_com_or_env_str</code> .....	106, 107, 110
<code>\@@_computations_for_large_nodes:</code> ...	..... 2468, 2481, 2486
<code>\@@_computations_for_medium_nodes:</code> ..	..... 2388, 2457, 2467, 2478
<code>\@@_convert_to_str_seq:N</code> .....	2752, 2764
<code>\@@_create_col_nodes:</code> ....	1029, 1051, 1075
<code>\@@_create_large_nodes:</code> .....	1353, 2462
<code>\@@_create_medium_and_large_nodes:</code> ..	..... 1350, 2473
<code>\@@_create_medium_nodes:</code> .....	1351, 2452
<code>\@@_create_nodes:</code> 2459, 2470, 2480, 2483, 2527	
<code>\@@_ddots</code> .....	642, 2001
<code>\g_@@_ddots_int</code> .....	1340, 1769, 1770
<code>\@@_define_com:nnn</code> .....	..... 2721, 2729, 2730, 2731, 2732, 2733
<code>\@@_define_env:n</code> .....	... 1262, 1286, 1287, 1288, 1289, 1290, 1291
<code>\g_@@_delta_x_one_dim</code> ....	1342, 1772, 1782
<code>\g_@@_delta_x_two_dim</code> ....	1344, 1823, 1833
<code>\g_@@_delta_y_one_dim</code> ....	1343, 1774, 1782
<code>\g_@@_delta_y_two_dim</code> ....	1345, 1825, 1833
<code>\@@_double_int_eval:n</code> ....	2125, 2135, 2136
<code>\g_@@_dp_ante_last_row_dim</code> ....	456, 628
<code>\g_@@_dp_last_row_dim</code> .....	..... 456, 457, 631, 632, 766, 767, 947, 2217
<code>\g_@@_dp_row_zero_dim</code> .....	..... 476, 477, 622, 623, 901, 930, 940
<code>\c_@@_draft_bool</code> .....	.... 18, 19, 57, 532, 2034, 2140, 2268, 2315
<code>\@@_draw_Cdots:nnn</code> .....	1616
<code>\@@_draw_Ddots:nnn</code> .....	1737
<code>\@@_draw_Iddots:nnn</code> .....	1788
<code>\@@_draw_Ldots:nnn</code> .....	1574
<code>\@@_draw_Vdots:nnn</code> .....	1668
<code>\@@_draw_dotted_lines:</code> .....	1366, 1391
<code>\@@_draw_dotted_lines_i:</code> .....	1394, 1398
<code>\@@_draw_line:</code> .....	1614, 1666, 1735, 1786, 1837, 1841, 2177, 2313, 2347
<code>\@@_draw_line_ii:nn</code> .....	2156, 2161
<code>\@@_draw_line_iii:nn</code> .....	2164, 2168
<code>\@@_draw_non_standard_dotted_line:</code> ..	..... 1847, 1849
<code>\@@_draw_non_standard_dotted_line:n</code> ..	..... 1852, 1855
<code>\@@_draw_standard_dotted_line:</code> ..	1846, 1867
<code>\@@_draw_vlines:</code> .....	1367, 2200
<code>\g_@@_empty_cell_bool</code> .....	129, 496, 499, 506, 1978, 1986, 1994, 2002, 2010, 2014
<code>\@@_end_Cell:</code> 178, 489, 636, 637, 638, 684, 698	
<code>\l_@@_end_of_row_tl</code> .....	..... 253, 254, 280, 1043, 1044, 2827
<code>\c_@@_endpgfortikzpicture_tl</code> .....	..... 41, 45, 1395, 2165, 2290
<code>\@@_env:</code> .....	73, 75, 462, 467, 522, 528, 569, 574, 1084, 1086, 1104, 1106, 1120, 1125, 1378, 1468, 1532, 1551, 1553, 2067, 2085, 2149, 2151, 2171, 2174, 2410, 2412, 2420, 2534, 2543, 2561, 2644, 2650, 2651, 2652
<code>\g_@@_env_int</code> .....	72, 73, 752, 775, 778, 793, 796, 1303, 1324, 1840, 2570
<code>\@@_error:n</code> .....	25, 265, 272, 373, 382, 385, 403, 413, 415, 421, 426, 874, 918, 1975, 1983, 1991, 1999, 2007, 2592
<code>\@@_error:nn</code> .....	26, 326
<code>\@@_error:nnn</code> .....	27, 2154
<code>\@@_error_too_much_cols:</code> .....	845, 2771
<code>\@@_everycr:</code> .....	561, 617, 620
<code>\@@_everycr_i:</code> .....	561, 562
<code>\l_@@_exterior_arraycolsep_bool</code> .....	..... 237, 370, 830, 841
<code>\l_@@_extra_left_margin_dim</code> .....	..... 251, 308, 853, 1167
<code>\l_@@_extra_right_margin_dim</code> .....	..... 252, 309, 865, 1208
<code>\@@_extract_coords_values:</code> ....	2552, 2559
<code>\@@_fatal:n</code> .....	28, 84, 744, 1034, 1038, 1040, 1072, 1078, 2776, 2777, 2779
<code>\@@_fatal:nn</code> .....	29
<code>\l_@@_final_anchor_tl</code> .....	122
<code>\@@_final_cell:</code> .....	1552, 1571
<code>\l_@@_final_i_int</code> .....	..... 1356, 1419, 1424, 1427, 1448, 1456, 1460, 1469, 1477, 1553, 1608, 1761, 1812, 2058, 2086
<code>\l_@@_final_j_int</code> ....	1357, 1420, 1425, 1433, 1439, 1449, 1457, 1461, 1470, 1478, 1553, 1605, 1645, 1763, 1814, 2079, 2089, 2091
<code>\l_@@_final_open_bool</code> .....	..... 1359, 1426, 1430, 1436, 1442, 1446, 1462, 1603, 1643, 1652, 1663, 1689, 1702, 1710, 1721, 1759, 1810, 1896, 1911, 1942, 1943, 2056, 2080, 2092, 2146, 2287, 2321
<code>\l_@@_final_suffix_tl</code> .....	121
<code>\@@_find_extremities_of_line:nnnn</code> ...	..... 1414, 1578, 1620, 1673, 1741, 1792
<code>\l_@@_first_col_int</code> .....	..... 135, 136, 333, 419, 432, 823, 883, 1079, 2182, 2239, 2256, 2397, 2407, 2441, 2489, 2531, 2632, 2703, 2709, 2715

<code>\l_@@_first_row_int</code> .....	<code>\l_@@_last_col_without_value_bool</code> ...
..... 133, 134, 334, 423, 670, 898, 914,	..... 140, 395, 1297
927, 938, 2390, 2404, 2431, 2488, 2529,	<code>\l_@@_last_row_int</code> .....
2701, 2841, 3076, 3083, 3090, 3097, 3104, 3117	..... 137, 138, 335, 424, 446, 585, 713,
<code>\@@_full_name_env:</code> .....	760, 770, 777, 784, 868, 872, 875, 882, 944,
108,	1045, 1046, 1144, 1145, 1185, 1186, 1318,
2789, 2796, 2804, 2851, 2860, 2874, 2886, 2978	1583, 1625, 2115, 2188, 2196, 2215, 2713, 2873
<code>\@@_hdottedline:</code> .....	<code>\l_@@_last_row_without_value_bool</code> ...
651, 2269, 2271	..... 139, 772, 870, 1316
<code>\@@_hdottedline:n</code> .....	<code>\g_@@_last_vdotted_col_int</code> 716, 718, 724, 726
2280, 2284	<code>\@@_ldots</code> .....
<code>\@@_hdottedline_i:</code> .....	639, 1977
2274, 2277	<code>\l_@@_left_delim_dim</code> 808, 812, 817, 1016, 1165
<code>\@@_hdottedline_i:n</code> .....	<code>\l_@@_left_delim_tl</code> .....
2289, 2293	737, 2307
<code>\l_@@_hlines_bool</code> .....	<code>\l_@@_left_margin_dim</code> .....
240, 291, 579	..... 247, 302, 852, 1166, 2302, 2522
<code>\g_@@_ht_last_row_dim</code> .....	<code>\l_@@_letter_for_dotted_lines_str</code> ...
..... 458, 629, 630, 764, 765, 946, 2218	..... 381, 387, 388, 706, 707
<code>\g_@@_ht_row_one_dim</code> ....	<code>\l_@@_light_syntax_bool</code> .....
484, 485, 626, 627	..... 234, 278, 855, 860, 1319
<code>\g_@@_ht_row_zero_dim</code> .....	<code>\@@_line</code> .....
..... 478, 479, 624, 625, 901, 930, 941	1382, 2127
<code>\@@_i:</code> .....	<code>\@@_line_i:nn</code> .....
2390, 2392,	2134, 2141, 2143
2393, 2394, 2395, 2404, 2410, 2412, 2413,	<code>\@@_line_with_light_syntax:n</code> ...
2414, 2415, 2420, 2421, 2422, 2423, 2431,	1050, 1063
2434, 2436, 2437, 2438, 2490, 2492, 2495,	<code>\@@_line_with_light_syntax_i:n</code> .....
2496, 2500, 2501, 2529, 2534, 2536, 2538,	..... 1049, 1055, 1066
2542, 2543, 2554, 2561, 2563, 2565, 2569, 2570	<code>\g_@@_max_cell_width_dim</code> .....
<code>\@@_iddots</code> .....	..... 493, 494, 754, 1095, 2355, 2381
643, 2009	<code>\l_@@_max_delimiter_width_bool</code> 256, 277, 968
<code>\g_@@_iddots_int</code> .....	<code>\c_@@_max_l_dim</code> .....
1341, 1820, 1821	1886, 1891
<code>\l_@@_in_env_bool</code> .....	<code>\l_@@_medium_nodes_bool</code> 245, 298, 1347, 2647
79, 744, 745	<code>\@@_message_hdotsfor:</code> 2781, 2789, 2796, 2804
<code>\l_@@_initial_anchor_tl</code> .....	<code>\@@_msg_new:nn</code> .....
120	... 30, 55, 2786, 2793, 2801, 2808, 2813,
<code>\@@_initial_cell:</code> .....	2819, 2824, 2830, 2838, 2845, 2850, 2852,
1550, 1566	2858, 2865, 2871, 2879, 2884, 2890, 3030, 3035
<code>\l_@@_initial_i_int</code> ...	<code>\@@_msg_new:nnn</code> ..
1354, 1417, 1488,	31, 2895, 2931, 2975, 3017
1491, 1512, 1520, 1524, 1533, 1541, 1551,	<code>\@@_msg_redirect_name:nn</code> ....
1599, 1654, 1656, 1753, 1804, 2057, 2058, 2068	32, 376, 2747
<code>\l_@@_initial_j_int</code> .....	<code>\@@_multicolumn:nnn</code> .....
..... 1355, 1418, 1489, 1497, 1503,	654, 2018
1513, 1521, 1525, 1534, 1542, 1551, 1596,	<code>\g_@@_multicolumn_cells_seq</code> .....
1638, 1712, 1714, 1755, 1806, 2061, 2071, 2073	..... 668, 2023, 2415, 2423, 2548
<code>\l_@@_initial_open_bool</code> ...	<code>\g_@@_multicolumn_sizes_seq</code> 669, 2025, 2549
1358, 1490,	<code>\g_@@_name_env_str</code> .....
1494, 1500, 1506, 1510, 1526, 1594, 1636,	..... 105, 111, 112, 740, 741, 1071, 1217,
1651, 1661, 1689, 1696, 1708, 1751, 1802,	1218, 1224, 1225, 1232, 1233, 1240, 1241,
1894, 1941, 2055, 2062, 2074, 2145, 2286, 2320	1248, 1249, 1256, 1257, 1266, 1386, 2725, 2773
<code>\l_@@_initial_suffix_tl</code> .....	<code>\l_@@_name_str</code> .....
119	244, 328, 464, 468,
<code>\@@_instruction_of_type:nn</code> .....	524, 527, 571, 575, 773, 782, 785, 791, 800,
..... 533, 535, 1976, 1984, 1992, 2000, 2008	803, 1085, 1086, 1105, 1106, 1122, 1126,
<code>\l_@@_inter_dots_dim</code> .....	1306, 1310, 1327, 1331, 2539, 2542, 2566, 2569
.. 99, 100, 1363, 1899, 1906, 1917, 1925,	<code>\g_@@_names_seq</code> .....
1932, 1937, 1949, 1957, 2308, 2311, 2343, 2345	78, 325, 327, 3028
<code>\g_@@_internal_code_after_tl</code> .....	<code>\l_@@_nb_cols_int</code> .....
..... 114, 719, 1368, 1369, 2279, 2578	..... 2692, 2697, 2700, 2704, 2710, 2716
<code>\@@_j:</code> .....	<code>\l_@@_nb_rows_int</code> .....
2397, 2399,	2691, 2696, 2707
2400, 2401, 2402, 2407, 2410, 2412, 2415,	<code>\@@_node_for_multicolumn:nn</code> ....
2417, 2418, 2420, 2423, 2425, 2426, 2441,	2550, 2557
2444, 2446, 2447, 2448, 2503, 2505, 2508,	<code>\@@_node_for_the_cell:</code> 503, 508, 1163, 1209
2510, 2514, 2515, 2531, 2534, 2535, 2537,	<code>\l_@@_nullify_dots_bool</code> .....
2542, 2543, 2555, 2561, 2562, 2564, 2569, 2570	..... 242, 297, 1977, 1985, 1993, 2001, 2009
<code>\l_@@_l_dim</code> .....	<code>\c_@@_obsolete_environments_bool</code> ....
..... 1872, 1873, 1886, 1887, 1899, 1905,	..... 2734, 2743, 3041
1916, 1924, 1932, 1937, 1949, 1950, 1957, 1958	<code>\@@_old_multicolumn</code> .....
<code>\l_@@_large_nodes_bool</code> 246, 299, 1349, 1353	2017, 2020
<code>\g_@@_last_col_found_bool</code> .....	<code>\l_@@_parallelize_diags_bool</code> .....
..... 143, 727, 880, 972, 1110, 1176, 1295	..... 238, 239, 294, 1338, 1767, 1818
<code>\l_@@_last_col_int</code> .....	<code>\@@_pgf_rect_node:nnn</code> .....
141,	210, 2649
142, 396, 398, 414, 422, 789, 795, 802, 834,	<code>\@@_pgf_rect_node:nnnnn</code> 185, 2533, 2560, 2643
1279, 1281, 1296, 1299, 1678, 2705, 2711,	
2717, 2775, 2790, 3045, 3051, 3057, 3063,	
3069, 3075, 3082, 3089, 3096, 3103, 3110, 3116	

<code>\c_@@pgfortikzpicture_tl</code> .....	<code>\c_@@table_print_tl</code> .....
..... 40, 44, 1393, 2163, 2288	164, 165, 178
<code>\@@_pre_array:</code> .....	<code>\@@_test_if_math_mode:</code> .....
..... 595, 807	..... 81, 743, 1226, 1234, 1242, 1250, 1258
<code>\c_@@preamble_first_col_tl</code> .....	<code>\l_@@_the_array_box</code> .....
<code>\c_@@preamble_last_col_tl</code> .....	..... 821, 846, 903, 907, 933, 960
<code>\@@_pred:n</code> .....	<code>\c_@@tikz_loaded_bool</code> ..
..... 184, 1281	34, 39, 1370, 2322
<code>\@@_put_box_in_flow:</code> .....	<code>\l_@@tikz_tl</code> .....
..... 970, 979, 1018	2597, 2642
<code>\@@_put_box_in_flow_bis:nn</code> .....	<code>\l_@@_type_of_col_tl</code> .....
..... 969, 985	..... 399, 400, 401, 402, 1267, 1269
<code>\@@_qpoint:</code> .....	<code>\c_@@_types_of_matrix_seq</code> .....
..... 74, 893, 895, 922, 924,	..... 2766, 2773
1596, 1599, 1605, 1608, 1638, 1645, 1654,	<code>\@@_update_for_first_and_last_row:</code> ..
1656, 1698, 1704, 1712, 1714, 1753, 1755,	..... 472, 495, 762, 1157, 1198
1761, 1763, 1804, 1806, 1812, 1814, 2171,	<code>\@@_vdots</code> .....
2174, 2208, 2211, 2219, 2221, 2232, 2249,	..... 641, 1993
2296, 2300, 2303, 2339, 2342, 2344, 2436,	<code>\@@_vdottedline:n</code> .....
2446, 2618, 2620, 2622, 2624, 2656, 2657, 2663	..... 720, 2316, 2318
<code>\l_@@_radius_dim</code> .....	<code>\@@_vdottedline_i:n</code> .....
..... 103, 104, 714,	..... 2325, 2330, 2335
1362, 1612, 1613, 1966, 2273, 2298, 2340, 2341	<code>\@@_vline:</code> .....
<code>\l_@@_real_left_delim_dim</code> ..	..... 755, 2198
..... 987, 1002, 1017	<code>\@@_vline_i:</code> .....
<code>\l_@@_real_right_delim_dim</code> ..	..... 93, 2198, 2199
..... 988, 1014, 1020	<code>\l_@@_vlines_bool</code> .....
<code>\@@_renew_NC@rewrite@S:</code> .....	..... 241, 292, 829, 840, 847, 866, 1367
..... 169, 725	<code>\l_@@_white_bool</code> .....
<code>\l_@@_renew_dots_bool</code> .....	..... 2599, 2626
..... 295, 658, 2737	<code>\g_@@_width_first_col_dim</code> .....
<code>\@@_renew_matrix:</code> .....	..... 250, 886, 1158, 1159
..... 366, 2671, 2739	<code>\g_@@_width_last_col_dim</code> ..
<code>\@@_restore_iRow_jCol:</code> .....	249, 974, 1199, 1200
..... 1387, 1409	<code>\l_@@_x_final_dim</code> .....
<code>\c_@@_revtex_bool</code> .....	..... 125, 1561, 1606, 1607, 1646,
..... 48, 50, 53, 549	1647, 1694, 1716, 1718, 1722, 1724, 1729,
<code>\l_@@_right_delim_dim</code> .....	1731, 1764, 1773, 1781, 1815, 1824, 1832,
..... 809, 813, 819, 1019, 1206	1864, 1879, 1931, 1947, 2175, 2304, 2311, 2341
<code>\l_@@_right_delim_tl</code> .....	<code>\l_@@_x_initial_dim</code> ....
..... 738, 2310	..... 123, 1556, 1597,
<code>\l_@@_right_margin_dim</code> .....	1598, 1639, 1640, 1694, 1715, 1716, 1718,
..... 248, 304, 864, 1207, 2305, 2525	1722, 1724, 1726, 1729, 1731, 1756, 1773,
<code>\@@_rotate:</code> .....	1781, 1807, 1824, 1832, 1863, 1879, 1931,
..... 656, 2109	1945, 1947, 1965, 1967, 2172, 2301, 2308, 2340
<code>\@@_rotate_i:</code> .....	<code>\l_@@_xdots_color_tl</code> .....
..... 2109, 2110	..... 255, 268,
<code>\@@_rotate_ii:</code> .....	1587, 1629, 1670, 1744, 1795, 1853, 2103, 2131
..... 2110, 2111	<code>\l_@@_xdots_line_style_tl</code> .....
<code>\@@_rotate_iii:</code> .....	..... 130, 132, 264, 1845, 1853, 2312, 2346
..... 2111, 2112	<code>\l_@@_xdots_shorten_dim</code> .....
<code>\g_@@_row_of_col_done_bool</code> .....	..... 101,
..... 118, 583, 739, 1081	102, 270, 1364, 1860, 1861, 1905, 1916, 1924
<code>\g_@@_row_total_int</code> .....	<code>\l_@@_y_final_dim</code> .....
..... 671, 881, 915, 1318, 1325,	..... 126, 1562, 1609, 1613, 1658, 1662,
1332, 2099, 2221, 2390, 2404, 2431, 2529, 2842	1664, 1705, 1762, 1775, 1778, 1813, 1826,
<code>\g_@@_rows_seq</code> ..	1829, 1864, 1881, 1936, 1955, 2176, 2299, 2345
..... 1042, 1044, 1046, 1048, 1050	<code>\l_@@_y_initial_dim</code> .....
<code>\l_@@_save_iRow_int</code> .....	..... 124, 1557, 1600, 1612, 1657,
..... 116, 599, 1411	1658, 1662, 1664, 1699, 1754, 1775, 1780,
<code>\l_@@_save_jCol_int</code> .....	1805, 1826, 1831, 1863, 1881, 1936, 1953,
..... 117, 602, 1412	1955, 1965, 1968, 2173, 2297, 2298, 2299, 2343
<code>\@@_set_final_coords:</code> .....	<code>\l_@@_y_initial_dim</code> .....
..... 1559, 1572	..... 1039,
<code>\@@_set_final_coords_from_anchor:n</code> ..	1066, 2705, 2711, 2717, 2826, 2842, 2847,
..... 1569, 1611, 1649, 1692, 1707, 1766, 1817	2855, 2861, 2868, 2881, 2887, 2892, 2898,
<code>\@@_set_initial_coords:</code> .....	2899, 2934, 2935, 2978, 2979, 3023, 3024, 3038
..... 1554, 1567	<code>\{</code> .....
<code>\@@_set_initial_coords_from_anchor:n</code> ..	..... 112, 1243, 2733, 2854, 2881, 2934
..... 1564, 1602, 1642, 1691, 1701, 1758, 1809	<code>\}</code> .....
<code>\@@_set_seq_of_str_from_clist:Nn</code> ..	..... 112, 1243, 2733, 2854, 2881, 2934
..... 2761, 2766	<code>\l</code> .....
<code>\@@_set_size:n</code> .....	..... 1259, 2732
..... 2689, 2698	<code>\sqcup</code> .....
<code>\c_@@_siunitx_loaded_bool</code> ..	..... 2784, 2789, 2796, 2804,
..... 144, 148, 153, 725	2833, 2841, 2842, 2847, 2851, 2874, 2875, 2886
<code>\l_@@_small_bool</code> .....	<b>A</b>
..... 290, 436, 605, 1139, 1181, 1360	<code>\array</code> .....
<code>\@@_standard_ialign:</code> .....	..... 557
..... 560, 633	
<code>\c_@@_standard_tl</code> ..	
..... 131, 132, 1845, 2312, 2346	
<code>\l_@@_stop_loop_bool</code> .....	
..... 1421, 1422,	
1450, 1463, 1472, 1485, 1486, 1514, 1527, 1536	
<code>\@@_succ:n</code> .....	
..... 183, 569, 1120, 1125,	
1126, 1605, 1645, 1656, 1704, 1714, 1761,	
1763, 1806, 1812, 2211, 2219, 2221, 2227,	
2303, 2344, 2496, 2500, 2510, 2514, 2622, 2624	
<code>\l_@@_suffix_tl</code> .....	
..... 2458, 2469,	
2479, 2482, 2534, 2542, 2543, 2561, 2569, 2570	
<code>\c_@@_table_collect_begin_tl</code> ..	
..... 161, 163, 176	



`\arraycolsep` ..... 303, 305, 307, 608,  
                   812, 813, 849, 885, 959, 961, 975, 1097,  
                   1169, 1201, 1598, 1607, 1640, 1647, 2302, 2305  
`\arrayrulewidth` 588, 589, 849, 850, 866, 2207,  
                   2236, 2240, 2253, 2257, 2381, 2633, 2636, 2637  
`\arraystretch` ..... 607  
`\AtBeginDocument` .. 35, 88, 145, 1389, 2159, 2282  
`\AutoNiceMatrixWithDelims` ..... 2694, 2726

## B

`\begin` ..... 187, 212,  
           856, 857, 1275, 1851, 2519, 2628, 2641, 2699  
`\Block` ..... 655, 2588, 2593, 2847  
`\BNiceArray` ..... 3058, 3091  
`\bNiceArray` ..... 3052, 3084  
`\BNiceMatrix` ..... 2686  
`\bNiceMatrix` ..... 2683

bool commands:

`\bool_do_until:Nn` ..... 1422, 1486  
`\bool_gset_false:N` ..... 499, 506, 727, 739  
`\bool_gset_true:N` .....  
           1081, 1176, 1978, 1986, 1994, 2002, 2010, 2014  
`\bool_if:NTF` ..... 57,  
           153, 436, 496, 532, 549, 579, 583, 587, 605,  
           612, 658, 725, 744, 753, 756, 810, 847,  
           855, 860, 866, 870, 888, 968, 972, 1089,  
           1110, 1139, 1181, 1295, 1297, 1316, 1319,  
           1338, 1349, 1353, 1360, 1367, 1370, 1446,  
           1510, 1594, 1603, 1636, 1643, 1661, 1663,  
           1696, 1702, 1708, 1710, 1721, 1728, 1751,  
           1759, 1767, 1802, 1810, 1818, 1894, 1896,  
           1911, 1941, 1942, 1943, 1977, 1985, 1993,  
           2001, 2009, 2034, 2140, 2203, 2226, 2227,  
           2268, 2322, 2337, 2364, 2374, 2626, 2647, 3041  
`\bool_if:nTF` ..... 880, 912, 1347,  
           1974, 1982, 1990, 1998, 2006, 2147, 2315, 2607  
`\bool_lazy_all:nTF` ..... 826, 837  
`\bool_lazy_and:nnTF` ..... 1092, 1140, 1650  
`\bool_lazy_or:nnTF` ... 261, 1184, 1689, 1885  
`\bool_lazy_or_p:nn` ..... 1143  
`\bool_new:N` ..... 18, 34, 48, 79, 80, 87,  
           118, 129, 139, 140, 143, 144, 234, 237, 238,  
           240, 241, 242, 243, 245, 246, 256, 2349, 2734  
`\bool_not_p:n` ..... 829, 830, 840, 841, 1094  
`\bool_set:Nn` ..... 1693  
`\bool_set_false:N` .... 1358, 1359, 1421,  
           1426, 1485, 1490, 1688, 2055, 2056, 2145, 2146  
`\bool_set_true:N` . 19, 39, 50, 53, 92, 148,  
           239, 318, 395, 745, 772, 1216, 1430, 1436,  
           1442, 1450, 1462, 1463, 1472, 1494, 1500,  
           1506, 1514, 1526, 1527, 1536, 2062, 2074,  
           2080, 2092, 2286, 2287, 2320, 2321, 2354, 2356  
`\l_tmpa_bool` ..... 1688, 1693, 1728

box commands:

`\box_clear_new:N` ..... 597, 821  
`\box_dp:N` 457, 477, 623, 632, 767, 982, 996, 1009  
`\box_ht:N` ..... 458, 479,  
           485, 625, 627, 630, 765, 981, 996, 1009, 2120  
`\box_move_up:nn` ..... 63, 65, 67, 903, 932  
`\box_rotate:Nn` ..... 2114  
`\box_set_dp:Nn` ..... 982  
`\box_set_ht:Nn` ..... 981  
`\box_use:N` ..... 2121

`\box_use_drop:N` ..... 498, 504, 521, 687,  
           702, 903, 907, 933, 960, 983, 1164, 2660, 2665  
`\box_wd:N` ..... 494, 502,  
           817, 819, 1003, 1015, 1159, 1162, 1200, 1204  
`\l_tmpa_box` ..... 816,  
           817, 818, 819, 950, 981, 982, 983, 996, 1009  
`\l_tmpb_box` ..... 989, 1003, 1004, 1015

## C

`\Cdots` ..... 647, 2832  
`\cdots` ..... 640, 661  
`\color` 1581, 1584, 1587, 1623, 1626, 1629, 1670,  
           1676, 1679, 1744, 1795, 2097, 2100, 2103, 2131  
`\colorlet` ..... 97, 98, 442, 449, 1149, 1189  
`\cr` ..... 1130  
`\crrcr` ..... 1077

cs commands:

`\cs_generate_variant:Nn` ..... 2386, 2387  
`\cs_gset:Npn` ... 1303, 1310, 1324, 1331, 2379  
`\cs_gset_eq:NN` ..... 166, 616  
`\cs_if_exist:NTF` ..... 16, 598, 601,  
           746, 749, 775, 782, 793, 800, 1411, 1412,  
           1453, 1466, 1517, 1530, 2065, 2083, 2366, 2409  
`\cs_if_exist_p:N` ..... 262  
`\cs_if_free:NTF` 1576, 1618, 1671, 1739, 1790  
`\cs_if_free_p:N` ..... 2149, 2151  
`\cs_new:Nn` ..... 1550, 1552  
`\cs_new:Npn` .. 73, 108, 183, 184, 561, 1075,  
           2018, 2029, 2125, 2269, 2271, 2575, 2590, 2781  
`\cs_new_protected:Npn` ..... 25,  
           26, 27, 28, 29, 30, 31, 32, 74, 81, 169,  
           185, 210, 428, 453, 472, 489, 508, 535, 547,  
           562, 595, 979, 985, 1055, 1063, 1068, 1069,  
           1262, 1273, 1292, 1398, 1409, 1414, 1554,  
           1559, 1564, 1569, 1574, 1592, 1616, 1634,  
           1668, 1686, 1737, 1749, 1788, 1800, 1841,  
           1849, 1855, 1867, 1892, 2012, 2053, 2109,  
           2110, 2111, 2112, 2141, 2143, 2168, 2180,  
           2193, 2198, 2200, 2277, 2293, 2316, 2318,  
           2335, 2388, 2452, 2462, 2473, 2486, 2527,  
           2552, 2557, 2576, 2603, 2689, 2752, 2761, 2771  
`\cs_new_protected:Npx` .... 1391, 2161, 2284  
`\cs_set:Npn` ..... 554,  
           607, 610, 1416, 1474, 1538, 2107, 2554, 2555  
`\cs_set_eq:NN` ..... 157,  
           551, 552, 553, 560, 633, 639, 640, 641,  
           642, 643, 644, 645, 646, 647, 648, 649,  
           650, 651, 652, 653, 654, 655, 656, 657,  
           660, 661, 662, 663, 664, 665, 666, 673, 674,  
           675, 700, 1382, 2017, 2179, 2199, 2588, 2593  
`\cs_set_protected:Npn` .....  
           ..... 93, 151, 533, 755, 2671, 2721, 2723

## D

`\Ddots` ..... 649, 2833  
`\ddots` ..... 642, 663  
`\DeclareOption` ..... 19, 20  
dim commands:  
`\dim_abs:n` ..... 200, 203, 227, 228, 1886  
`\dim_add:Nn` .... 849, 941, 947, 1598, 1612,  
           1613, 1640, 1967, 1968, 2218, 2220, 2523, 2633  
`\dim_compare:nNnTF` .....  
           ..... 502, 1090, 1162, 1204, 1724, 2433, 2443  
`\dim_compare_p:nNn` ..... 1694, 1886, 1887



<code>\dim_const:Nn</code> .....	1891	<code>\endBNiceMatrix</code> .....	2687
<code>\dim_eval:n</code> .....	2381	<code>\endbNiceMatrix</code> .....	2684
<code>\dim_gadd:Nn</code> .....	900, 929, 1945, 1953, 2308	<code>\endNiceArrayWithDelims</code> .....	1221, 1229, 1237, 1245, 1253, 1261, 3113, 3120
<code>\dim_gset:Nn</code> .....	457, 458, 476, 478, 484, 493, 623, 625, 627, 630, 632, 764, 766, 812, 813, 1158, 1199, 1772, 1774, 1823, 1825	<code>\endpgfpicture</code> .....	45, 470, 530, 577, 897, 926, 1087, 1107, 1128, 2265, 2331, 2460, 2471, 2484, 2667
<code>\dim_gset_eq:NN</code> .....	456, 894, 923	<code>\endpNiceArray</code> .....	3048, 3079
<code>\dim_gsub:Nn</code> .....	896, 925, 2311	<code>\endpNiceMatrix</code> .....	2675
<code>\dim_gzero_new:N</code> .....	622, 624, 626, 628, 629, 631, 754, 1342, 1343, 1344, 1345, 2355	<code>\endtiktzpicture</code> .....	41, 2326
<code>\dim_max:nn</code> .....	477, 479, 485, 494, 765, 767, 1159, 1200, 1728, 2387, 2422, 2426	<code>\endVNiceArray</code> .....	3072, 3107
<code>\dim_min:nn</code> .....	1728, 2386, 2414, 2418	<code>\endvNiceArray</code> .....	3066, 3100
<code>\dim_new:N</code> ...	77, 99, 101, 103, 123, 124, 125, 126, 127, 128, 247, 248, 249, 250, 251, 252	<code>\endVNiceMatrix</code> .....	2681
<code>\dim_ratio:nn</code> .....	1782, 1833, 1899, 1904, 1915, 1923, 1932, 1937, 1948, 1956	<code>\endvNiceMatrix</code> .....	2678
<code>\dim_set:Nn</code> .....	100, 102, 104, 319, 374, 608, 817, 819, 1002, 1014, 1362, 1363, 1364, 1657, 1715, 1726, 1778, 1829, 1873, 1929, 1934, 2301, 2304, 2340, 2341, 2343, 2345, 2368, 2395, 2402, 2413, 2417, 2421, 2425, 2437, 2438, 2447, 2448, 2492, 2505	<code>\everycr</code> .....	620
<code>\dim_set_eq:NN</code> .....	940, 946, 1299, 1318, 1556, 1557, 1561, 1562, 1597, 1600, 1606, 1609, 1639, 1646, 1655, 1658, 1662, 1664, 1699, 1705, 1713, 1716, 1718, 1722, 1731, 1754, 1756, 1762, 1764, 1805, 1807, 1813, 1815, 2172, 2173, 2175, 2176, 2209, 2217, 2297, 2299, 2393, 2400, 2500, 2514, 2619, 2621, 2623, 2625	exp commands:	
<code>\dim_sub:Nn</code> .....	1607, 1647, 2212, 2222, 2223, 2298, 2518, 2520	<code>\exp_after:wN</code> .....	173
<code>\dim_use:N</code> .....	2434, 2444, 2495, 2496, 2508, 2509, 2535, 2536, 2537, 2538, 2562, 2563, 2564, 2565	<code>\exp_args:Nnc</code> .....	2368
<code>\dim_zero:N</code> .....	943, 949, 2214, 2362	<code>\exp_args:NNV</code> .....	1044
<code>\dim_zero_new:N</code> .....	808, 809, 987, 988, 1872, 2392, 2394, 2399, 2401	<code>\exp_args:Nnx</code> .....	1269
<code>\c_max_dim</code> .....	2393, 2395, 2400, 2402, 2434, 2444	<code>\exp_args:No</code> .....	1852
<code>\g_tmpa_dim</code> .....	894, 896, 900, 903, 923, 925, 929, 932	<code>\exp_args:Nv</code> .....	707, 1026, 1047, 1049
<code>\l_tmpa_dim</code> .....	221, 228, 940, 941, 943, 956, 981, 1655, 1657, 1713, 1715, 1929, 1967, 2209, 2212, 2220, 2222, 2223, 2260, 2619, 2636, 2645, 2656, 2658	<code>\exp_args:Nx</code> .....	2642
<code>\l_tmpb_dim</code> .....	221, 227, 946, 947, 949, 963, 982, 1934, 1968, 2214, 2217, 2218, 2223, 2243, 2260, 2621, 2633, 2636, 2645	<code>\exp_not:N</code> .....	40, 41, 44, 45, 2286, 2287
<code>\l_tmpc_dim</code> .....	127, 2623, 2637, 2645	<code>\exp_not:n</code> .....	2586
<code>\l_tmpd_dim</code> .....	128, 2625, 2637, 2645	<code>\ExplSyntaxOff</code> .....	1314, 1335, 2383
<code>\c_zero_dim</code> .....	190, 191, 192, 215, 216, 217, 502, 511, 515, 516, 1090, 1162, 1204, 1887	<code>\ExplSyntaxOn</code> .....	1300, 1321, 2376
<code>\dots</code> .....	665		
<code>\draw</code> .....	1857		
<b>E</b>			
else commands:		<b>F</b>	
<code>\else:</code> .....	83	fi commands:	
<code>\end</code> .....	208, 232, 861, 862, 1024, 1073, 1271, 1865, 2517, 2639, 2646, 2719	<code>\fi:</code> .....	85
<code>\endarray</code> .....	1030, 1052	<code>\firsthline</code> .....	644
<code>\endBNiceArray</code> .....	3060, 3093	fp commands:	
<code>\endbNiceArray</code> .....	3054, 3086	<code>\fp_to_dim:n</code> .....	1875
		<b>G</b>	
		group commands:	
		<code>\group_begin:</code> ....	155, 1294, 1579, 1621, 1674, 1742, 1793, 1871, 2095, 2129, 2202, 2605
		<code>\group_end:</code> .....	160, 1385, 1589, 1631, 1683, 1746, 1797, 1889, 2105, 2138, 2266, 2669
		<code>\group_insert_after:N</code> ....	2109, 2110, 2111
		<b>H</b>	
		<code>\halign</code> .....	634
		<code>\hbox</code> .....	565, 957, 1100
		hbox commands:	
		<code>\hbox:n</code> .....	63, 65, 68
		<code>\hbox_overlap_left:n</code> .....	1160
		<code>\hbox_overlap_right:n</code> .....	1202
		<code>\hbox_set:Nn</code> ..	816, 818, 950, 989, 1004, 2614
		<code>\hbox_set:Nw</code> ..	434, 679, 693, 846, 1137, 1179
		<code>\hbox_set_end:</code> ..	492, 685, 699, 867, 1156, 1197
		<code>\hbox_to_wd:nn</code> .....	203, 228
		<code>\Hdotsfor</code> .....	653, 2784
		<code>\hdotsfor</code> .....	666
		<code>\hdottedline</code> .....	651
		<code>\hfil</code> .....	700
		<code>\hline</code> .....	644, 645
		<code>\hrule</code> .....	588, 589
		<code>\Hspace</code> .....	652
		<code>\hspace</code> .....	2015
		<code>\hss</code> .....	700

**I**

`\ialign` ..... 560, 610, 633  
`\iddots` ..... 650, 2833  
`\iddots` ..... 58, 643, 664  
if commands:  
    `\if_mode_math:` ..... 83  
`\ifstandalone` ..... 749  
int commands:  
    `\int_add:Nn` 1424, 1425, 1512, 1513, 1524, 1525  
    `\int_compare:nNnTF` ..... 431, 432, 437,  
    439, 446, 474, 482, 581, 585, 711, 713, 716,  
    760, 770, 789, 823, 834, 868, 872, 882, 883,  
    898, 927, 938, 944, 1045, 1078, 1079, 1182,  
    1279, 1427, 1429, 1433, 1435, 1439, 1441,  
    1491, 1493, 1497, 1499, 1503, 1505, 1580,  
    1583, 1622, 1625, 1675, 1678, 1770, 1821,  
    2021, 2059, 2077, 2096, 2099, 2115, 2182,  
    2185, 2187, 2188, 2195, 2196, 2215, 2237,  
    2239, 2254, 2256, 2630, 2632, 2654, 2701,  
    2703, 2705, 2709, 2711, 2713, 2715, 2717, 2775  
    `\int_compare:nTF` ..... 380  
    `\int_compare_p:nNn` ..... 914,  
    915, 1141, 1144, 1145, 1185, 1186, 2609, 2610  
    `\int_eval:n` ..... 2024, 2069,  
    2087, 2126, 2339, 2564, 2583, 2584, 2790, 2805  
    `\int_gadd:Nn` ..... 2027  
    `\int_gdecr:N` ..... 880, 882  
    `\int_gincr:N` .....  
    . 430, 455, 752, 1116, 1177, 1769, 1820, 2361  
    `\int_gset:Nn` ..... 433, 670, 726, 1109  
    `\int_gset_eq:NN` .....  
    ..... 718, 875, 879, 881, 1178, 1411, 1412  
    `\int_gzero:N` ..... 564  
    `\int_gzero_new:N` .....  
    ..... 600, 603, 671, 672, 724, 1340, 1341  
    `\int_max:nn` ..... 433  
    `\int_new:N` 72, 76, 116, 117, 133, 135, 137, 141  
    `\int_set:Nn` ..... 134, 136,  
    138, 142, 396, 398, 777, 784, 795, 802, 911,  
    919, 1046, 1417, 1418, 1419, 1420, 1898,  
    1902, 1913, 1921, 1939, 2057, 2061, 2071,  
    2073, 2079, 2089, 2091, 2488, 2489, 2691, 2692  
    `\int_set_eq:NN` ..... 599, 602, 1296, 2058  
    `\int_step_inline:nnn` ..... 1962, 2106, 2225  
    `\int_step_variable:nNn` ..... 2490, 2503  
    `\int_step_variable:nnNn` .....  
    2390, 2397, 2404, 2406, 2431, 2441, 2529, 2531  
    `\int_sub:Nn` 1448, 1449, 1460, 1461, 1488, 1489  
    `\int_use:N` ..... 73, 462,  
    467, 468, 522, 527, 528, 541, 542, 574, 575,  
    720, 775, 778, 793, 796, 895, 924, 1303,  
    1304, 1311, 1324, 1325, 1332, 1456, 1457,  
    1469, 1470, 1477, 1478, 1520, 1521, 1533,  
    1534, 1541, 1542, 1551, 1553, 1596, 1599,  
    1608, 1638, 1654, 1712, 1753, 1755, 1804,  
    1814, 1840, 2024, 2045, 2046, 2068, 2086,  
    2280, 2366, 2369, 2380, 2524, 2570, 2581,  
    2582, 2798, 2840, 2841, 2842, 2873, 2874, 2875  
    `\int_zero:N` ..... 333,  
    334, 414, 419, 422, 423, 3045, 3051, 3057,  
    3063, 3069, 3075, 3076, 3082, 3083, 3089,  
    3090, 3096, 3097, 3103, 3104, 3110, 3116, 3117

`\int_zero_new:N` .....  
    ..... 1354, 1355, 1356, 1357, 2696, 2697  
`\g_tmpa_int` .... 1109, 1116, 1120, 1125, 1126  
`\l_tmpa_int` .... 911, 914, 915, 919, 924,  
    1898, 1902, 1913, 1921, 1949, 1957, 1962, 2840  
`\l_tmpb_int` ..... 1939, 1951, 1959  
iow commands:  
    `\iow_shipout:Nn` ..... 1300, 1301, 1308,  
    1314, 1321, 1322, 1329, 1335, 2376, 2377, 2383

## K

`\kern` ..... 68  
keys commands:  
    `\keys_define:nn` ..... 257, 274,  
    314, 338, 364, 391, 405, 417, 2350, 2595, 2735  
    `\l_keys_key_str` .....  
    ..... 16, 2816, 2892, 2897, 2933, 2977  
    `\keys_set:nn` 276, 390, 757, 758, 1268, 1586,  
    1628, 1681, 1743, 1794, 2102, 2130, 2363, 2606  
    `\l_keys_value_tl` ..... 2863, 3019

## L

`\lasthline` ..... 645  
`\ldots` ..... 646, 2832  
`\ldots` ..... 639, 660  
`\left` ..... 953, 992, 1007  
legacy commands:  
    `\legacy_if:nTF` ..... 322  
`\line` ..... 1382, 2854

## M

`\makebox` ..... 686, 701  
math commands:  
    `\c_math_toggle_token` .....  
    ..... 435, 491, 854, 863, 952, 966,  
    991, 1000, 1006, 1012, 1138, 1155, 1180, 1196  
`\mathinner` ..... 60  
msg commands:  
    `\msg_error:nn` ..... 17, 25  
    `\msg_error:nnn` ..... 26  
    `\msg_error:nnnn` ..... 27, 2612  
    `\msg_fatal:nn` ..... 28  
    `\msg_fatal:nnn` ..... 29  
    `\msg_new:nnn` ..... 10, 30  
    `\msg_new:nnnn` ..... 31  
    `\msg_redirect_name:nnn` ..... 33  
    `\msg_warning:nn` ..... 57  
`\multicolumn` ..... 654, 2017, 2031, 2037, 2050  
`\myfiledate` ..... 7  
`\myfileversion` ..... 8

## N

`\newcolumntype` .... 636, 637, 638, 676, 690, 707  
`\NewDocumentCommand` ..... 389, 1972, 1980,  
    1988, 1996, 2004, 2036, 2040, 2127, 2573, 2694  
`\NewDocumentEnvironment` ..... 735,  
    1022, 1032, 1214, 1222, 1230, 1238, 1246,  
    1254, 1264, 2359, 3043, 3049, 3055, 3061,  
    3067, 3073, 3080, 3087, 3094, 3101, 3108, 3114  
`\NewExpandableDocumentCommand` ..... 1839  
`\NiceArrayWithDelims` .....  
    1219, 1227, 1235, 1243, 1251, 1259, 3111, 3118  
`\NiceMatrixLastEnv` ..... 1839  
`\NiceMatrixOptions` ..... 389, 2898

<code>\noalign</code> . . . . .	561, 616, 2273	<code>\RenewDocumentEnvironment</code> . . . . .	
<code>\normalbaselines</code> . . . . .	604	. . . . .	2673, 2676, 2679, 2682, 2685
<code>\nulldelimiterspace</code> . . . . .	1003, 1015	<code>\RequirePackage</code> . . . . .	1, 3, 4, 22, 23, 24
<code>\numexpr</code> . . . . .	183, 184	<code>\right</code> . . . . .	965, 999, 1011
<b>O</b>		<code>\rotate</code> . . . . .	656
<code>\omit</code> . . . . .	1079, 1080, 1115	<b>S</b>	
<code>\OnlyMainNiceMatrix</code> . . . . .	657, 2179	<code>\scriptstyle</code> . . . . .	436, 1139, 1181
<b>P</b>		seq commands:	
peek commands:		<code>\seq_clear:N</code> . . . . .	2754
<code>\peek_meaning_ignore_spaces:N</code> . . . . .	1024	<code>\seq_count:N</code> . . . . .	1046
<code>\pgfcoordinate</code> . . . . .	461, 569, 1084, 1104, 1120	<code>\seq_gclear_new:N</code> . . . . .	668, 669, 1042, 1057
<code>\pgfextracty</code> . . . . .	2656	<code>\seq_gpop_left:NN</code> . . . . .	1048, 1059
<code>\pgfgetlastxy</code> . . . . .	221	<code>\seq_gput_left:Nn</code> . . . . .	327, 2023, 2025
<code>\pgfnode</code> . . . . .	195, 222, 518, 2659, 2664	<code>\seq_gset_split:Nnn</code> . . . . .	1044, 1058
<code>\pgfnodealias</code> . . . . .		<code>\seq_if_in:NnTF</code> . . . . .	325, 2415, 2423, 2773
. . . . .	466, 526, 573, 1086, 1106, 1124, 2541, 2568	<code>\seq_map_function:NN</code> . . . . .	1050, 1061
<code>\pgfpathcircle</code> . . . . .	1964	<code>\seq_map_inline:Nn</code> . . . . .	2755
<code>\pgfpathlineto</code> . . . . .	2246	<code>\seq_maphread_function:NNN</code> . . . . .	2547
<code>\pgfpathmoveto</code> . . . . .	2229	<code>\seq_new:N</code> . . . . .	78
<code>\pgfpathrectanglecorners</code> . . . . .	2635	<code>\seq_put_left:Nn</code> . . . . .	2757
<code>\pgfpicture</code> . . . . .	44, 459, 510, 567, 892, 921, 1082, 1102, 1118, 2204, 2329, 2454, 2464, 2475, 2615	<code>\seq_set_eq:NN</code> . . . . .	2759
<code>\pgfpoint</code> . . . . .	194, 1965, 2234, 2251, 2636, 2637, 2658	<code>\seq_set_from_clist:Nn</code> . . . . .	2763
<code>\pgfpointadd</code> . . . . .	219, 2231, 2248	<code>\seq_use:Nnnn</code> . . . . .	3028
<code>\pgfpointanchor</code> . . . . .		<code>\l_tmpa_seq</code> . . . . .	2754, 2757, 2759
. . . . .	75, 1566, 1571, 2412, 2420, 2651, 2652	skip commands:	
<code>\pgfpointdiff</code> . . . . .	220	<code>\skip_gadd:Nn</code> . . . . .	1097
<code>\pgfpointorigin</code> . . . . .	463, 570, 1084, 1104, 1121	<code>\skip_gset:Nn</code> . . . . .	1088
<code>\pgfpointscale</code> . . . . .	219	<code>\skip_gset_eq:NN</code> . . . . .	1095, 1096
<code>\pgfpointshapeborder</code> . . . . .	2171, 2174	<code>\skip_horizontal:N</code> . . . . .	714, 850, 852, 853, 864, 865, 866, 885, 886, 959, 961, 974, 975, 1016, 1017, 1019, 1020, 1099, 1117, 1165, 1166, 1167, 1169, 1201, 1206, 1207, 1208
<code>\pgfrememberpicturepositiononpagetrue</code> . . . . .	460, 512, 568, 1083, 1103, 1119, 1400, 1843, 1869, 2170, 2205, 2295, 2338, 2455, 2465, 2476, 2616	<code>\skip_vertical:N</code> . . . . .	956, 963, 2273
<code>\pgfset</code> . . . . .	188, 213, 513, 2642	<code>\skip_vertical:n</code> . . . . .	2120
<code>\pgfsetbaseline</code> . . . . .	511	<code>\g_tmpa_skip</code> . . . . .	1088, 1095, 1096, 1097, 1099, 1117
<code>\pgfsetfillcolor</code> . . . . .	2629	<code>\c_zero_skip</code> . . . . .	621
<code>\pgfsetlinewidth</code> . . . . .	2207	<code>\space</code> . . . . .	111, 112
<code>\pgftransformshift</code> . . . . .	194, 219, 2658, 2663	str commands:	
<code>\pgfusepathqfill</code> . . . . .	1970, 2210, 2213, 2638	<code>\c_backslash_str</code> . . . . .	111
<code>\pgfusepathqstroke</code> . . . . .	2264	<code>\c_colon_str</code> . . . . .	388
<code>\phantom</code> . . . . .	1977, 1985, 1993, 2001, 2009	<code>\str_gclear:N</code> . . . . .	1386
<code>\pNiceArray</code> . . . . .	3046, 3077	<code>\str_gset:Nn</code> . . . . .	741, 1218, 1225, 1233, 1241, 1249, 1257, 1266, 2725
<code>\pNiceMatrix</code> . . . . .	2674	<code>\str_if_empty:N</code> . . . . .	464, 524, 571, 740, 773, 791, 1085, 1105, 1122, 1217, 1224, 1232, 1240, 1248, 1256, 1306, 1327, 2539, 2566
prg commands:		<code>\str_if_eq:nnTF</code> . . . . .	110, 317, 372, 558, 890, 906, 909, 1071
<code>\prg_do_nothing:</code> . . . . .	157, 166, 616	<code>\str_if_eq_p:nn</code> . . . . .	263
<code>\prg_replicate:nn</code> . . . . .	1111, 1112, 2037, 2050, 2704, 2707, 2710, 2716	<code>\str_lowercase:n</code> . . . . .	686, 701
<code>\ProcessKeysOptions</code> . . . . .	2751	<code>\str_new:N</code> . . . . .	105, 106, 235, 244, 387
<code>\ProcessOptions</code> . . . . .	21	<code>\str_set:Nn</code> . . . . .	107, 236, 324, 381, 407, 408, 409, 910
<code>\ProvideDocumentCommand</code> . . . . .	58	<code>\str_set_eq:NN</code> . . . . .	328, 388
<code>\ProvidesExplPackage</code> . . . . .	5	<code>\l_tmpa_str</code> . . . . .	324, 325, 327, 328
<b>Q</b>		<b>T</b>	
quark commands:		<code>\tabskip</code> . . . . .	621
<code>\q_stop</code> . . . . .	2125, 2135, 2136, 2552, 2559, 2574, 2575, 2689, 2698	TeX and L <sup>A</sup> T <sub>E</sub> X commands:	
<b>R</b>		<code>\@acol</code> . . . . .	553
<code>\relax</code> . . . . .	21, 183, 184, 674, 675	<code>\@acoll</code> . . . . .	551
<code>\renewcommand</code> . . . . .	171	<code>\@acolr</code> . . . . .	552
		<code>\@addtopreamble</code> . . . . .	755



# Contents

<b>1</b>	<b>Presentation</b>	<b>1</b>
<b>2</b>	<b>The environments of this extension</b>	<b>2</b>
<b>3</b>	<b>The continuous dotted lines</b>	<b>2</b>
3.1	The option nullify-dots . . . . .	3
3.2	The command \Hdotsfor . . . . .	4
3.3	How to generate the continuous dotted lines transparently . . . . .	4
3.4	Customization of the dotted lines . . . . .	5
<b>4</b>	<b>The PGF/Tikz nodes created by nicematrix</b>	<b>6</b>
<b>5</b>	<b>The code-after</b>	<b>7</b>
<b>6</b>	<b>The environment {NiceArray}</b>	<b>8</b>
<b>7</b>	<b>The exterior rows and columns</b>	<b>9</b>
<b>8</b>	<b>The dotted lines to separate rows or columns</b>	<b>10</b>
<b>9</b>	<b>The width of the columns</b>	<b>11</b>
<b>10</b>	<b>Block matrices</b>	<b>12</b>
<b>11</b>	<b>Advanced features</b>	<b>13</b>
11.1	Alignement option in NiceMatrix . . . . .	13
11.2	The command \rotate . . . . .	13
11.3	The option small . . . . .	14
11.4	The counters iRow and jCol . . . . .	14
11.5	The options hlines, vlines and hvlines . . . . .	15
11.6	The option light-syntax . . . . .	15
11.7	Use of the column type S of siunitx . . . . .	16
<b>12</b>	<b>Technical remarks</b>	<b>16</b>
12.1	Definition of new column types . . . . .	16
12.2	Intersections of dotted lines . . . . .	16
12.3	The names of the PGF nodes created by nicematrix . . . . .	17
12.4	Diagonal lines . . . . .	17
12.5	The “empty” cells . . . . .	17
12.6	The option exterior-arraycolsep . . . . .	18
12.7	The class option draft . . . . .	18
12.8	A technical problem with the argument of \\\ . . . . .	18
12.9	Obsolete environments . . . . .	19
<b>13</b>	<b>Examples</b>	<b>19</b>
13.1	Dotted lines . . . . .	19
13.2	Width of the columns . . . . .	21
13.3	How to highlight cells of the matrix . . . . .	21
13.4	Direct use of the Tikz nodes . . . . .	24
<b>14</b>	<b>Implementation</b>	<b>25</b>
<b>15</b>	<b>History</b>	<b>97</b>
	<b>Index</b>	<b>100</b>