

# The package `nicematrix`\*

F. Pantigny  
 fpantigny@wanadoo.fr

February 28, 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` and `tikz`. It also loads the Tikz library `fit`. 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.

$$\begin{array}{c} \textcolor{blue}{L_1} \\ \textcolor{blue}{L_2} \\ \vdots \\ \textcolor{blue}{L_n} \end{array} \begin{array}{c} \textcolor{blue}{C_1} \\ \textcolor{blue}{C_2} \cdots \textcolor{blue}{C_n} \end{array} \left[ \begin{array}{cccc} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{array} \right]$$

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 3.12 of `nicematrix`, at the date of 2020/02/28.

<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}`.

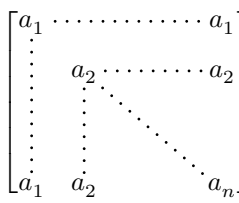
The environment `{NiceArray}` is similar to the environment `{array}` of the package `{array}`. However, for technical reasons, in the preamble of the environment `{NiceArray}`, the user must use the letters L, C and R instead of l, c and r. It's possible to use the constructions `w{...}{...}`, `W{...}{...}`<sup>2</sup>, `l{...}`, `>{...}`, `<{...}`, `@{...}`, `!{...}` and `*{n}{...}` but the letters p, m and b should not be used. See p. 7 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.

<code>\begin{bNiceMatrix}</code>		
<code>a_1</code>	<code>&amp; \Cdots &amp;</code>	<code>&amp; &amp; a_1 \\</code>
<code>\Vdots</code>	<code>&amp; a_2 &amp;</code>	<code>&amp; \Cdots &amp; &amp; a_2 \\</code>
	<code>&amp; \Vdots &amp;</code>	<code>&amp; \Ddots \\</code>
<code>\\</code>		
<code>a_1</code>	<code>&amp; a_2 &amp;</code>	<code>&amp; &amp; a_n</code>
<code>\end{bNiceMatrix}</code>		



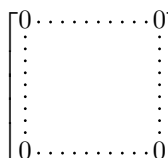
In order to represent the null matrix, one can use the following codage:

<code>\begin{bNiceMatrix}</code>		
<code>0</code>	<code>&amp; \Cdots &amp;</code>	<code>0 \\</code>
<code>\Vdots</code>	<code>&amp;</code>	<code>&amp; \Vdots \\</code>
<code>0</code>	<code>&amp; \Cdots &amp;</code>	<code>0</code>
<code>\end{bNiceMatrix}</code>		



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`:

<code>\begin{bNiceMatrix}</code>		
<code>0</code>	<code>&amp; \Cdots &amp;</code>	<code>\Cdots &amp; 0 \\</code>
<code>\Vdots</code>	<code>&amp;</code>	<code>&amp; \Vdots \\</code>
<code>\Vdots</code>	<code>&amp;</code>	<code>&amp; \Vdots \\</code>
<code>0</code>	<code>&amp; \Cdots &amp;</code>	<code>\Cdots &amp; 0</code>
<code>\end{bNiceMatrix}</code>		



<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. 16).

In the first column of this exemple, there are two instructions `\Vdots` but only one dotted line is drawn (there is no overlapping graphic objects in the resulting PDF<sup>5</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 & \\
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>6</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}
```

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

### 3.1 The option `nullify-dots`

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

```
$A = \begin{pmatrix}
a_0 & b \\
a_1 & \\
a_2 & \\
a_3 & \\
a_4 & \\
a_5 & b
\end{pmatrix}$
```

$$A = \begin{pmatrix} a_0 & b \\ a_1 & \\ a_2 & \\ a_3 & \\ a_4 & \\ a_5 & b \end{pmatrix}$$

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

```
$B = \begin{pmatrix}
a_0 & b \\
a_1 & \vdots \\
a_2 & \vdots \\
a_3 & \vdots \\
a_4 & \vdots \\
a_5 & b
\end{pmatrix}$
```

$$B = \begin{pmatrix} a_0 & b \\ a_1 & \vdots \\ a_2 & \vdots \\ a_3 & \vdots \\ a_4 & \vdots \\ a_5 & b \end{pmatrix}$$

<sup>5</sup>And it's not possible to draw a `\Ldots` and a `\Cdots` line between the same cells.

<sup>6</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. 10

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

$$\begin{array}{l}
 \$C = \begin{pNiceMatrix} \\
 a_0 \& b \quad \backslash \\
 a_1 \& \Vdots \quad \backslash \\
 a_2 \& \Vdots \quad \backslash \\
 a_3 \& \Vdots \quad \backslash \\
 a_4 \& \Vdots \quad \backslash \\
 a_5 \& b \\
 \end{pNiceMatrix} \$
 \end{array}
 \qquad
 C = \begin{pmatrix} a_0 & b \\ \vdots & \\ a_1 & \vdots \\ \vdots & \\ a_2 & \vdots \\ \vdots & \\ a_3 & \vdots \\ \vdots & \\ a_4 & \vdots \\ \vdots & \\ a_5 & b \end{pmatrix}$$

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

$$\begin{array}{l}
 \$D = \begin{pNiceMatrix}[nullify-dots] \\
 a_0 \& b \quad \backslash \\
 a_1 \& \Vdots \quad \backslash \\
 a_2 \& \quad \backslash \\
 a_3 \& \quad \backslash \\
 a_4 \& \quad \backslash \\
 a_5 \& b \\
 \end{pNiceMatrix} \$
 \end{array}
 \qquad
 D = \begin{pmatrix} a_0 & b \\ a_1 & \vdots \\ a_2 & \vdots \\ a_3 & \vdots \\ a_4 & \vdots \\ a_5 & b \end{pmatrix}$$

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

**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.

$$\begin{array}{l}
 \$\begin{pNiceMatrix} \\
 1 \& 2 \& 3 \& 4 \& 5 \quad \backslash \\
 1 \& \Hdotsfor{3} \& 5 \quad \backslash \\
 1 \& 2 \& 3 \& 4 \& 5 \quad \backslash \\
 1 \& 2 \& 3 \& 4 \& 5 \\
 \end{pNiceMatrix} \$
 \end{array}
 \qquad
 \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & \dots & \dots & \dots & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$$

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

$$\begin{array}{l}
 \$\begin{pNiceMatrix} \\
 1 \& 2 \& 3 \& 4 \& 5 \quad \backslash \\
 \& \Hdotsfor{3} \quad \backslash \\
 1 \& 2 \& 3 \& 4 \& 5 \quad \backslash \\
 1 \& 2 \& 3 \& 4 \& 5 \\
 \end{pNiceMatrix} \$
 \end{array}
 \qquad
 \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ & \dots & \dots & \dots & \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$$

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>7</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`.

- 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}
\end{pmatrix}
```

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

### 3.4 Fine tuning of the dotted lines

The distance between a node and the end of a dotted line is set by `dotted-lines-margin`. The initial value of this key is 0.3 em (it’s recommended to use a unit dependent of the current font). For an example, cf. p. 18.

## 4 The Tikz nodes created by `nicematrix`

The package `nicematrix` creates a Tikz node for each 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. 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.

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

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

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

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

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

<sup>7</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.)

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>8</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.

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

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>9</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>10</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. 20).

<sup>8</sup>There is also an option `create-extra-nodes` which is an alias for the conjunction of `create-medium-nodes` and `create-large-nodes`.

<sup>9</sup>There is no “large nodes” created in the exterior rows and columns (for these rows and columns, cf. p. 8).

<sup>10</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.

## 5 The code-after

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

In the `code-after`, the Tikz nodes should be accessed by a name of the form  $i$ - $j$  (without the prefix of 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}]
0 & 0 & 0 & \\\
0 & & 0 & \\\
0 & 0 & 0 & \\
\end{pNiceMatrix}
```

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

## 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>11</sup> instead of l, c and r. It's possible to use the constructions `w{...}{...}`, `W{...}{...}`, `|`, `>{...}`, `<{...}`, `@{...}`, `!{...}` and `*{n}{...}` but the letters p, m and b should not be used.<sup>12</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. 13.)

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

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 `\firstline`<sup>13</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}
```

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

<sup>11</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>12</sup>In a command `\multicolumn`, one should also use the letters L, C, R.

<sup>13</sup>It's also possible to use `\firstline` with `{NiceArray}`.

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:

```
$\left[\begin{NiceArray}{CCCC|C}
a_1 & ? & & \Cdots & ? & ? \\
0 & & & \Ddots & \Vdots & \Vdots \\
\Vdots & \Ddots & \Ddots & \Ddots & ? & \\
0 & & \Cdots & 0 & a_n & ?
\end{NiceArray}\right]$
```

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

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`).

```
$\begin{pNiceArray}{LCR}
a_{11} & & \Cdots & a_{1n} \\
a_{21} & & & a_{2n} \\
\Vdots & & & \Vdots \\
a_{n-1,1} & & \Cdots & a_{n-1,n}
\end{pNiceArray}$
```

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

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

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

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

## 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 has the number 0 (and not 1). Idem for the potential first column. In general cases, one must specify the number of the last row and the number of the last column as values of `last-row` and `last-col`.

```
$\begin{pNiceMatrix}[first-row,last-row=5,first-col,last-col=5]
& C_1 & C_2 & C_3 & C_4 & \\
L_1 & a_{11} & a_{12} & a_{13} & a_{14} & L_1 \\
L_2 & a_{21} & a_{22} & a_{23} & a_{24} & L_2 \\
L_3 & a_{31} & a_{32} & a_{33} & a_{34} & L_3 \end{pNiceMatrix}
```



```
L_4 & a_{41} & a_{42} & a_{43} & a_{44} & L_4 \\
& C_1 & C_2 & C_3 & C_4 & \\
\end{pNiceMatrix}$
```

$$\begin{array}{cccc} C_1 & C_2 & C_3 & C_4 \\ L_1 & \left( \begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \end{array} \right) & L_1 \\ L_2 & \left( \begin{array}{cc|cc} a_{21} & a_{22} & a_{23} & a_{24} \end{array} \right) & L_2 \\ L_3 & \left( \begin{array}{cc|cc} a_{31} & a_{32} & a_{33} & a_{34} \end{array} \right) & L_3 \\ L_4 & \left( \begin{array}{cc|cc} a_{41} & a_{42} & a_{43} & a_{44} \end{array} \right) & L_4 \\ C_1 & C_2 & C_3 & C_4 \end{array}$$

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.
- In an environment with an explicit preamble, the option `last-col` must be used *without* value: the number of columns will be automatically computed from the preamble of the array.
- For the potential last row, the option `last-row` may, in fact, be used without value. In this case, `nicematrix` computes, during the first compilation, the number of rows of the array and writes that information in the `.aux` file for the second run. In the following example, the option `last-row` will be used without value.

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}}
$\begin{pNiceArray}{CC|CC}[first-row,last-row,first-col,last-col]
& C_1 & C_2 & C_3 & C_4 & \\
L_1 & a_{11} & a_{12} & a_{13} & a_{14} & L_1 \\
L_2 & a_{21} & a_{22} & a_{23} & a_{24} & L_2 \\
\hline
L_3 & a_{31} & a_{32} & a_{33} & a_{34} & L_3 \\
L_4 & a_{41} & a_{42} & a_{43} & a_{44} & L_4 \\
& C_1 & C_2 & C_3 & C_4 & \\
\end{pNiceArray}$
```

$$\begin{array}{cccc} C_1 & C_2 & C_3 & C_4 \\ L_1 & \left( \begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \end{array} \right) & L_1 \\ L_2 & \left( \begin{array}{cc|cc} a_{21} & a_{22} & a_{23} & a_{24} \end{array} \right) & L_2 \\ L_3 & \left( \begin{array}{cc|cc} a_{31} & a_{32} & a_{33} & a_{34} \end{array} \right) & L_3 \\ L_4 & \left( \begin{array}{cc|cc} a_{41} & a_{42} & a_{43} & a_{44} \end{array} \right) & L_4 \\ C_1 & C_2 & C_3 & C_4 \end{array}$$

#### 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>14</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 15.
- Logically, the potential option `columns-width` (described p. 10) doesn't apply to the "first column" and "last column".
- For technical reasons, it's not possible to use the option of the command `\` after the "first row" or before the "last row" (the placement of the delimiters would be wrong).

<sup>14</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.

## 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}
```

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

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}$
```

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

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 = V}
\left(\begin{NiceArray}{C|C:CVC}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12
\end{NiceArray}\right)
```

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

*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>15</sup>. In `nicematrix`, the dotted lines drawn `\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`.

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

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

<sup>15</sup>In fact, this is true only for `\hline` and “|” but not for `\cline`.

In the environments of `nicematrix`, it's also possible to fix the 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}
```

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

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>16</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 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>17</sup>. The environment `{NiceMatrixBlock}` has no direct link with the command `\Block` presented just below (cf. p. 11).

```
\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`.

<sup>16</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>17</sup>At this time, this is the only usage of the environment `{NiceMatrixBlock}` but it may have other usages in the future.

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

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

$$\begin{array}{ccc} \text{image of } e_1 & \text{image of } e_2 & \text{image of } e_3 \\ \left( \begin{array}{ccc} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{array} \right) & e_1 & e_2 & e_3 \end{array}$$

### 11.2 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`).

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

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

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.

### 11.3 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>18</sup>. Of course, the user must not change the value of these counters which are used internally by `nicematrix`.

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

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

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

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).

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

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

### 11.4 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`.

<sup>18</sup>We recall that the first row (if it exists) has the number 0 and that the first column (if it exists) has also the number 0.

```

\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}
$\begin{pNiceMatrix}[vlines]
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 2 & 3 & 4 & 5 & 6 \\
\end{pNiceMatrix}$

```

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

## 11.5 The option `light-syntax`

The option `light-syntax`<sup>19</sup> allow 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 a space to separate the columns. However, as usual in the TeX world, the spaces after a control sequence are discarded and the elements between curly braces are considered as a whole.

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

```

$\begin{bNiceMatrix}[light-syntax,first-row,first-col]
{} \alpha \qquad \qquad \beta \qquad ;
\alpha \ 2 \backslash \cos \alpha \qquad \{ \backslash \cos \alpha + \backslash \cos \beta \} ;
\beta \backslash \cos \alpha + \backslash \cos \beta \ 2 \backslash \cos \beta \}
\end{bNiceMatrix}$

```

$$\begin{matrix} \alpha & \beta \\ \alpha \begin{bmatrix} 2 \cos \alpha & \cos \alpha + \cos \beta \\ \cos \alpha + \cos \beta & 2 \cos \beta \end{bmatrix} \end{matrix}$$

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.

## 11.6 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`.

```

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

```

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

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

<sup>19</sup>This option is inspired by the extension `spalign` of Joseph Rabinoff.

## 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) thick rule of width 1 pt. The following definition will do the job<sup>20</sup>:

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

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

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

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

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>21</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.

```
\begin{pNiceMatrix}[nullify-dots]
1 & 2 & 3 & \Cdots & n \\
1 & 2 & 3 & \Cdots & n \\
\Vdots & \Cdots & & \Hspace*{15mm} & \Vdots \\
& \Cdots & & & \\
& \Cdots & & & \\
& \Cdots & & & \\
\end{pNiceMatrix}
```

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

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

We have said that, when a name is given to an environment of `nicematrix`, it's possible to access the Tikz nodes through this name (cf. p. 5).

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` have the same name, suffixed by `-medium` and `-large`.

<sup>20</sup>The command `\vrule` is a TeX (and not LaTeX) command.

<sup>21</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.

## 12.4 Diagonal lines

By default, all the diagonal lines<sup>22</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):

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

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

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

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

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

The same example without parallelization:

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

## 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 with contents `\hspace*{1cm}` may be considered as empty.

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

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

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

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

- Each cell whose TeX output has a width less than 0.5 pt is empty.
- A cell which contains a command `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots` or `\Iddots` is empty. We recall that these commands should be used alone in a cell.
- A cell with a command `\Hspace` (or `\Hspace*`) is empty. This command `\Hspace` is a command defined by the package `nicematrix` with the same meaning as `\hspace` except that the cell where it is used is considered as empty. This command can be used to fix the width of some columns of the matrix without interfering with `nicematrix`.

<sup>22</sup>We speak of the lines created by `\Ddots` and not the lines created by a command `\line` in `code-after`.



## 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>23</sup>. The environment `{matrix}` of `amsmath` and its variants (`{pmatrix}`, `{vmatrix}`, etc.) of `amsmath` prefer to delete these spaces with explicit instructions `\hskip -\arraycolsep`. 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`

The package `nicematrix` is rather slow when drawing the dotted lines (generated by `\Cdots`, `\Ldots`, `\Ddots`, etc. but also by `\hdottedline` or the specifier `:`).<sup>24</sup> That's why, 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{AB}{c} \\
b & c
\end{pNiceMatrix}
```

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

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{AB}{c} \\
\noalign{\kern2mm}
b & c
\end{pNiceMatrix}
```

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

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

```
\begin{pNiceMatrix}
a & \multicolumn{1}{c}{\frac{AB}{c}} \\
b & c
\end{pNiceMatrix}
```

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

## 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`.

<sup>23</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).* It's possible to suppress these spaces for a given environment `{array}` with a construction like `\begin{array}{@{}cccc@{}}...\end{array}`.

<sup>24</sup>The main reason is that we want dotted lines with round dots (and not square dots) with the same space on both extremities of the lines. To achieve this goal, we have to construct our own system of dotted lines.

## 13 Examples

### 13.1 Dotted lines

A tridiagonal matrix:

```

 $\begin{pNiceMatrix}[nullify-dots]$ 
a      & b      & 0      & & & \Cdots & 0      & \\
b      & a      & b      & & \Ddots & & & \Vdots \\
0      & b      & a      & & \Ddots & & & \\
      & & \Ddots & & \Ddots & & & \\
\Vdots & & & & & & & b      & \\
0      & \Cdots & & & 0      & b      & a      & \\
 $\end{pNiceMatrix}$ 

```

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

A permutation matrix (as an example, we have raised the value of `dotted-lines-margin`).

```

 $\begin{pNiceMatrix}[\textcolor{blue}{dotted-lines-margin=0.6em}]$ 
0      & 1 & 0 & & & \Cdots & 0      & \\
\Vdots & & & & \Ddots & & & \Vdots \\
      & & & & \Ddots & & & \\
      & & & & \Ddots & & 0      & \\
0      & 0 & & & & & 1      & \\
1      & 0 & & & \Cdots & & 0      & \\
 $\end{pNiceMatrix}$ 

```

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

An example with `\Iddots`:

```

 $\begin{pNiceMatrix}$ 
1      & \Cdots & & & 1      & \\
\Vdots & & & & 0      & \\
      & \textcolor{blue}{\Iddots} & \textcolor{blue}{\Iddots} & & \Vdots & \\
1      & 0      & & \Cdots & 0      & \\
 $\end{pNiceMatrix}$ 

```

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

An example with `\multicolumn`:

```

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

```

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

An example with `\Hdotsfor`:

```
\begin{pNiceMatrix}[nullify-dots]
0 & 1 & 1 & 1 & 1 & 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}
\[\begin{vNiceArray}{CCCC:CCC}[columns-width=6mm]
a_0 & & & & & & & & \\
a_1 & & \Ddots & & & b_1 & & \Ddots & \\
\Vdots & \Ddots & & & & \Vdots & \Ddots & b_0 & \\
a_p & & & a_0 & & & & b_1 & \\
& & \Ddots & & a_1 & & & \Vdots & \\
& & & & \Vdots & & & \Ddots & \\
& & & a_p & & & & & b_q
\end{vNiceArray}\]
```

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

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

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

## 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 $\begin{pNiceArray}{CCCC:C}[last-col]
1&1&1&1&\backslash\backslash
2&4&8&16&9&\backslash\backslash
3&9&27&81&36&\backslash\backslash
4&16&64&256&100&
\end{pNiceArray}$
...
\end{NiceMatrixBlock}
```

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

## 13.3 How to highlight cells of the matrix

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”).

In order to have the continuity of the lines, we have to set `inner sep = -\pgflinewidth/2`.

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

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

We should remark that the rules we have drawn are drawn *after* the construction of the array and, hence, they don't spread the cells of the array. We recall that, on the other side, the command `\hline` and the specifier “|” spread the cells (when the package `array` is loaded but, when the package `nicematrix` is loaded, `array` is always loaded).<sup>25</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`.

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

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

Considerer now the following matrix which we have named **example**.

---

<sup>25</sup>On the other side, the command `\cline` doesn't spread the rows of the array.

```

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

```

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

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

```

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

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

```

We obtain the following matrix.

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

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

```

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

```

We obtain the following matrix.

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

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

```

\begin{pNiceArray}{>\strut}CCCC}%
[create-large-nodes,margin,extra-margin=2pt,
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} \backslash \backslash
A_{21} \& A_{22} \& A_{23} \& A_{24} \backslash \backslash
A_{31} \& A_{32} \& A_{33} \& A_{34} \backslash \backslash
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}
```

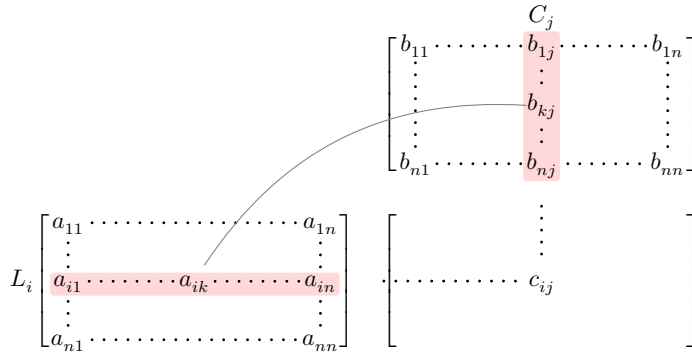
```
\begin{tikzpicture}[remember picture, overlay]
```

```
\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}
```



## 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 `\ialign` to begin the `\halign`.

## Declaration of the package and extensions loaded

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

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

 $\langle @@ = \text{nicematrix} \rangle$ 

First, `tikz` and the Tikz library `fit` are loaded before the `\ProvidesExplPackage`. They are loaded this way because `\usetikzlibrary` in `expl3` code fails.<sup>26</sup>

```
1 \RequirePackage{tikz}
2 \usetikzlibrary{fit}
3 \RequirePackage{expl3}[2020/02/08]
```

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

```

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

```

<sup>26</sup>cf. [tex.stackexchange.com/questions/57424/using-of-usetikzlibrary-in-an-expl3-package-fails](https://tex.stackexchange.com/questions/57424/using-of-usetikzlibrary-in-an-expl3-package-fails)



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 `array` and `amsmath`.

```

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

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.

```

34 \bool_new:N \c_@@_revtex_bool
35 \ifclassloaded { revtex4-1 }
36 { \bool_set_true:N \c_@@_revtex_bool }
37 { }
38 \ifclassloaded { revtex4-2 }
39 { \bool_set_true:N \c_@@_revtex_bool }
40 { }

```

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

```

41 \@@_msg_new:nn { Draft-mode }
42 { The~compilation-is-in-draft-mode:-the-dotted-lines-won't-be-drawn. }

43 \bool_if:NT \c_@@_draft_bool
44 { \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.

```

45 \ProvideDocumentCommand \iddots { }
46 {
47   \mathinner
48   {
49     \tex_mkern:D 1 mu
50     \box_move_up:nn { 1 pt } { \hbox:n { . } }
51     \tex_mkern:D 2 mu
52     \box_move_up:nn { 4 pt } { \hbox:n { . } }
53     \tex_mkern:D 2 mu
54     \box_move_up:nn { 7 pt }
55     { \vbox:n { \kern 7 pt \hbox:n { . } } }
56     \tex_mkern:D 1 mu
57   }
58 }

```

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.

```
59 \int_new:N \g_@@_env_int
```

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

```
60 \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).

```
61 \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`.

```
62 \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.

```
63 \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`).

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

```
65 \cs_new_protected:Npn \@@_test_if_math_mode:
66 {
67   \if_mode_math: \else:
68     \@@_fatal:n { Outside-math-mode }
69   \fi:
70 }
```

Consider the following code:

```

$\begin{pNiceMatrix}
a & b & c \\
d & e & \Vdots \\
f & \Cdots & \\
g & h & i \\
\end{pNiceMatrix}$

```

First, the dotted line created by the `\Vdots` will be drawn. The implicit cell in position 2-3 will be considered as “dotted”. Then, we will have to draw the dotted line specified by the `\Cdots`; the final extremity of that line will be exactly in position 2-3 and, for that new second line, it should be considered as a *closed* extremity (since it is dotted). However, we don't have the (normal) Tikz node of that node (since it's an implicit cell): we can't draw such a line. That's why that dotted line will be said *impossible* and an error will be raised.<sup>27</sup>

```
71 \bool_new:N \l_@@_impossible_line_bool
```

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

```

72 \bool_new:N \c_@@_colortbl_loaded_bool
73 \AtBeginDocument
74 {
75   \@ifpackageloaded { colortbl }
76   {

```

---

<sup>27</sup>Of course, the user should solve the problem by adding the lacking ampersands.

```

77     \bool_set_true:N \c_@@_colortbl_loaded_bool
78     \cs_set_protected:Npn \@@_vline_i: { { \CT@arc@ \vline } }
79   }
80   { }
81 }

```

We have put a argument *w* (*weird*) for the following function because its argument should be a specifier of pgf point between rounded brackets.

```

82 \cs_set_protected:Npn \@@_extract_coords:w
83 { \tikz@parse@node \pgfutil@firstofone }

```

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.

```

84 \dim_new:N \l_@@_inter_dots_dim
85 \dim_set:Nn \l_@@_inter_dots_dim { 0.45 em }

```

The length `\l_@@_dotted_lines_margin_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).

```

86 \dim_new:N \l_@@_dotted_lines_margin_dim
87 \dim_set:Nn \l_@@_dotted_lines_margin_dim { 0.3 em }

```

The length `\l_@@_radius_dim` is the radius of the dots for the dotted lines. The default value is 0.53 pt but it will be changed if the option `small` is used (to 0.37 pt).

```

88 \dim_new:N \l_@@_radius_dim
89 \dim_set:Nn \l_@@_radius_dim { 0.53 pt }

```

The name of the current environment or the current command (despite the name).

```

90 \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 a an environment of *nicematrix*. The default value is *environment*.

```

91 \str_new:N \g_@@_com_or_env_str
92 \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). This command must *not* be protected since it’s used in error messages.

```

93 \cs_new:Npn \@@_full_name_env:
94 {
95   \str_if_eq:VnTF \g_@@_com_or_env_str { command }
96     { command \space \c_backslash_str \g_@@_name_env_str }
97     { environment \space \{ \g_@@_name_env_str \} }
98 }

```

```

99 \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.

```

100 \int_new:N \l_@@_save_iRow_int
101 \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).

```

102 \bool_new:N \g_@@_row_of_col_done_bool

```

## 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`.

```
103 \int_new:N \l_@@_first_row_int
104 \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.

```
105 \int_new:N \l_@@_first_col_int
106 \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).

```
107 \int_new:N \l_@@_last_row_int
108 \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>28</sup>

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

- **Last column**

For the eventual “last column”, we use an integer. A value of `-1` means that there is no last column.

```
110 \int_new:N \l_@@_last_col_int
111 \int_set:Nn \l_@@_last_col_int { -1 }
```

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.

```
112 \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`.

```
113 \bool_new:N \c_@@_siunitx_loaded_bool
114 \AtBeginDocument
115 {
116   \@ifpackageloaded { siunitx }
117   { \bool_set_true:N \c_@@_siunitx_loaded_bool }
118   { }
119 }
```

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:

<sup>28</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.

```

\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 *toks* list `\@temptokena`, we use the LaTeX command `\NC@rewrite@S` in a group (`\group_begin:-\group_end:`) and we extract the two command names which are in the *toks* `\@temptokena`. However, this extraction can be done only when `siunitx` is loaded (and it may be loaded after `nicematrix`) and, in fact, after the beginning of the document — because some instructions of `siunitx` are executed in a `\AtBeginDocument`). That's why this extraction will be done only at the first use of an environment of `nicematrix` with the command `\@@_adapt_S_column:`.

```

120 \cs_set_protected:Npn \@@_adapt_S_column:
121 {

```

In the preamble of the LaTeX document, the boolean `\c_@@_siunitx_loaded_bool` won't be known. That's why we test the existence of `\c_@@_siunitx_loaded_bool` and not its value.<sup>29</sup>

```

122   \bool_if:NT \c_@@_siunitx_loaded_bool
123   {
124     \group_begin:
125     \@temptokena = { }

```

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

```

126     \cs_set_eq:NN \NC@find \prg_do_nothing:
127     \NC@rewrite@S { }

```

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

```

128     \tl_gset:NV \g_tmpa_tl \@temptokena
129     \group_end:
130     \tl_new:N \c_@@_table_collect_begin_tl
131     \tl_set:Nx \l_tmpa_tl { \tl_item:Nn \g_tmpa_tl 2 }
132     \tl_gset:Nx \c_@@_table_collect_begin_tl { \tl_item:Nn \l_tmpa_tl 1 }
133     \tl_new:N \c_@@_table_print_tl
134     \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).

```

135     \cs_gset_eq:NN \@@_adapt_S_column: \prg_do_nothing:
136   }
137 }

```

---

<sup>29</sup>Indeed, `nicematrix` may be used in the preamble of the LaTeX document. For example, you can compose a matrix in a box before the loading of `arydshln` (not totally compatible with `nicematrix`).

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).

```

138 \cs_new_protected:Npn \@@_renew_NC@rewrite@S:
139 {
140   \renewcommand*{\NC@rewrite@S}[1] []
141   {
142     \@temptokena \exp_after:wN
143     {
144       \tex_the:D \@temptokena
145       > { \@@_Cell: \c_@@_table_collect_begin_tl S {##1} }
146       c
147       < { \c_@@_table_print_tl \@@_end_Cell: }
148     }
149     \NC@find
150   }
151 }

```

## The options

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

```

152 \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.

```

153 \str_new:N \l_@@_baseline_str
154 \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`).

```

155 \bool_new:N \l_@@_exterior_arraycolsep_bool

```

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

```

156 \bool_new:N \l_@@_parallelize_diags_bool
157 \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`.

```

158 \bool_new:N \l_@@_hlines_bool
159 \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.).

```

160 \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).

```

161 \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.

```

162 \str_new:N \l_@@_name_str

```

The boolean `\l_@@_extra_medium_bool` will be used to indicate whether the “medium nodes” are created in the array. Idem for the “large nodes”.

```
163 \bool_new:N \l_@@_medium_nodes_bool
164 \bool_new:N \g_@@_medium_nodes_bool
165 \bool_new:N \l_@@_large_nodes_bool
166 \bool_new:N \g_@@_large_nodes_bool
```

The dimension `\l_@@_left_margin_dim` correspond to the option `left-margin` (idem for the right margin).

```
167 \dim_new:N \l_@@_left_margin_dim
168 \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”.

```
169 \dim_new:N \g_@@_width_last_col_dim
170 \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`.

```
171 \dim_new:N \l_@@_extra_left_margin_dim
172 \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.

```
173 \tl_new:N \l_@@_end_of_row_tl
174 \tl_set:Nn \l_@@_end_of_row_tl { ; }
```

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

```
175 \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.

```
176 \keys_define:nn { NiceMatrix / Global }
177 {
178   max-delimiter-width .bool_set:N = \l_@@_max_delimiter_width_bool ,
179   dotted-lines-margin .dim_set:N = \l_@@_dotted_lines_margin_dim ,
180   dotted-lines-margin .value_required:n = true ,
181   light-syntax .bool_set:N = \l_@@_light_syntax_bool ,
182   light-syntax .default:n = true ,
183   end-of-row .tl_set:N = \l_@@_end_of_row_tl ,
184   end-of-row .value_required:n = true ,
185   code-for-first-col .tl_set:N = \l_@@_code_for_first_col_tl ,
186   code-for-first-col .value_required:n = true ,
187   code-for-last-col .tl_set:N = \l_@@_code_for_last_col_tl ,
188   code-for-last-col .value_required:n = true ,
189   code-for-first-row .tl_set:N = \l_@@_code_for_first_row_tl ,
190   code-for-first-row .value_required:n = true ,
191   code-for-last-row .tl_set:N = \l_@@_code_for_last_row_tl ,
192   code-for-last-row .value_required:n = true ,
193   small .bool_set:N = \l_@@_small_bool ,
194   hlines .bool_set:N = \l_@@_hlines_bool ,
195   vlines .bool_set:N = \l_@@_vlines_bool ,
196   hvlines .meta:n = { hlines , vlines } ,
197   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`.

```

198   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
199   renew-dots .value_forbidden:n = true ,
200   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”).

```

201   create-medium-nodes .bool_set:N = \l_@@_medium_nodes_bool ,
202   create-large-nodes .bool_set:N = \l_@@_large_nodes_bool ,
203   create-extra-nodes .meta:n =
204     { create-medium-nodes , create-large-nodes } ,
205   left-margin .dim_set:N = \l_@@_left_margin_dim ,
206   left-margin .default:n = \arraycolsep ,
207   right-margin .dim_set:N = \l_@@_right_margin_dim ,
208   right-margin .default:n = \arraycolsep ,
209   margin .meta:n = { left-margin = #1 , right-margin = #1 } ,
210   margin .default:n = \arraycolsep ,
211   extra-left-margin .dim_set:N = \l_@@_extra_left_margin_dim ,
212   extra-right-margin .dim_set:N = \l_@@_extra_right_margin_dim ,
213   extra-margin .meta:n =
214     { extra-left-margin = #1 , extra-right-margin = #1 }
215 }

```

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

```

216 \keys_define:nn { NiceMatrix / Env }
217 {
218   columns-width .code:n =
219     \str_if_eq:nnTF { #1 } { auto }
220     { \bool_set_true:N \l_@@_auto_columns_width_bool }
221     { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,
222   columns-width .value_required:n = true ,
223   name .code:n =
224     \legacy_if:nF { measuring@ }
225     {
226       \str_set:Nn \l_tmpa_str { #1 }
227       \seq_if_in:NVTF \g_@@_names_seq \l_tmpa_str
228         { \@@_error:nn { Duplicate-name } { #1 } }
229         { \seq_gput_left:NV \g_@@_names_seq \l_tmpa_str }
230       \str_set_eq:NN \l_@@_name_str \l_tmpa_str
231     } ,
232   name .value_required:n = true ,
233   code-after .tl_gset:N = \g_@@_code_after_tl ,
234   code-after .value_required:n = true ,
235   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
236   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
237   last-row .int_set:N = \l_@@_last_row_int ,
238   last-row .default:n = -1 ,
239 }

```

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

```

240 \keys_define:nn { NiceMatrix }
241 {
242   NiceMatrixOptions .inherit:n =
243     {
244       NiceMatrix / Global ,
245     } ,
246   NiceMatrix .inherit:n =
247     {
248       NiceMatrix / Global ,
249       NiceMatrix / Env
250     } ,

```



```

251   NiceArray .inherit:n =
252   {
253       NiceMatrix / Global ,
254       NiceMatrix / Env ,
255   } ,
256   pNiceArray .inherit:n =
257   {
258       NiceMatrix / Global ,
259       NiceMatrix / Env ,
260   }
261 }

```

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

```

262 \keys_define:nn { NiceMatrix / NiceMatrixOptions }
263 {

```

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

```

264   renew-matrix .code:n = \@@_renew_matrix: ,
265   renew-matrix .value_forbidden:n = true ,
266   transparent .meta:n = { renew-dots , renew-matrix } ,
267   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}`.

```

268   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.

```

269   columns-width .code:n =
270   \str_if_eq:nnTF { #1 } { auto }
271   { \@@_error:n { Option-auto-for-columns-width } }
272   { \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.

```

273   allow-duplicate-names .code:n =
274   \@@_msg_redirect_name:nn { Duplicate-name } { none } ,
275   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`).

```

276   letter-for-dotted-lines .code:n =
277   {
278       \int_compare:nTF { \tl_count:n { #1 } = 1 }
279       { \str_set:Nx \l_@@_letter_for_dotted_lines_str { #1 } }
280       { \@@_error:n { Bad-value-for-letter-for-dotted-lines } }
281   } ,
282   letter-for-dotted-lines .value_required:n = true ,

283   unknown .code:n = \@@_error:n { Unknown-key-for-NiceMatrixOptions }
284   }

285 \str_new:N \l_@@_letter_for_dotted_lines_str
286 \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.

```
287 \NewDocumentCommand \NiceMatrixOptions { m }
288 { \keys_set:nn { NiceMatrix / NiceMatrixOptions } { #1 } }
```

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

```
289 \keys_define:nn { NiceMatrix / NiceMatrix }
290 {
291   last-col .code:n = \tl_if_empty:nTF {#1}
292     { \@@_error:n { last-col~empty~for~NiceMatrix } }
293     { \int_set:Nn \l_@@_last_col_int { #1 } } ,
294   unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix }
295 }
```

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

```
296 \keys_define:nn { NiceMatrix / NiceArray }
297 {
```

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

```
298   c .code:n = \str_set:Nn \l_@@_baseline_str c ,
299   t .code:n = \str_set:Nn \l_@@_baseline_str t ,
300   b .code:n = \str_set:Nn \l_@@_baseline_str b ,
301   baseline .tl_set:N = \l_@@_baseline_str ,
302   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.

```
303   last-col .code:n = \tl_if_empty:nF { #1 }
304     { \@@_error:n { last-col~non-empty~for~NiceArray } }
305     { \int_zero:N \l_@@_last_col_int ,
306   unknown .code:n = \@@_error:n { Unknown~option~for~NiceArray }
307 }
308 \keys_define:nn { NiceMatrix / pNiceArray }
309 {
310   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
311   last-col .code:n = \tl_if_empty:nF {#1}
312     { \@@_error:n { last-col~non-empty~for~NiceArray } }
313     { \int_zero:N \l_@@_last_col_int ,
314   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
315   last-row .int_set:N = \l_@@_last_row_int ,
316   last-row .default:n = -1 ,
317   unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix }
318 }
```

## 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}`).

```
319 \cs_new_protected:Npn \@@_Cell:
320 {
```

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

```
321   \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.

```

322 \int_compare:nNnT \c@jCol = 1
323 {
324     \int_compare:nNnT \l_@@_first_col_int = 1
325     \@@_begin_of_row:
326 }
327 \int_gset:Nn \g_@@_col_total_int
328 { \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).

```

329 \hbox_set:Nw \l_@@_cell_box
330 \c_math_toggle_token
331 \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.

```

332 \int_compare:nNnTF \c@iRow = 0
333 { \int_compare:nNnT \c@jCol > 0 \l_@@_code_for_first_row_tl }
334 {
335     \int_compare:nNnT \c@iRow = \l_@@_last_row_int
336     \l_@@_code_for_last_row_tl
337 }
338 }

```

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.

```

339 \cs_new_protected:Npn \@@_begin_of_row:
340 {
341     \int_gincr:N \c@iRow
342     \dim_gset_eq:NN \g_@@_dp_ante_last_row_dim \g_@@_dp_last_row_dim
343     \dim_gset:Nn \g_@@_dp_last_row_dim { \box_dp:N \@arstrutbox }
344     \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
345     \tikz [ remember-picture , baseline ] \coordinate
346         ( nm - \int_use:N \g_@@_env_int - row - \int_use:N \c@iRow - base ) ;
347 }

```

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.

```

348 \cs_new_protected:Npn \@@_update_for_first_and_last_row:
349 {
350     \int_compare:nNnTF \c@iRow = 0
351     {
352         \dim_gset:Nn \g_@@_dp_row_zero_dim
353         { \dim_max:nn \g_@@_dp_row_zero_dim { \box_dp:N \l_@@_cell_box } }
354         \dim_gset:Nn \g_@@_ht_row_zero_dim
355         { \dim_max:nn \g_@@_ht_row_zero_dim { \box_ht:N \l_@@_cell_box } }
356     }
357     {
358         \int_compare:nNnT \c@iRow = 1
359         {
360             \dim_gset:Nn \g_@@_ht_row_one_dim
361             { \dim_max:nn \g_@@_ht_row_one_dim { \box_ht:N \l_@@_cell_box } }
362         }
363     }
364 }

```

```

365 \cs_new_protected:Npn \@@_end_Cell:
366 {
367   \c_math_toggle_token
368   \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”).

```

369   \dim_gset:Nn \g_@@_max_cell_width_dim
370   { \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”.

```

371   \@@_update_for_first_and_last_row:

```

Now, we can create the Tikz node of the cell.

```

372   \tikz
373   [
374     remember~picture ,
375     inner~sep = \c_zero_dim ,
376     minimum~width = \c_zero_dim ,
377     baseline
378   ]
379   \node
380   [
381     anchor = base ,
382     name = nm - \int_use:N \g_@@_env_int -
383             \int_use:N \c@iRow -
384             \int_use:N \c@jCol ,
385     alias =
386       \str_if_empty:NF \l_@@_name_str
387       {
388         \l_@@_name_str -
389         \int_use:N \c@iRow -
390         \int_use:N \c@jCol
391       }
392   ]
393   { \box_use_drop:N \l_@@_cell_box } ;
394 }
395 \cs_generate_variant:Nn \dim_set:Nn { N x }

```

In the environment `{NiceArrayWithDelims}`, we will have to redefine the column types `w` and `W`. The redefinition of these two column types are very close and that’s why we use a macro `\@@_renewcolumnntype:nn`. The first argument is the type of the column (`w` or `W`) and the second argument is a code inserted at a special place and which is the only difference between the two definitions.

```

396 \cs_new_protected:Npn \@@_renewcolumnntype:nn #1 #2
397 {
398   \newcolumnntype #1 [ 2 ]
399   {
400     > {
401       \hbox_set:Nw \l_@@_cell_box
402       \@@_Cell:
403     }
404     c
405     < {
406       \@@_end_Cell:
407       \hbox_set_end:
408       #2
409       \makebox [ ##2 ] [ ##1 ] { \box_use_drop:N \l_@@_cell_box }
410     }
411   }
412 }

```

The argument of the following command `\@@_instruction_of_type:n` defined below is the type of the instruction (`Cdots`, `Vdots`, `Ddots`, etc.). 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 & & \\
\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:nn {2}{2}
\@@_draw_Cdots:nn {3}{2}
```

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

```
413 \bool_if:NTF \c_@@_draft_bool
414 { \cs_set_protected:Npn \@@_instruction_of_type:n #1 { } }
415 {
416   \cs_new_protected:Npn \@@_instruction_of_type:n #1
417   {
```

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).

```
418     \tl_gput_right:cx
419     { \g_@@_#1_lines_tl }
420     {
421       \use:c { @@_draw_#1 : nn }
422       { \int_use:N \c@iRow }
423       { \int_use:N \c@jCol }
424     }
425   }
426 }
```

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`.

```
427 \cs_new_protected:Npn \@@_array:
428 {
429   \bool_if:NTF \c_@@_revtex_bool
430   {
431     \cs_set_eq:NN \@acoll \@arrayacol
432     \cs_set_eq:NN \@acolr \@arrayacol
433     \cs_set_eq:NN \@acol \@arrayacol
434     \cs_set:Npn \@halignto { }
435     \@array@array
436   }
437   \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.

```
438   [ \str_if_eq:VnTF \l_@@_baseline_str c c t ]
439 }
```

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.

```
440 \cs_set_eq:NN \@@_standard_ialign: \ialign
```

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

```
441 \cs_new:Npn \@@_everycr: { \noalign { \@@_everycr_i: } }
442 \cs_new_protected:Npn \@@_everycr_i:
443 {
444   \int_gzero:N \c@jCol
```

The `\hbox:n` is mandatory.

```

445 \hbox:n
446 {
447   \tikz [ remember-picture ]
448   \coordinate
449   ( nm - \int_use:N \g_@@_env_int - row - \int_eval:n { \c@iRow + 1 } ) ;
450 }

```

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

```

451 \bool_if:NT \l_@@_hlines_bool
452 {

```

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.

```

453   \int_compare:nNnT \c@iRow > { -1 }
454   {
455     \bool_if:NF \g_@@_row_of_col_done_bool
456     {
457       \int_compare:nNnF \c@iRow = \l_@@_last_row_int
458       {
459         \bool_if:NTF \c_@@_colortbl_loaded_bool
460         { { \CT@arc@ \hrule height \arrayrulewidth } }
461         { \hrule height \arrayrulewidth }
462       }
463     }
464   }
465 }
466 }

```

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

```

467 \cs_new_protected:Npn \@@_pre_array:
468 {
469   \box_clear_new:N \l_@@_cell_box
470   \cs_if_exist:NT \theiRow
471   { \int_set_eq:NN \l_@@_save_iRow_int \c@iRow }
472   \int_gzero_new:N \c@iRow
473   \cs_if_exist:NT \thejCol
474   { \int_set_eq:NN \l_@@_save_jCol_int \c@jCol }
475   \int_gzero_new:N \c@jCol
476   \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}`).

```

477   \bool_if:NT \l_@@_small_bool
478   {
479     \cs_set:Npn \arraystretch { 0.47 }
480     \dim_set:Nn \arraycolsep { 1.45 pt }
481   }

```

We switch to a global version of the `\l_@@_medium_nodes_bool` and `\l_@@_large_nodes_bool` because these booleans may be raised in cells of the array (for exemple in commands `\Block`).

```

482   \bool_gset_eq:NN \g_@@_medium_nodes_bool \l_@@_medium_nodes_bool
483   \bool_gset_eq:NN \g_@@_large_nodes_bool \l_@@_large_nodes_bool

```

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`.

```

484   \cs_set:Npn \ialign
485   {
486     \bool_if:NTF \c_@@_colortbl_loaded_bool
487     {
488       \CT@everycr

```

```

489         {
490             \noalign { \cs_gset_eq:NN \CT@row@color \prg_do_nothing: }
491             \@@_everycr:
492         }
493     }
494     { \everycr { \@@_everycr: } }
495     \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>30</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`.

```

496     \dim_gzero_new:N \g_@@_dp_row_zero_dim
497     \dim_gset:Nn \g_@@_dp_row_zero_dim { \box_dp:N \@arstrutbox }
498     \dim_gzero_new:N \g_@@_ht_row_zero_dim
499     \dim_gset:Nn \g_@@_ht_row_zero_dim { \box_ht:N \@arstrutbox }
500     \dim_gzero_new:N \g_@@_ht_row_one_dim
501     \dim_gset:Nn \g_@@_ht_row_one_dim { \box_ht:N \@arstrutbox }
502     \dim_gzero_new:N \g_@@_dp_ante_last_row_dim
503     \dim_gzero_new:N \g_@@_ht_last_row_dim
504     \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
505     \dim_gzero_new:N \g_@@_dp_last_row_dim
506     \dim_gset:Nn \g_@@_dp_last_row_dim { \box_dp:N \@arstrutbox }

```

After its first use, the definition of `\ialign` will revert automatically to its default definition. With this programming, we will have, in the cells of the array, a clean version of `\ialign`.<sup>31</sup>

```

507     \cs_set_eq:NN \ialign \@@_standard_ialign:
508     \halign
509 }

```

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}`.

```

510     \newcolumntype L { > \@@_Cell: l < \@@_end_Cell: }
511     \newcolumntype C { > \@@_Cell: c < \@@_end_Cell: }
512     \newcolumntype R { > \@@_Cell: r < \@@_end_Cell: }
513     \cs_set_eq:NN \firsthline \hline
514     \cs_set_eq:NN \lasthline \hline
515     \cs_set_eq:NN \Ldots \@@_Ldots
516     \cs_set_eq:NN \Cdots \@@_Cdots
517     \cs_set_eq:NN \Vdots \@@_Vdots
518     \cs_set_eq:NN \Ddots \@@_Ddots
519     \cs_set_eq:NN \Iddots \@@_Iddots
520     \cs_set_eq:NN \hdottedline \@@_hdottedline:
521     \cs_set_eq:NN \Hspace \@@_Hspace:
522     \cs_set_eq:NN \Hdotsfor \@@_Hdotsfor:
523     \cs_set_eq:NN \multicolumn \@@_multicolumn:nnn
524     \cs_set_eq:NN \Block \@@_Block:
525     \cs_set_eq:NN \rotate \@@_rotate:
526     \cs_set_eq:NN \OnlyMainNiceMatrix \@@_OnlyMainNiceMatrix:n
527     \bool_if:NT \l_@@_renew_dots_bool
528     {
529         \cs_set_eq:NN \ldots \@@_Ldots
530         \cs_set_eq:NN \cdots \@@_Cdots
531         \cs_set_eq:NN \vdots \@@_Vdots
532         \cs_set_eq:NN \ddots \@@_Ddots
533         \cs_set_eq:NN \iddots \@@_Iddots
534         \cs_set_eq:NN \dots \@@_Ldots
535         \cs_set_eq:NN \hdotsfor \@@_Hdotsfor:
536     }

```

<sup>30</sup>The option `small` of `nicematrix` changes (among other) the value of `\arraystretch`. This is done, of course, before the call of `{array}`.

<sup>31</sup>The user will probably not employ directly `\ialign` in the array... but more likely environments that utilize `\ialign` internally (e.g.: `{substack}`).

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

```
537 \seq_gclear_new:N \g_@@_multicolumn_cells_seq
538 \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).

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

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

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

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

The counter `\c@jCol` will be used to count the columns of the array. Since we want to know the total number of columns of the matrix, we also create a counter `\g_@@_col_total_int`. These counters are updated in the command `\@@_Cell`: executed at the beginning of each cell.

```
541 \int_gzero_new:N \g_@@_col_total_int
542 \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:`.

```
543 \cs_set_eq:NN \NC@find@w \relax
544 \cs_set_eq:NN \NC@find@W \relax
545 \@@_renewcolumnntype:nn w { }
546 \@@_renewcolumnntype:nn W { \cs_set_eq:NN \hss \hfil }
```

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).

```
547 \tl_set_rescan:Nno
548 \l_@@_letter_for_dotted_lines_str { } \l_@@_letter_for_dotted_lines_str
549 \exp_args:NV \newcolumnntype \l_@@_letter_for_dotted_lines_str
550 {
551   !
552 }
```

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).

```
553 \int_compare:nNnF \c@iRow = 0
554 {
555   \int_compare:nNnF \c@iRow = \l_@@_last_row_int
556   { \skip_horizontal:n { 2 \l_@@_radius_dim } }
557 }
```

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`.

```
558 \int_compare:nNnT \c@jCol > \g_@@_last_vdotted_col_int
```



```

559         {
560             \int_gset_eq:NN \g_@@_last_vdotted_col_int \c@jCol
561             \tl_gput_right:Nx \g_@@_code_after_tl

```

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

```

562         { \@@_vdottedline:n { \int_use:N \c@jCol } }
563     }
564 }
565 }
566 \int_gzero_new:N \g_@@_last_vdotted_col_int
567 \bool_if:NT \c_@@_siunitx_loaded_bool \@@_renew_NC@rewrite@S:
568 \int_gset:Nn \g_@@_last_vdotted_col_int { -1 }
569 \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.

```

570 \tl_gclear_new:N \g_@@_Cdots_lines_tl
571 \tl_gclear_new:N \g_@@_Ldots_lines_tl
572 \tl_gclear_new:N \g_@@_Vdots_lines_tl
573 \tl_gclear_new:N \g_@@_Ddots_lines_tl
574 \tl_gclear_new:N \g_@@_Iddots_lines_tl
575 \tl_gclear_new:N \g_@@_Hdotsfor_lines_tl
576 }

```

## The environment `{NiceArrayWithDelims}`

```

577 \NewDocumentEnvironment { NiceArrayWithDelims } { m m O { } m ! O { } }
578 {
579     \bool_gset_false:N \g_@@_row_of_col_done_bool
580     \str_if_empty:NT \g_@@_name_env_str
581     { \str_gset:Nn \g_@@_name_env_str { NiceArrayWithDelims } }
582     \@@_adapt_S_column:
583     \@@_test_if_math_mode:
584     \bool_if:NT \l_@@_in_env_bool { \@@_fatal:n { Yet~in~env } }
585     \bool_set_true:N \l_@@_in_env_bool

```

We deactivate Tikz externalization.

Since we use Tikz pictures with the options `overlay` and `remember picture`, there would be errors).

```

586 \cs_if_exist:NT \tikz@library@external@loaded
587 {
588     \tikzset { external / export = false }
589     \cs_if_exist:NT \ifstandalone
590     { \tikzset { external / optimize = false } }
591 }

```

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

```

592 \int_gincr:N \g_@@_env_int
593 \bool_if:NF \l_@@_block_auto_columns_width_bool
594 { \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.

```

595 \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` and `b`.

```

596 \bool_if:NFT \l_@@_NiceArray_bool
597 { \keys_set:nn { NiceMatrix / NiceArray } }
598 { \keys_set:nn { NiceMatrix / pNiceArray } }
599 { #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).

```

600 \int_compare:nNnT \l_@@_last_row_int > { -2 }

```

```

601 {
602   \tl_put_right:Nn \@@_update_for_first_and_last_row:
603   {
604     \dim_gset:Nn \g_@@_ht_last_row_dim
605     { \dim_max:nn \g_@@_ht_last_row_dim { \box_ht:N \l_@@_cell_box } }
606     \dim_gset:Nn \g_@@_dp_last_row_dim
607     { \dim_max:nn \g_@@_dp_last_row_dim { \box_dp:N \l_@@_cell_box } }
608   }
609 }
610 \int_compare:nNnT \l_@@_last_row_int = { -1 }
611 {
612   \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.

```

613   \str_if_empty:NTF \l_@@_name_str
614   {
615     \cs_if_exist:cT { @@_last_row_ \int_use:N \g_@@_env_int }
616     {
617       \int_set:Nn \l_@@_last_row_int
618       { \use:c { @@_last_row_ \int_use:N \g_@@_env_int } }
619     }
620   }
621   {
622     \cs_if_exist:cT { @@_last_row_ \l_@@_name_str }
623     {
624       \int_set:Nn \l_@@_last_row_int
625       { \use:c { @@_last_row_ \l_@@_name_str } }
626     }
627   }
628 }

```

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

```

629 \@@_pre_array:

```

We compute the width of the two delimiters.

```

630 \dim_zero_new:N \l_@@_left_delim_dim
631 \dim_zero_new:N \l_@@_right_delim_dim
632 \bool_if:NTF \l_@@_NiceArray_bool
633 {
634   \dim_gset:Nn \l_@@_left_delim_dim { 2 \arraycolsep }
635   \dim_gset:Nn \l_@@_right_delim_dim { 2 \arraycolsep }
636 }
637 {
638   \hbox_set:Nn \l_tmpa_box
639   {
640     \c_math_toggle_token
641     \left #1 \vcenter to 3 cm { } \right.
642     \c_math_toggle_token
643   }
644   \dim_set:Nn \l_@@_left_delim_dim
645   { \box_wd:N \l_tmpa_box - \nulldelimiterspace }
646   \hbox_set:Nn \l_tmpa_box
647   {
648     \c_math_toggle_token
649     \left. \vcenter to 3 cm { } \right #2
650     \c_math_toggle_token
651   }
652   \dim_set:Nn \l_@@_right_delim_dim
653   { \box_wd:N \l_tmpa_box - \nulldelimiterspace }
654 }

```

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

```

655 \box_clear_new:N \l_@@_the_array_box

```

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

```

656 \tl_set:Nn \l_tmpa_tl { #4 }
657 \int_compare:nNnTF \l_@@_first_col_int = 0
658 { \tl_put_left:NV \l_tmpa_tl \c_@@_preamble_first_col_tl }
659 {
660   \bool_if:nT
661   {
662     \l_@@_NiceArray_bool
663     && ! \l_@@_vlines_bool
664     && ! \l_@@_exterior_arraycolsep_bool
665   }
666   { \tl_put_left:Nn \l_tmpa_tl { @ { } } }
667 }
668 \int_compare:nNnTF \l_@@_last_col_int > { -1 }
669 { \tl_put_right:NV \l_tmpa_tl \c_@@_preamble_last_col_tl }
670 {
671   \bool_if:nT
672   {
673     \l_@@_NiceArray_bool
674     && ! \l_@@_vlines_bool
675     && ! \l_@@_exterior_arraycolsep_bool
676   }
677   { \tl_put_right:Nn \l_tmpa_tl { @ { } } }
678 }
679 \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).

```

680 \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.

```

681 \bool_if:NT \l_@@_vlines_bool
682 {
683   \dim_add:Nn \arraycolsep { 0.5 \arrayrulewidth }
684   \skip_horizontal:n { 0.5 \arrayrulewidth }
685 }
686 \skip_horizontal:n \l_@@_left_margin_dim
687 \skip_horizontal:n \l_@@_extra_left_margin_dim
688 \c_math_toggle_token
689 \bool_if:NTF \l_@@_light_syntax_bool
690 { \begin { @@-light-syntax } }
691 { \begin { @@-normal-syntax } }
692 }
693 {
694   \bool_if:NTF \l_@@_light_syntax_bool
695   { \end { @@-light-syntax } }
696   { \end { @@-normal-syntax } }
697   \c_math_toggle_token
698   \skip_horizontal:n \l_@@_right_margin_dim
699   \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.

```

700 \bool_if:NT \l_@@_vlines_bool { \skip_horizontal:n { 0.5 \arrayrulewidth } }
701 \hbox_set_end:

```

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

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

```

702 \int_compare:nNnTF \l_@@_last_row_int > { -2 }
703 {
704   \bool_if:NF \l_@@_last_row_without_value_bool
705   {

```

```

706         \int_compare:nNnF \l_@@_last_row_int = \c@iRow
707         {
708             \@@_error:n { Wrong~last~row }
709             \int_gset_eq:NN \l_@@_last_row_int \c@iRow
710         }
711     }
712 }

```

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>32</sup>

```

713     \int_gset_eq:NN \c@jCol \g_@@_col_total_int
714     \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.

```

715     \int_gset_eq:NN \g_@@_row_total_int \c@iRow
716     \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. 50).

```

717     \int_compare:nNnT \l_@@_first_col_int = 0
718     {
719         \skip_horizontal:n \arraycolsep
720         \skip_horizontal:n \g_@@_width_first_col_dim
721     }

```

The construction of the real box is different in `{NiceArray}` and in its variants (`{pNiceArray}`, etc.) because, in `{NiceArray}`, we have to take into account the option of position (`t`, `c` or `b`). We begin with `{NiceArray}`.

```

722     \bool_if:NTF \l_@@_NiceArray_bool
723     {

```

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`.

```

724         \str_if_eq:VnTF \l_@@_baseline_str { b }
725         {
726             \begin { tikzpicture }
727                 \@@_extract_coords:w ( nm - \int_use:N \g_@@_env_int - row - 1 )
728                 \dim_gset_eq:NN \g_tmpa_dim \pgf@y
729                 \@@_extract_coords:w
730                 (
731                     nm - \int_use:N \g_@@_env_int -
732                     row - \int_use:N \c@iRow - base
733                 )
734                 \dim_gsub:Nn \g_tmpa_dim \pgf@y
735             \end { tikzpicture }
736             \int_compare:nNnT \l_@@_first_row_int = 0
737             {
738                 \dim_gadd:Nn \g_tmpa_dim
739                 { \g_@@_ht_row_zero_dim + \g_@@_dp_row_zero_dim }
740             }
741             \box_move_up:nn \g_tmpa_dim { \box_use_drop:N \l_@@_the_array_box }
742         }
743         {
744             \str_if_eq:VnTF \l_@@_baseline_str { c }
745             { \box_use_drop:N \l_@@_the_array_box }
746             {

```

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

```

747                 \str_if_eq:VnT \l_@@_baseline_str { t }
748                 { \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`.

```

749                 \int_set:Nn \l_tmpa_int \l_@@_baseline_str
750                 \bool_if:nT
751                 {

```

---

<sup>32</sup>We remind that the potential “first column” has the number 0.

```

752         \int_compare:nNn \l_tmpa_int < 0
753         || \int_compare:nNn \l_tmpa_int > \g_@@_row_total_int
754     }
755     {
756         \@@_error:n { bad-value-for-baseline }
757         \int_set:Nn \l_tmpa_int 1
758     }

```

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

```

759     \begin { tikzpicture }
760     \@@_extract_coords:w
761     ( nm - \int_use:N \g_@@_env_int - row - 1 )
762     \dim_gset_eq:NN \g_tmpa_dim \pgf@y
763     \@@_extract_coords:w
764     (
765         nm - \int_use:N \g_@@_env_int -
766         row - \int_use:N \l_tmpa_int- base
767     )
768     \dim_gsub:Nn \g_tmpa_dim \pgf@y
769     \end { tikzpicture }
770     \int_compare:nNnT \l_@@_first_row_int = 0
771     {
772         \dim_gadd:Nn \g_tmpa_dim
773         { \g_@@_ht_row_zero_dim + \g_@@_dp_row_zero_dim }
774     }
775     \box_move_up:nn \g_tmpa_dim
776     { \box_use_drop:N \l_@@_the_array_box }
777 }
778 }
779 }

```

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).

```

780 {
781     \int_compare:nNnTF \l_@@_first_row_int = 0
782     {
783         \dim_set:Nn \l_tmpa_dim
784         { \g_@@_dp_row_zero_dim + \g_@@_ht_row_zero_dim }
785     }
786     { \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>33</sup>

```

787     \int_compare:nNnTF \l_@@_last_row_int > { -2 }
788     {
789         \dim_set:Nn \l_tmpb_dim
790         { \g_@@_ht_last_row_dim + \g_@@_dp_last_row_dim }
791     }
792     { \dim_zero:N \l_tmpb_dim }
793     \hbox_set:Nn \l_tmpa_box
794     {
795         \c_math_toggle_token
796         \left #1
797         \vcenter
798         {

```

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

```

799         \skip_vertical:n { - \l_tmpa_dim }
800         \hbox:n
801         {
802             \skip_horizontal:n { - \arraycolsep }
803             \box_use_drop:N \l_@@_the_array_box
804             \skip_horizontal:n { - \arraycolsep }
805         }

```

<sup>33</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`).

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

```

806         \skip_vertical:n { - \l_tmpb_dim }
807     }
808     \right #2
809     \c_math_toggle_token
810 }

```

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.

```

811     \bool_if:NTF \l_@@_max_delimiter_width_bool
812     { \@@_put_box_in_flow_bis:nn { #1 } { #2 } }
813     \@@_put_box_in_flow:
814 }

```

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. 51).

```

815     \bool_if:NT \g_@@_last_col_found_bool
816     { \skip_horizontal:n { \g_@@_width_last_col_dim + \arraycolsep } }
817     \@@_after_array:
818 }

```

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`).

```

819 \cs_new_protected:Npn \@@_put_box_in_flow:
820 {
821     \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }
822     \box_set_dp:Nn \l_tmpa_box { \box_dp:N \l_tmpa_box + \l_tmpb_dim }
823     \box_use_drop:N \l_tmpa_box
824 }

```

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.

```

825 \cs_new_protected:Npn \@@_put_box_in_flow_bis:nn #1 #2
826 {

```

We will compute the real width of both delimiters used.

```

827     \dim_zero_new:N \l_@@_real_left_delim_dim
828     \dim_zero_new:N \l_@@_real_right_delim_dim
829     \hbox_set:Nn \l_tmpb_box
830     {
831         \c_math_toggle_token
832         \left #1
833         \vcenter
834         {
835             \vbox_to_ht:nn
836             { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
837             { }
838         }
839         \right .
840         \c_math_toggle_token
841     }
842     \dim_set:Nn \l_@@_real_left_delim_dim
843     { \box_wd:N \l_tmpb_box - \nulldelimiterspace }
844     \hbox_set:Nn \l_tmpb_box
845     {
846         \c_math_toggle_token
847         \left .
848         \vbox_to_ht:nn
849         { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
850         { }

```

```

851     \right #2
852     \c_math_toggle_token
853   }
854   \dim_set:Nn \l_@@_real_right_delim_dim
855     { \box_wd:N \l_tmpb_box - \nullldelimiterspace }

```

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

```

856   \skip_horizontal:n { \l_@@_left_delim_dim - \l_@@_real_left_delim_dim }
857   \@@_put_box_in_flow:
858   \skip_horizontal:n { \l_@@_right_delim_dim - \l_@@_real_right_delim_dim }
859 }

```

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).

```

860 \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).

```

861 {
862   \peek_meaning_ignore_spaces:NTF \end
863   { \@@_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`).

```

864   { \exp_args:NV \@@_array: \l_tmpa_tl }
865 }
866 {

```

If all the columns must have the same width (if the user has used the option `columns-width` or the option `auto-columns-width` of the environment `{NiceMatrixBlock}`), we will add a row in the array to fix the width of the columns and construct the “col” nodes `nm-a-col-j` (these nodes will be used by the horizontal open dotted lines and by the commands `\@@_vdottedline:n`). We have written a dedicated function for that job.

```

867   \@@_create_col_nodes:
868   \endarray
869 }

```

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`).

```

870 \NewDocumentEnvironment { @@-light-syntax } { b }
871 {

```

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`.

```

872   \tl_if_empty:nT { #1 } { \@@_fatal:n { empty-environment } }

```

Here is the call to `\array` (we have a dedicated macro `\@@_array:` because of compatibility with the classes `revtex4-1` and `revtex4-2`).

```

873   \exp_args:NV \@@_array: \l_tmpa_tl

```

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

```

874   \seq_gclear_new:N \g_@@_rows_seq
875   \tl_set_rescan:Nno \l_@@_end_of_row_tl { } \l_@@_end_of_row_tl
876   \exp_args:NNV \seq_gset_split:Nnn \g_@@_rows_seq \l_@@_end_of_row_tl { #1 }

```

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

```

877   \seq_gpop_left:NN \g_@@_rows_seq \l_tmpa_tl
878   \exp_args:NV \@@_line_with_light_syntax_i:n \l_tmpa_tl
879   \seq_map_function:NN \g_@@_rows_seq \@@_line_with_light_syntax:n
880   \@@_create_col_nodes:
881   \endarray
882 }

```

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`.

```

883 { }

```

```

884 \cs_new_protected:Npn \@@_line_with_light_syntax_i:n #1
885 {
886   \seq_gclear_new:N \g_@@_cells_seq
887   \seq_gset_split:Nnn \g_@@_cells_seq { ~ } { #1 }
888   \seq_gpop_left:NN \g_@@_cells_seq \l_tmpa_tl
889   \l_tmpa_tl
890   \seq_map_function:NN \g_@@_cells_seq \@@_cell_with_light_syntax:n
891 }

892 \cs_new_protected:Npn \@@_line_with_light_syntax:n #1
893 {
894   \tl_if_empty:nF { #1 }
895   { \ \ \@@_line_with_light_syntax_i:n { #1 } }
896 }

897 \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).

```

898 \cs_new_protected:Npn \@@_analyze_end:Nn #1 #2
899 {
900   \str_if_eq:VnT \g_@@_name_env_str { #2 }
901   { \@@_fatal:n { empty~environment } }

```

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

```

902   \end { #2 }
903 }

```

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).

```

904 \cs_new:Npn \@@_create_col_nodes:
905 {
906   \crrc
907   \int_compare:nNnT \c@iRow = 0 { \@@_fatal:n { Zero~row } }
908   \int_compare:nNnT \l_@@_first_col_int = 0 { \omit & }
909   \omit

```

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

```

910   \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`).

```

911   \tikz [ remember~picture ]
912   \coordinate ( nm - \int_use:N \g_@@_env_int - col - 1 ) ;

```

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.

```

913   \bool_if:nTF
914   {
915     \l_@@_auto_columns_width_bool
916     || \dim_compare_p:nNn \l_@@_columns_width_dim > \c_zero_dim
917   }
918   {
919     \bool_if:nTF
920     {
921       \l_@@_auto_columns_width_bool
922       && ! \l_@@_block_auto_columns_width_bool
923     }
924     {
925       \skip_gset:Nn \g_tmpa_skip
926       { \g_@@_max_cell_width_dim + 2 \arraycolsep }
927     }
928     {
929       \skip_gset:Nn \g_tmpa_skip

```



```

930         { \l_@@_columns_width_dim + 2 \arraycolsep }
931     }
932 }
933 { \skip_gset:Nn \g_tmpa_skip { 0 pt~plus 1 fill } }
934 \skip_horizontal:N \g_tmpa_skip
935 \hbox:n
936 {
937     \tikz [ remember~picture ]
938     \coordinate ( nm - \int_use:N \g_@@_env_int - col - 2 ) ;
939 }

```

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.

```

940 \int_gset:Nn \g_tmpa_int 1
941 \bool_if:NTF \g_@@_last_col_found_bool
942 { \prg_replicate:nn { \g_@@_col_total_int - 2 } }
943 { \prg_replicate:nn { \g_@@_col_total_int - 1 } }
944 {
945     &
946     \omit

```

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

```

947     \int_gincr:N \g_tmpa_int
948     \skip_horizontal:N \g_tmpa_skip

```

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

```

949     \tikz [ remember~picture ]
950     \coordinate
951     (
952         nm - \int_use:N \g_@@_env_int -
953         col - \int_eval:n { \g_tmpa_int + 1 }
954     ) ;
955 }
956 \cr
957 }

```

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

```

958 \tl_const:Nn \c_@@_preamble_first_col_tl
959 {
960     >
961     {
962         \@@_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.

```

963     \hbox_set:Nw \l_@@_cell_box
964     \c_math_toggle_token
965     \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”.

```

966     \bool_if:nT
967     {
968         \int_compare_p:nNn \c@iRow > 0
969         &&
970         (
971             \int_compare_p:nNn \l_@@_last_row_int < 0
972             ||
973             \int_compare_p:nNn \c@iRow < \l_@@_last_row_int
974         )
975     }
976     { \l_@@_code_for_first_col_tl }
977 }

```

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`.

```

978     l

```

```

979 <
980 {
981   \c_math_toggle_token
982   \hbox_set_end:
983   \@@_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.

```

984   \dim_gset:Nn \g_@@_width_first_col_dim
985   {
986     \dim_max:nn
987     \g_@@_width_first_col_dim
988     { \box_wd:N \l_@@_cell_box }
989   }

```

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

```

990   \hbox_overlap_left:n
991   {
992     \tikz
993     [
994       remember~picture ,
995       inner~sep = \c_zero_dim ,
996       minimum~width = \c_zero_dim ,
997       baseline
998     ]
999     \node
1000     [
1001       anchor = base ,
1002       name =
1003         nm -
1004         \int_use:N \g_@@_env_int -
1005         \int_use:N \c@iRow -
1006         0 ,
1007       alias =
1008         \str_if_empty:NF \l_@@_name_str
1009         {
1010           \l_@@_name_str -
1011           \int_use:N \c@iRow -
1012           0
1013         }
1014     ]
1015     { \box_use_drop:N \l_@@_cell_box } ;
1016     \skip_horizontal:n
1017     {
1018       \l_@@_left_delim_dim +
1019       \l_@@_left_margin_dim +
1020       \l_@@_extra_left_margin_dim
1021     }
1022   }
1023   \skip_horizontal:n { - 2 \arraycolsep }
1024 }
1025 }

```

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

```

1026 \tl_const:Nn \c_@@_preamble_last_col_tl
1027 {
1028   >
1029   {

```

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

```

1030   \bool_gset_true:N \g_@@_last_col_found_bool
1031   \int_gincr:N \c@jCol
1032   \int_gset:Nn \g_@@_col_total_int
1033   { \int_max: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.

```

1034   \hbox_set:Nw \l_@@_cell_box

```

```

1035         \c_math_toggle_token
1036         \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".
1037         \bool_if:nT
1038         {
1039             \int_compare_p:nNn \c@iRow > 0
1040             &&
1041             (
1042                 \int_compare_p:nNn \l_@@_last_row_int < 0
1043                 ||
1044                 \int_compare_p:nNn \c@iRow < \l_@@_last_row_int
1045             )
1046         }
1047         { \l_@@_code_for_last_col_tl }
1048     }
1049     l
1050     <
1051     {
1052         \c_math_toggle_token
1053         \hbox_set_end:
1054         \@@_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.

```

1055         \dim_gset:Nn \g_@@_width_last_col_dim
1056         {
1057             \dim_max:nn
1058             \g_@@_width_last_col_dim
1059             { \box_wd:N \l_@@_cell_box }
1060         }
1061         \skip_horizontal:n { - 2 \arraycolsep }
The content of the cell is inserted in an overlapping position.
1062         \hbox_overlap_right:n
1063         {
1064             \skip_horizontal:n
1065             {
1066                 \l_@@_right_delim_dim +
1067                 \l_@@_right_margin_dim +
1068                 \l_@@_extra_right_margin_dim
1069             }
1070             \tikz
1071             [
1072                 remember~picture ,
1073                 inner~sep = \c_zero_dim ,
1074                 minimum~width = \c_zero_dim ,
1075                 baseline
1076             ]
1077             \node
1078             [
1079                 anchor = base ,
1080                 name =
1081                 nm -
1082                 \int_use:N \g_@@_env_int -
1083                 \int_use:N \c@iRow -
1084                 \int_use:N \c@jCol ,
1085                 alias =
1086                 \str_if_empty:NF \l_@@_name_str
1087                 {
1088                     \l_@@_name_str -
1089                     \int_use:N \c@iRow -
1090                     \int_use:N \c@jCol
1091                 }
1092             ]
1093             { \box_use_drop:N \l_@@_cell_box } ;

```

```

1094     }
1095   }
1096 }

```

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.

```

1097 \NewDocumentEnvironment { NiceArray } { }
1098 {
1099   \bool_set_true:N \l_@@_NiceArray_bool
1100   \str_if_empty:NT \g_@@_name_env_str
1101     { \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).

```

1102   \NiceArrayWithDelims . .
1103 }
1104 { \endNiceArrayWithDelims }

```

We create the variants of the environment `{NiceArrayWithDelims}`. These variants exist since the version 3.0 of `nicematrix`.

```

1105 \NewDocumentEnvironment { pNiceArray } { }
1106 {
1107   \str_if_empty:NT \g_@@_name_env_str
1108     { \str_gset:Nn \g_@@_name_env_str { pNiceArray } }
1109   \@@_test_if_math_mode:
1110   \NiceArrayWithDelims ( )
1111 }
1112 { \endNiceArrayWithDelims }

1113 \NewDocumentEnvironment { bNiceArray } { }
1114 {
1115   \str_if_empty:NT \g_@@_name_env_str
1116     { \str_gset:Nn \g_@@_name_env_str { bNiceArray } }
1117   \@@_test_if_math_mode:
1118   \NiceArrayWithDelims [ ]
1119 }
1120 { \endNiceArrayWithDelims }

1121 \NewDocumentEnvironment { BNiceArray } { }
1122 {
1123   \str_if_empty:NT \g_@@_name_env_str
1124     { \str_gset:Nn \g_@@_name_env_str { BNiceArray } }
1125   \@@_test_if_math_mode:
1126   \NiceArrayWithDelims \{ \}
1127 }
1128 { \endNiceArrayWithDelims }

1129 \NewDocumentEnvironment { vNiceArray } { }
1130 {
1131   \str_if_empty:NT \g_@@_name_env_str
1132     { \str_gset:Nn \g_@@_name_env_str { vNiceArray } }
1133   \@@_test_if_math_mode:
1134   \NiceArrayWithDelims | |
1135 }
1136 { \endNiceArrayWithDelims }

1137 \NewDocumentEnvironment { VNiceArray } { }
1138 {
1139   \str_if_empty:NT \g_@@_name_env_str
1140     { \str_gset:Nn \g_@@_name_env_str { VNiceArray } }
1141   \@@_test_if_math_mode:
1142   \NiceArrayWithDelims \| \|
1143 }
1144 { \endNiceArrayWithDelims }

```

## The environment {NiceMatrix} and its variants

```

1145 \cs_new_protected:Npn \@@_define_env:n #1
1146 {
1147   \NewDocumentEnvironment { #1 NiceMatrix } { ! O { } }
1148   {
1149     \str_gset:Nn \g_@@_name_env_str { #1 NiceMatrix }
1150     \keys_set:nn { NiceMatrix / NiceMatrix } { ##1 }
1151     \begin { #1 NiceArray }
1152       {
1153         *
1154         {
1155           \int_compare:nNnTF \l_@@_last_col_int = { -1 }
1156             \c@MaxMatrixCols
1157             { \int_eval:n { \l_@@_last_col_int - 1 } }
1158         }
1159         C
1160       }
1161     }
1162     { \end { #1 NiceArray } }
1163   }
1164 \@@_define_env:n { }
1165 \@@_define_env:n p
1166 \@@_define_env:n b
1167 \@@_define_env:n B
1168 \@@_define_env:n v
1169 \@@_define_env:n V

```

## How to know whether a cell is “empty”

The conditionnal `\@@_if_not_empty_cell:nnT` tests whether a cell is empty. The first two arguments must be LaTeX3 counters for the row and the column of the considered cell.

```

1170 \prg_set_conditional:Npnn \@@_if_not_empty_cell:nn #1 #2 { T , TF }
1171 {

```

First, we want to test whether the cell is in the virtual sequence of “non-empty” cells. There are several important remarks:

- we don’t use a `expl3` sequence for efficiency;
- the “non-empty” cells in this sequence are not, in fact, all the non-empty cells of the array: on the contrary they are only cells declared as non-empty for a special reason (as of now, there are only cells which are on a dotted line which is already drawn or which will be drawn “just after”);
- the flag `\l_tmpa_bool` will be raised when the cell is actually on this virtual sequence.

```

1172   \bool_set_false:N \l_tmpa_bool
1173   \cs_if_exist:cTF
1174     { @@ _ dotted _ \int_use:N #1 - \int_use:N #2 }
1175     \prg_return_true:
1176     {

```

We know that the cell is not in the virtual sequence of the “non-empty” cells. Now, we test whether the cell is a “virtual cell”, that is to say a cell after the `\` of the line of the array. It’s easy to know whether a cell is virtual: the cell is virtual if, and only if, the corresponding Tikz node doesn’t exist.

```

1177   \cs_if_free:cTF
1178   {
1179     pgf@sh@ns@nm -
1180     \int_use:N \g_@@_env_int -
1181     \int_use:N #1 -
1182     \int_use:N #2
1183   }
1184   { \prg_return_false: }
1185   {

```

Now, we want to test whether the cell is in the virtual sequence of “empty” cells. There are several important remarks:

- we don't use a `expl3` sequence for efficiency ;
- the “empty” cells in this sequence are not, in fact, all the non-empty cells of the array: on the contrary they are only cells declared as non-empty for a special reason ;
- the flag `\l_tmpa_bool` will be raised when the cell is actually on this virtual sequence.

```

1186         \bool_set_false:N \l_tmpa_bool
1187         \cs_if_exist:cT
1188         { @@ _ empty _ \int_use:N #1 - \int_use:N #2 }
1189         {
1190             \int_compare:nNnT
1191             { \use:c { @@ _ empty _ \int_use:N #1 - \int_use:N #2 } }
1192             =
1193             \g_@@_env_int
1194             { \bool_set_true:N \l_tmpa_bool }
1195         }
1196         \bool_if:NTF \l_tmpa_bool
1197         \prg_return_false:

```

In the general case, we consider the width of the Tikz node corresponding to the cell. In order to compute this width, we have to extract the coordinate of the west and east anchors of the node. This extraction needs a command environment `{pgfpicture}` but, in fact, nothing is drawn.

```

1198         {
1199             \begin { pgfpicture }

```

We store the name of the node corresponding to the cell in `\l_tmpa_tl`.

```

1200             \tl_set:Nx \l_tmpa_tl
1201             {
1202                 nm -
1203                 \int_use:N \g_@@_env_int -
1204                 \int_use:N #1 -
1205                 \int_use:N #2
1206             }
1207             \pgfpointanchor \l_tmpa_tl { east }
1208             \dim_gset:Nn \g_tmpa_dim \pgf@x
1209             \pgfpointanchor \l_tmpa_tl { west }
1210             \dim_gset:Nn \g_tmpb_dim \pgf@x
1211             \end { pgfpicture }
1212             \dim_compare:nNnTF
1213             { \dim_abs:n { \g_tmpb_dim - \g_tmpa_dim } } < { 0.5 pt }
1214             \prg_return_false:
1215             \prg_return_true:
1216         }
1217     }
1218 }
1219 }

```

## After the construction of the array

We deactivate Tikz externalization.

Since we use Tikz pictures with the options `overlay` and `remember picture`, there would be errors.

```

1220 \cs_new_protected:Npn \@@_after_array:
1221 {
1222     \group_begin:
1223     \cs_if_exist:NT \tikz@library@external@loaded
1224     { \tikzset { external / export = false } }

```

In the user has used the option `last-row` without value, we write in the `aux` file the number of that last row for the next run.

```

1225     \bool_if:NT \l_@@_last_row_without_value_bool
1226     {
1227         \iow_now:Nn \@mainaux \ExplSyntaxOn
1228         \iow_now:Nx \@mainaux

```

```

1229     {
1230         \cs_gset:cpn { @@_last_row_ \int_use:N \g_@@_env_int }
1231         { \int_use:N \g_@@_row_total_int }
1232     }

```

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.

```

1233     \str_if_empty:NF \l_@@_name_str
1234     {
1235         \iow_now:Nx \@mainaux
1236         {
1237             \cs_gset:cpn { @@_last_row_ \l_@@_name_str }
1238             { \int_use:N \g_@@_row_total_int }
1239         }
1240     }
1241     \iow_now:Nn \@mainaux \ExplSyntaxOff
1242 }

```

By default, the diagonal lines will be parallelized<sup>34</sup>. There are two types of diagonals lines: the `\Ddots` diagonals and the `\Ddots` 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.

```

1243     \bool_if:NT \l_@@_parallelize_diags_bool
1244     {
1245         \int_zero_new:N \l_@@_ddots_int
1246         \int_zero_new:N \l_@@_iddots_int

```

The dimensions `\l_@@_delta_x_one_dim` and `\l_@@_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 `\l_@@_delta_x_two_dim` and `\l_@@_delta_y_two_dim` are the  $\Delta_x$  and  $\Delta_y$  of the first `\Ddots` diagonal.

```

1247         \dim_zero_new:N \l_@@_delta_x_one_dim
1248         \dim_zero_new:N \l_@@_delta_y_one_dim
1249         \dim_zero_new:N \l_@@_delta_x_two_dim
1250         \dim_zero_new:N \l_@@_delta_y_two_dim
1251     }

```

The booleans `\g_@@_medium_nodes_bool` and `\g_@@_large_nodes_bool` may be raised directly in cells of the array (for example in commands `\Block`) but also because the user has used the options `create-medium-nodes` and `create-large-nodes` (these options raise `\l_@@_medium_nodes_bool` and `\l_@@_large_nodes_bool` but theses booleans are converted into the global versions before the creation of the array).

```

1252     \bool_if:nTF \g_@@_medium_nodes_bool
1253     {
1254         \bool_if:NTF \g_@@_large_nodes_bool
1255         \@@_create_medium_and_large_nodes:
1256         \@@_create_medium_nodes:
1257     }
1258     { \bool_if:NT \g_@@_large_nodes_bool \@@_create_large_nodes: }
1259     \int_zero_new:N \l_@@_initial_i_int
1260     \int_zero_new:N \l_@@_initial_j_int
1261     \int_zero_new:N \l_@@_final_i_int
1262     \int_zero_new:N \l_@@_final_j_int
1263     \bool_set_false:N \l_@@_initial_open_bool
1264     \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) are changed.

```

1265     \bool_if:NT \l_@@_small_bool
1266     {
1267         \dim_set:Nn \l_@@_radius_dim { 0.37 pt }
1268         \dim_set:Nn \l_@@_inter_dots_dim { 0.25 em }
1269     }

```

Now, we really draw the lines. The code to draw the lines has been constructed in the token lists `\g_@@_Vdots_lines_tl`, etc.

---

<sup>34</sup>It's possible to use the option `parallelize-diags` to disable this parallelization.

```

1270 \g_@@_Hdotsfor_lines_tl
1271 \g_@@_Vdots_lines_tl
1272 \g_@@_Ddots_lines_tl
1273 \g_@@_Iddots_lines_tl
1274 \g_@@_Cdots_lines_tl
1275 \g_@@_Ldots_lines_tl

```

Now, the code-after.

```

1276 \tikzset
1277 {
1278   every-picture / .style =
1279   {
1280     overlay ,
1281     remember-picture ,
1282     name-prefix = nm - \int_use:N \g_@@_env_int -
1283   }
1284 }
1285 \bool_if:NT \l_@@_vlines_bool \@@_draw_vlines:
1286 \cs_set_eq:NN \line \@@_line:nn
1287 \g_@@_code_after_tl
1288 \tl_gclear:N \g_@@_code_after_tl
1289 \group_end:
1290 \str_gclear:N \g_@@_name_env_str
1291 \@@_restore_iRow_jCol:
1292 }

```

```

1293 \cs_new_protected:Npn \@@_draw_vlines:
1294 {
1295   \group_begin:

```

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

```

1296 \bool_if:NT \c_@@_colortbl_loaded_bool \CT@arc@
1297 \begin { tikzpicture } [ line-width = \arrayrulewidth ]

```

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

```

1298 \@@_extract_coords:w ( row - 1 )
1299 \dim_set_eq:NN \l_tmpa_dim \pgf@y
1300 \@@_extract_coords:w ( row - \int_eval:n { \c@iRow + 1 } )
1301 \dim_set:Nn \l_tmpa_dim { \l_tmpa_dim - \pgf@y }

```

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

```

1302 \dim_zero:N \l_tmpb_dim
1303 \int_compare:nNnT \l_@@_last_row_int > { -1 }
1304 {
1305   \dim_set:Nn \l_tmpb_dim
1306   { \g_@@_dp_last_row_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”.

```

1307 \@@_extract_coords:w ( row - \int_eval:n { \c@iRow + 1 } )
1308 \dim_add:Nn \l_tmpa_dim \pgf@y
1309 \@@_extract_coords:w ( row - \int_eval:n { \g_@@_row_total_int + 1 } )
1310 \dim_sub:Nn \l_tmpa_dim \pgf@y
1311 \dim_sub:Nn \l_tmpa_dim \l_tmpb_dim
1312 }

```

Now, we can draw the lines with a loop.

```

1313 \int_step_inline:nnn
1314 { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
1315 { \bool_if:NTF \l_@@_NiceArray_bool { \c@jCol + 1 } \c@jCol }
1316 {
1317   \draw ( col - ##1 ) ++ ( 0 , \dim_use:N \l_tmpb_dim )
1318   -- ++ ( 0 , \dim_use:N \l_tmpa_dim ) ;
1319 }
1320 \end { tikzpicture }
1321 \group_end:
1322 }

```



```

1323 \cs_new_protected:Npn \@@_restore_iRow_jCol:
1324 {
1325   \cs_if_exist:NT \theiRow { \int_gset_eq:NN \c@iRow \l_@@_save_iRow_int }
1326   \cs_if_exist:NT \thejCol { \int_gset_eq:NN \c@jCol \l_@@_save_jCol_int }
1327 }

```

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.

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

For a closed extremity, we use the normal node and for an open one, we use the “medium node” or the `col`-node.

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

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.

```

1328 \cs_new_protected:Npn \@@_find_extremities_of_line:nnnn #1 #2 #3 #4
1329 {

```

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

```

1330   \cs_set:cpn { @@ _ dotted _ #1 - #2 } { }

```

Initialization of variables.

```

1331   \int_set:Nn \l_@@_initial_i_int { #1 }
1332   \int_set:Nn \l_@@_initial_j_int { #2 }
1333   \int_set:Nn \l_@@_final_i_int { #1 }
1334   \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.

```

1335   \bool_set_false:N \l_@@_stop_loop_bool
1336   \bool_do_until:Nn \l_@@_stop_loop_bool
1337   {
1338     \int_add:Nn \l_@@_final_i_int { #3 }
1339     \int_add:Nn \l_@@_final_j_int { #4 }

```

We test if we are still in the matrix.

```

1340     \bool_set_false:N \l_@@_final_open_bool
1341     \int_compare:nNnTF \l_@@_final_i_int > \c@iRow
1342     {
1343       \int_compare:nNnT { #3 } = 1
1344       { \bool_set_true:N \l_@@_final_open_bool }
1345     }
1346     {
1347       \int_compare:nNnTF \l_@@_final_j_int < 1
1348       {

```

```

1349         \int_compare:nNtT { #4 } = { -1 }
1350         { \bool_set_true:N \l_@@_final_open_bool }
1351     }
1352     {
1353         \int_compare:nNtT \l_@@_final_j_int > \c@jCol
1354         {
1355             \int_compare:nNtT { #4 } = 1
1356             { \bool_set_true:N \l_@@_final_open_bool }
1357         }
1358     }
1359 }
1360 \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 a *open* extremity.

```

1361 {

```

We do a step backwards because we will draw the dotted line upon the last cell in the matrix (we will use the “medium node” of this cell).

```

1362         \int_sub:Nn \l_@@_final_i_int { #3 }
1363         \int_sub:Nn \l_@@_final_j_int { #4 }
1364         \bool_set_true:N \l_@@_stop_loop_bool
1365     }

```

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`.

```

1366     {
1367         \@@_if_not_empty_cell:nnTF \l_@@_final_i_int \l_@@_final_j_int
1368         { \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.

```

1369         {
1370             \cs_set:cpn
1371             {
1372                 @@ _ dotted _
1373                 \int_use:N \l_@@_final_i_int -
1374                 \int_use:N \l_@@_final_j_int
1375             }
1376             { }
1377         }
1378     }
1379 }

```

We test whether the initial extremity of the dotted line is an implicit cell already dotted (by another dotted line). In this case, we can't draw the line because we have no Tikz node at the extremity of the arrow.

```

1380 \cs_if_free:cT
1381 {
1382     pgf@sh@ns@nm -
1383     \int_use:N \g_@@_env_int -
1384     \int_use:N \l_@@_final_i_int -
1385     \int_use:N \l_@@_final_j_int
1386 }
1387 {
1388     \bool_if:NF \l_@@_final_open_bool
1389     {
1390         \msg_error:nnx { nicematrix } { Impossible~line }
1391         { \int_use:N \l_@@_final_i_int }
1392         \bool_set_true:N \l_@@_impossible_line_bool
1393     }
1394 }

```

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

```

1395     \bool_set_false:N \l_@@_stop_loop_bool

```

```

1396 \bool_do_until:Nn \l_@@_stop_loop_bool
1397 {
1398   \int_sub:Nn \l_@@_initial_i_int { #3 }
1399   \int_sub:Nn \l_@@_initial_j_int { #4 }
1400   \bool_set_false:N \l_@@_initial_open_bool
1401   \int_compare:nNnTF \l_@@_initial_i_int < 1
1402   {
1403     \int_compare:nNnT { #3 } = 1
1404     { \bool_set_true:N \l_@@_initial_open_bool }
1405   }
1406   {
1407     \int_compare:nNnTF \l_@@_initial_j_int < 1
1408     {
1409       \int_compare:nNnT { #4 } = 1
1410       { \bool_set_true:N \l_@@_initial_open_bool }
1411     }
1412     {
1413       \int_compare:nNnT \l_@@_initial_j_int > \c@jCol
1414       {
1415         \int_compare:nNnT { #4 } = { -1 }
1416         { \bool_set_true:N \l_@@_initial_open_bool }
1417       }
1418     }
1419   }
1420   \bool_if:NTF \l_@@_initial_open_bool
1421   {
1422     \int_add:Nn \l_@@_initial_i_int { #3 }
1423     \int_add:Nn \l_@@_initial_j_int { #4 }
1424     \bool_set_true:N \l_@@_stop_loop_bool
1425   }
1426   {
1427     \@@_if_not_empty_cell:nnTF
1428     \l_@@_initial_i_int \l_@@_initial_j_int
1429     { \bool_set_true:N \l_@@_stop_loop_bool }
1430     {
1431       \cs_set:cpn
1432       {
1433         @@ _ dotted _
1434         \int_use:N \l_@@_initial_i_int -
1435         \int_use:N \l_@@_initial_j_int
1436       }
1437       { }
1438     }
1439   }
1440 }

```

We test whether the initial extremity of the dotted line is an implicit cell already dotted (by another dotted line). In this case, we can't draw the line because we have no Tikz node at the extremity of the arrow.

```

1441 \cs_if_free:cT
1442 {
1443   pgf@sh@ns@nm -
1444   \int_use:N \g_@@_env_int -
1445   \int_use:N \l_@@_initial_i_int -
1446   \int_use:N \l_@@_initial_j_int
1447 }
1448 {
1449   \bool_if:NF \l_@@_initial_open_bool
1450   {
1451     \msg_error:nnx { nicematrix } { Impossible~line }
1452     { \int_use:N \l_@@_initial_i_int }
1453     \bool_set_true:N \l_@@_impossible_line_bool
1454   }
1455 }

```

If we have at least one open extremity, we create the “medium nodes” in the matrix but we should change

that because, for an open extremity on the left or the right side of the array, we actually use the `col`-nodes. We remind that, when used once, the command `\@@_create_medium_nodes:` becomes no-op in the current TeX group.

```

1456     \bool_if:NT \l_@@_initial_open_bool \@@_create_medium_nodes:
1457     \bool_if:NT \l_@@_final_open_bool \@@_create_medium_nodes:
1458 }

```

The command `\@@_retrieve_coords:nn` retrieves the Tikz coordinates of the two extremities of the dotted line we will have to draw<sup>35</sup>. This command has four implicit arguments which are `\l_@@_initial_i_int`, `\l_@@_initial_j_int`, `\l_@@_final_i_int` and `\l_@@_final_j_int`.

The two arguments of the command `\@@_retrieve_coords:nn` are the suffix and the anchor that must be used for the two nodes.

The coordinates are stored in four variables:

- `\g_@@_x_initial_dim`
- `\g_@@_y_initial_dim`
- `\g_@@_x_final_dim`
- `\g_@@_y_final_dim`.

These variables are global for technical reasons: we have to do an affectation in an environment `{tikzpicture}`.

```

1459 \cs_new_protected:Npn \@@_retrieve_coords:nn #1 #2
1460 {
1461     \dim_gzero_new:N \g_@@_x_initial_dim
1462     \dim_gzero_new:N \g_@@_y_initial_dim
1463     \dim_gzero_new:N \g_@@_x_final_dim
1464     \dim_gzero_new:N \g_@@_y_final_dim
1465     \begin { tikzpicture } [ remember~picture ]
1466         \@@_extract_coords:w
1467         ( nm - \int_use:N \g_@@_env_int -
1468           \int_use:N \l_@@_initial_i_int -
1469           \int_use:N \l_@@_initial_j_int #1 )
1470         \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1471         \dim_gset:Nn \g_@@_y_initial_dim \pgf@y
1472         \@@_extract_coords:w
1473         ( nm - \int_use:N \g_@@_env_int -
1474           \int_use:N \l_@@_final_i_int -
1475           \int_use:N \l_@@_final_j_int #2 )
1476         \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1477         \dim_gset:Nn \g_@@_y_final_dim \pgf@y
1478     \end { tikzpicture }
1479 }
1480 \cs_generate_variant:Nn \@@_retrieve_coords:nn { x x }

```

For the horizontal lines with open extremities, we take into account the “col” nodes. The following command will recompute the  $x$ -value of the extremities in this case (erasing the value computed in `\@@_retrieve_coords:nn`).

```

1481 \cs_new_protected:Npn \@@_adjust_with_col_nodes:
1482 {
1483     \bool_if:NT \l_@@_initial_open_bool
1484     {
1485         \begin { tikzpicture } [ remember~picture ]
1486         \@@_extract_coords:w ( nm - \int_use:N \g_@@_env_int - col - 1 )
1487         \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1488         \end { tikzpicture }
1489     }
1490     \bool_if:NT \l_@@_final_open_bool
1491     {
1492         \begin { tikzpicture } [ remember~picture ]

```

<sup>35</sup>In fact, with diagonal lines, or vertical lines in columns of type L or R, an adjustment of one of the coordinates may be done.

```

1493 \@@_extract_coords:w
1494 ( nm - \int_use:N \g_@@_env_int - col - \int_eval:n { \c@jCol + 1 } )
1495 \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1496 \end { tikzpicture }
1497 }
1498 }

```

```

1499 \cs_new_protected:Npn \@@_draw_Ldots:nn #1 #2
1500 {
1501   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1502   {
1503     \bool_set_false:N \l_@@_impossible_line_bool
1504     \@@_find_extremities_of_line:nnnn { #1 } { #2 } 0 1
1505     \bool_if:NF \l_@@_impossible_line_bool \@@_actually_draw_Ldots:
1506   }
1507 }

```

The command `\@@_actually_draw_Ldots:` draws the `Ldots` line using the following variables:

- `\l_@@_initial_i_int`
- `\l_@@_initial_j_int`
- `\l_@@_initial_open_bool`
- `\l_@@_final_i_int`
- `\l_@@_final_j_int`
- `\l_@@_final_open_bool`.

We have a dedicated command because it is used also by `\Hdotsfor`.

```

1508 \cs_new_protected:Npn \@@_actually_draw_Ldots:
1509 {
1510   \@@_retrieve_coords:xx
1511   {
1512     \bool_if:NTF \l_@@_initial_open_bool
1513     { - medium.base~west }
1514     { .base~east }
1515   }
1516   {
1517     \bool_if:NTF \l_@@_final_open_bool
1518     { - medium.base~east }
1519     { .base~west }
1520   }
1521   \@@_adjust_with_col_nodes:
1522   \bool_if:NT \l_@@_initial_open_bool
1523   { \dim_gset_eq:NN \g_@@_y_initial_dim \g_@@_y_final_dim }
1524   \bool_if:NT \l_@@_final_open_bool
1525   { \dim_gset_eq:NN \g_@@_y_final_dim \g_@@_y_initial_dim }

```

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.

```

1526 \dim_gadd:Nn \g_@@_y_initial_dim { 0.53 pt }
1527 \dim_gadd:Nn \g_@@_y_final_dim { 0.53 pt }
1528 \@@_draw_tikz_line:
1529 }

```

```

1530 \cs_new_protected:Npn \@@_draw_Cdots:nn #1 #2
1531 {
1532   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1533   {
1534     \bool_set_false:N \l_@@_impossible_line_bool
1535     \@@_find_extremities_of_line:nnnn { #1 } { #2 } 0 1
1536     \bool_if:NF \l_@@_impossible_line_bool
1537     {

```

```

1538 \@@_retrieve_coords:xx
1539 {
1540   \bool_if:NTF \l_@@_initial_open_bool
1541   { - medium.mid~west }
1542   { .mid~east }
1543 }
1544 {
1545   \bool_if:NTF \l_@@_final_open_bool
1546   { - medium.mid~east }
1547   { .mid~west }
1548 }
1549 \@@_adjust_with_col_nodes:
1550 \bool_if:NT \l_@@_initial_open_bool
1551 { \dim_gset_eq:NN \g_@@_y_initial_dim \g_@@_y_final_dim }
1552 \bool_if:NT \l_@@_final_open_bool
1553 { \dim_gset_eq:NN \g_@@_y_final_dim \g_@@_y_initial_dim }
1554 \@@_draw_tikz_line:
1555 }
1556 }
1557 }

```

For the vertical dots, we have to distinguish different instances because we want really vertical lines. Be careful: it's not possible to insert the command `\@@_retrieve_coords:nn` in the arguments T and F of the `expl3` commands (why?).

```

1558 \cs_new_protected:Npn \@@_draw_Vdots:nn #1 #2
1559 {
1560   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1561   {
1562     \bool_set_false:N \l_@@_impossible_line_bool
1563     \@@_find_extremities_of_line:nnnn { #1 } { #2 } 1 0
1564     \bool_if:NF \l_@@_impossible_line_bool
1565     {
1566       \@@_retrieve_coords:xx
1567       {
1568         \bool_if:NTF \l_@@_initial_open_bool
1569         { - medium.north~west }
1570         { .south~west }
1571       }
1572       {
1573         \bool_if:NTF \l_@@_final_open_bool
1574         { - medium.south~west }
1575         { .north~west }
1576       }
1577     }
1578   }
1579 }

```

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

```

1577 \bool_set:Nn \l_tmpa_bool
1578 { \dim_compare_p:nNn \g_@@_x_initial_dim = \g_@@_x_final_dim }
1579 \@@_retrieve_coords:xx
1580 {
1581   \bool_if:NTF \l_@@_initial_open_bool
1582   { - medium.north }
1583   { .south }
1584 }
1585 {
1586   \bool_if:NTF \l_@@_final_open_bool
1587   { - medium.south }
1588   { .north }
1589 }

```

The boolean `\l_tmpb_bool` indicates whether the column is of type c (C of `{NiceArray}`) or may be considered as if.

```

1590 \bool_set:Nn \l_tmpb_bool
1591 { \dim_compare_p:nNn \g_@@_x_initial_dim = \g_@@_x_final_dim }

```

```

1592         \bool_if:NF \l_tmpb_bool
1593         {
1594             \dim_gset:Nn \g_@@_x_initial_dim
1595             {
1596                 \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
1597                 \g_@@_x_initial_dim \g_@@_x_final_dim
1598             }
1599             \dim_gset_eq:NN \g_@@_x_final_dim \g_@@_x_initial_dim
1600         }
1601         \@@_draw_tikz_line:
1602     }
1603 }
1604 }

```

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.

```

1605 \cs_new_protected:Npn \@@_draw_Ddots:nn #1 #2
1606 {
1607     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1608     {
1609         \bool_set_false:N \l_@@_impossible_line_bool
1610         \@@_find_extremities_of_line:nnnn { #1 } { #2 } 1 1
1611         \bool_if:NF \l_@@_impossible_line_bool
1612         {
1613             \@@_retrieve_coords:xx
1614             {
1615                 \bool_if:NTF \l_@@_initial_open_bool
1616                 { - medium.north-west }
1617                 { .south-east }
1618             }
1619             {
1620                 \bool_if:NTF \l_@@_final_open_bool
1621                 { - medium.south-east }
1622                 { .north-west }
1623             }
1624         }
1625     }
1626 }

```

We have retrieved the coordinates in the usual way (they are stored in `\g_@@_x_initial_dim`, etc.). If the parallelization of the diagonals is set, we will have (maybe) to adjust the fourth coordinate.

```

1624         \bool_if:NT \l_@@_parallelize_diags_bool
1625         {
1626             \int_incr:N \l_@@_ddots_int

```

We test if the diagonal line is the first one (the counter `\l_@@_ddots_int` is created for this usage).

```

1627         \int_compare:nNnTF \l_@@_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.

```

1628         {
1629             \dim_set:Nn \l_@@_delta_x_one_dim
1630             { \g_@@_x_final_dim - \g_@@_x_initial_dim }
1631             \dim_set:Nn \l_@@_delta_y_one_dim
1632             { \g_@@_y_final_dim - \g_@@_y_initial_dim }
1633         }

```

If the diagonal line is not the first one, we have to adjust the second extremity of the line by modifying the coordinate `\g_@@_y_initial_dim`.

```

1634         {
1635             \dim_gset:Nn \g_@@_y_final_dim
1636             {
1637                 \g_@@_y_initial_dim +
1638                 ( \g_@@_x_final_dim - \g_@@_x_initial_dim ) *
1639                 \dim_ratio:nn \l_@@_delta_y_one_dim \l_@@_delta_x_one_dim
1640             }

```

```

1641         }
1642     }

```

Now, we can draw the dotted line (after a possible change of `\g_@@_y_initial_dim`).

```

1643         \@@_draw_tikz_line:
1644     }
1645 }
1646 }

```

We draw the `\Iddots` diagonals in the same way.

```

1647 \cs_new_protected:Npn \@@_draw_Iddots:nn #1 #2
1648 {
1649     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1650     {
1651         \bool_set_false:N \l_@@_impossible_line_bool
1652         \@@_find_extremities_of_line:nnnn { #1 } { #2 } 1 { -1 }
1653         \bool_if:NF \l_@@_impossible_line_bool
1654         {
1655             \@@_retrieve_coords:xx
1656             {
1657                 \bool_if:NTF \l_@@_initial_open_bool
1658                 { - medium.north~east }
1659                 { .south~west }
1660             }
1661             {
1662                 \bool_if:NTF \l_@@_final_open_bool
1663                 { - medium.south~west }
1664                 { .north~east }
1665             }
1666             \bool_if:NT \l_@@_parallelize_diags_bool
1667             {
1668                 \int_incr:N \l_@@_iddots_int
1669                 \int_compare:nNnTF \l_@@_iddots_int = 1
1670                 {
1671                     \dim_set:Nn \l_@@_delta_x_two_dim
1672                     { \g_@@_x_final_dim - \g_@@_x_initial_dim }
1673                     \dim_set:Nn \l_@@_delta_y_two_dim
1674                     { \g_@@_y_final_dim - \g_@@_y_initial_dim }
1675                 }
1676                 {
1677                     \dim_gset:Nn \g_@@_y_final_dim
1678                     {
1679                         \g_@@_y_initial_dim +
1680                         ( \g_@@_x_final_dim - \g_@@_x_initial_dim ) *
1681                         \dim_ratio:nn \l_@@_delta_y_two_dim \l_@@_delta_x_two_dim
1682                     }
1683                 }
1684             }
1685             \@@_draw_tikz_line:
1686         }
1687     }
1688 }

```

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).

```

1689 \NewExpandableDocumentCommand \NiceMatrixLastEnv { }
1690 { \int_use:N \g_@@_env_int }

```



## The actual instructions for drawing the dotted line with Tikz

The command `\@@_draw_tikz_line`: draws the line using four implicit arguments:

`\g_@@_x_initial_dim`, `\g_@@_y_initial_dim`, `\g_@@_x_final_dim` and `\g_@@_y_final_dim`. These variables are global for technical reasons: their first affectation was in an instruction `\tikz`.

```
1691 \cs_new_protected:Npn \@@_draw_tikz_line:
1692 {
```

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.

```
1693   \dim_zero_new:N \l_@@_l_dim
1694   \dim_set:Nn \l_@@_l_dim
1695   {
1696     \fp_to_dim:n
1697     {
1698       sqrt
1699       (
1700         ( \dim_use:N \g_@@_x_final_dim
1701           - \dim_use:N \g_@@_x_initial_dim
1702         ) ^ 2
1703         +
1704         ( \dim_use:N \g_@@_y_final_dim
1705           - \dim_use:N \g_@@_y_initial_dim
1706         ) ^ 2
1707       )
1708     }
1709   }
```

We draw only if the length is not equal to zero (in fact, in the first compilation, the length may be equal to zero).

```
1710   \dim_compare:nNnF \l_@@_l_dim = \c_zero_dim
```

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

```
1711   {
1712     \bool_if:NTF \l_@@_initial_open_bool
1713     {
1714       \bool_if:NTF \l_@@_final_open_bool
1715       {
1716         \int_set:Nn \l_tmpa_int
1717         { \dim_ratio:nn \l_@@_l_dim \l_@@_inter_dots_dim }
1718       }
1719       {
1720         \int_set:Nn \l_tmpa_int
1721         {
1722           \dim_ratio:nn
1723           { \l_@@_l_dim - \l_@@_dotted_lines_margin_dim }
1724           \l_@@_inter_dots_dim
1725         }
1726       }
1727     }
1728     {
1729       \bool_if:NTF \l_@@_final_open_bool
1730       {
1731         \int_set:Nn \l_tmpa_int
1732         {
1733           \dim_ratio:nn
1734           { \l_@@_l_dim - \l_@@_dotted_lines_margin_dim }
1735           \l_@@_inter_dots_dim
1736         }
1737       }
1738       {
1739         \int_set:Nn \l_tmpa_int
1740         {
1741           \dim_ratio:nn
1742           { \l_@@_l_dim - ( 2 \l_@@_dotted_lines_margin_dim ) }
```

```

1743         \l_@@_inter_dots_dim
1744     }
1745 }
1746 }

```

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

```

1747 \dim_set:Nn \l_tmpa_dim
1748 {
1749     ( \g_@@_x_final_dim - \g_@@_x_initial_dim ) *
1750     \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
1751 }
1752 \dim_set:Nn \l_tmpb_dim
1753 {
1754     ( \g_@@_y_final_dim - \g_@@_y_initial_dim ) *
1755     \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
1756 }

```

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`.

```

1757 \int_set:Nn \l_tmpb_int
1758 {
1759     \bool_if:NTF \l_@@_initial_open_bool
1760     { \bool_if:NTF \l_@@_final_open_bool 1 0 }
1761     { \bool_if:NTF \l_@@_final_open_bool 2 1 }
1762 }

```

In the loop over the dots, the dimensions `\g_@@_x_initial_dim` and `\g_@@_y_initial_dim` will be used for the coordinates of the dots. But, before the loop, we must move until the first dot.

```

1763 \dim_gadd:Nn \g_@@_x_initial_dim
1764 {
1765     ( \g_@@_x_final_dim - \g_@@_x_initial_dim ) *
1766     \dim_ratio:nn
1767     { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
1768     { 2 \l_@@_l_dim }
1769     * \l_tmpb_int
1770 }
1771 \dim_gadd:Nn \g_@@_y_initial_dim
1772 {
1773     ( \g_@@_y_final_dim - \g_@@_y_initial_dim ) *
1774     \dim_ratio:nn
1775     { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
1776     { 2 \l_@@_l_dim }
1777     * \l_tmpb_int
1778 }
1779 \begin { tikzpicture } [ overlay ]
1780     \int_step_inline:nnn 0 \l_tmpa_int
1781     {
1782         \pgfpathcircle
1783         { \pgfpoint { \g_@@_x_initial_dim } { \g_@@_y_initial_dim } }
1784         { \l_@@_radius_dim }
1785         \pgfusepath { fill }
1786         \dim_gadd:Nn \g_@@_x_initial_dim \l_tmpa_dim
1787         \dim_gadd:Nn \g_@@_y_initial_dim \l_tmpb_dim
1788     }
1789 \end { tikzpicture }
1790 }
1791 }

```

## User commands available in the new environments

We give new names for the commands `\ldots`, `\cdots`, `\vdots` and `\ddots` because these commands will be redefined (if the option `renew-dots` is used).

```

1792 \cs_set_eq:NN \@@_ldots \ldots
1793 \cs_set_eq:NN \@@_cdots \cdots
1794 \cs_set_eq:NN \@@_vdots \vdots
1795 \cs_set_eq:NN \@@_ddots \ddots
1796 \cs_set_eq:NN \@@_iddots \iddots

```

The command `\@@_add_to_empty_cells:` declares the current cell as empty. For efficiency, this is done by creating a special control sequence.

```

1797 \cs_new_protected:Npn \@@_add_to_empty_cells:
1798 {
1799   \cs_gset:cpx
1800     { @@ _ empty _ \int_use:N \c@iRow - \int_use:N \c@jCol }
1801     { \int_use:N \g_@@_env_int }
1802 }

```

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 they are still available.

```

1803 \NewDocumentCommand \@@_Ldots { s }
1804 {
1805   \bool_if:nF { #1 } { \@@_instruction_of_type:n { Ldots } }
1806   \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_ldots }
1807   \@@_add_to_empty_cells:
1808 }

```

```

1809 \NewDocumentCommand \@@_Cdots { s }
1810 {
1811   \bool_if:nF { #1 } { \@@_instruction_of_type:n { Cdots } }
1812   \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_cdots }
1813   \@@_add_to_empty_cells:
1814 }

```

```

1815 \NewDocumentCommand \@@_Vdots { s }
1816 {
1817   \bool_if:nF { #1 } { \@@_instruction_of_type:n { Vdots } }
1818   \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_vdots }
1819   \@@_add_to_empty_cells:
1820 }

```

```

1821 \NewDocumentCommand \@@_Ddots { s }
1822 {
1823   \bool_if:nF { #1 } { \@@_instruction_of_type:n { Ddots } }
1824   \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_ddots }
1825   \@@_add_to_empty_cells:
1826 }

```

```

1827 \NewDocumentCommand \@@_Iddots { s }
1828 {
1829   \bool_if:nF { #1 } { \@@_instruction_of_type:n { Iddots } }
1830   \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_iddots }
1831   \@@_add_to_empty_cells:
1832 }

```

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

```

1833 \cs_new_protected:Npn \@@_Hspace:
1834 {
1835   \@@_add_to_empty_cells:
1836   \hspace
1837 }

```

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

```

1838 \cs_set_eq:NN \@@_old_multicolumn \multicolumn
1839 \cs_new:Npn \@@_multicolumn:nnn #1 #2 #3
1840 {
1841   \@@_old_multicolumn { #1 } { #2 } { #3 }
1842   \int_compare:nNnT #1 > 1
1843   {
1844     \seq_gput_left:Nx \g_@@_multicolumn_cells_seq
1845     { \int_eval:n \c@iRow - \int_use:N \c@jCol }
1846     \seq_gput_left:Nn \g_@@_multicolumn_sizes_seq { #1 }
1847   }
1848   \int_gadd:Nn \c@jCol { #1 - 1 }
1849 }

```

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 for all the cells of the array, even the implicit cells of the `\Hdotsfor`.

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

```

1850 \cs_new:Npn \@@_Hdotsfor:
1851 {
1852   \multicolumn { 1 } { C } { }
1853   \@@_Hdotsfor_i
1854 }

```

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:`).

```

1855 \bool_if:NTF \c_@@_draft_bool
1856 {
1857   \NewDocumentCommand \@@_Hdotsfor_i { 0 { } m }
1858   { \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { C } { } } }
1859 }
1860 {
1861   \NewDocumentCommand \@@_Hdotsfor_i { 0 { } m }
1862   {
1863     \tl_gput_right:Nx \g_@@_Hdotsfor_lines_tl
1864     {
1865       \@@_Hdotsfor:nnn
1866       { \int_use:N \c@iRow }
1867       { \int_use:N \c@jCol }
1868       { #2 }
1869     }
1870     \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { C } { } }
1871   }
1872 }

```

```

1873 \cs_new_protected:Npn \@@_Hdotsfor:nnn #1 #2 #3
1874 {
1875   \bool_set_false:N \l_@@_initial_open_bool
1876   \bool_set_false:N \l_@@_final_open_bool

```

For the row, it's easy.

```

1877 \int_set:Nn \l_@@_initial_i_int { #1 }
1878 \int_set:Nn \l_@@_final_i_int { #1 }

```

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

```

1879 \int_compare:nNnTF #2 = 1
1880 {
1881   \int_set:Nn \l_@@_initial_j_int 1
1882   \bool_set_true:N \l_@@_initial_open_bool
1883 }
1884 {
1885   \int_set:Nn \l_tmpa_int { #2 - 1 }
1886   \@@_if_not_empty_cell:nnTF \l_@@_initial_i_int \l_tmpa_int
1887   { \int_set:Nn \l_@@_initial_j_int { #2 - 1 } }
1888   {
1889     \int_set:Nn \l_@@_initial_j_int { #2 }
1890     \bool_set_true:N \l_@@_initial_open_bool
1891   }
1892 }
1893 \int_compare:nNnTF { #2 + #3 - 1 } = \c@jCol
1894 {
1895   \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
1896   \bool_set_true:N \l_@@_final_open_bool
1897 }
1898 {
1899   \int_set:Nn \l_tmpa_int { #2 + #3 }
1900   \@@_if_not_empty_cell:nnTF \l_@@_final_i_int \l_tmpa_int
1901   { \int_set:Nn \l_@@_final_j_int { #2 + #3 } }
1902   {
1903     \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
1904     \bool_set_true:N \l_@@_final_open_bool
1905   }
1906 }
1907 \bool_if:nT { \l_@@_initial_open_bool || \l_@@_final_open_bool }
1908 \@@_create_medium_nodes:

```

The command `\@@_adjust_with_col_nodes:` is used in the command `\@@_actually_draw_Ldots:` in order to recompute the  $x$ -value of the initial point and the  $x$ -value of the final point with the exact position of the left side and the right side of the array (these informations are in the `col`-node. However, we don't want this operation with the dotted lines drawn by `\Hdotsfor`. That's why we deactivate locally this command.

```

1909 \group_begin:
1910   \cs_set:Npn \@@_adjust_with_col_nodes: { }
1911   \@@_actually_draw_Ldots:
1912 \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).

```

1913 \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
1914 { \cs_set:cpn { @@ _ dotted _ #1 - ##1 } { } }
1915 }

```

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`.

```

1916 \cs_new_protected:Npn \@@_rotate: { \group_insert_after:N \@@_rotate_i: }
1917 \cs_new_protected:Npn \@@_rotate_i: { \group_insert_after:N \@@_rotate_ii: }
1918 \cs_new_protected:Npn \@@_rotate_ii: { \group_insert_after:N \@@_rotate_iii: }
1919 \cs_new_protected:Npn \@@_rotate_iii: { \box_rotate:Nn \l_@@_cell_box { 90 } }

```

## 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>36</sup>

```
1920 \cs_new:Npn \@@_double_int_eval:n #1-#2 \q_stop
1921   { \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).

```
1922 \cs_new_protected:Npn \@@_line:nn #1 #2
1923   {
1924     \use:x
1925     {
1926       \@@_line_i:nn
1927       { \@@_double_int_eval:n #1 \q_stop }
1928       { \@@_double_int_eval:n #2 \q_stop }
1929     }
1930   }

1931 \cs_new_protected:Npn \@@_line_i:nn #1 #2
1932   {
1933     \bool_if:NF \c_@@_draft_bool
1934     {
1935       \dim_zero_new:N \g_@@_x_initial_dim
1936       \dim_zero_new:N \g_@@_y_initial_dim
1937       \dim_zero_new:N \g_@@_x_final_dim
1938       \dim_zero_new:N \g_@@_y_final_dim
1939       \bool_set_false:N \l_@@_initial_open_bool
1940       \bool_set_false:N \l_@@_final_open_bool
1941       \bool_if:nTF
1942       {
1943         \cs_if_exist_p:c { pgf@sh@ns@nm - \int_use:N \g_@@_env_int - #1 }
1944         &&
1945         \cs_if_exist_p:c { pgf@sh@ns@nm - \int_use:N \g_@@_env_int - #2 }
1946       }
1947       {
1948         \begin { tikzpicture }
1949           \path~(#1)~---~(#2)~node[at~start]~(i)~{ }~node[at~end]~(f)~{ } ;
1950           \@@_extract_coords:w ( i )
1951           \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1952           \dim_gset:Nn \g_@@_y_initial_dim \pgf@y
1953           \@@_extract_coords:w ( f )
1954           \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1955           \dim_gset:Nn \g_@@_y_final_dim \pgf@y
1956         \end { tikzpicture }
1957         \@@_draw_tikz_line:
1958       }
1959       {
1960         \@@_error:nnn { unknown~cell~for~line~in~code~after }
1961         { #1 } { #2 }
1962       }
1963     }
1964   }
```

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).

---

<sup>36</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`.

## The vertical rules

We give to the user the possibility to define new types of columns (with `\newcolumntype` 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`.

```
1965 \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`.

```
1966 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix:n #1
1967 {
1968   \int_compare:nNnTF \l_@@_first_col_int = 0
1969     { \@@_OnlyMainNiceMatrix_i:n { #1 } }
1970     {
1971       \int_compare:nNnTF \c@jCol = 0
1972         {
1973           \int_compare:nNnF \c@iRow = { -1 }
1974             { \int_compare:nNnF \c@iRow = { \l_@@_last_row_int - 1 } { #1 } }
1975         }
1976       { \@@_OnlyMainNiceMatrix_i:n { #1 } }
1977     }
1978 }
```

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.

```
1979 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix_i:n #1
1980 {
1981   \int_compare:nNnF \c@iRow = 0
1982     { \int_compare:nNnF \c@iRow = \l_@@_last_row_int { #1 } }
1983 }
```

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 `\newcolumntype`:

```
\newcolumntype { | } { ! { \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).

```
1984 \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`.

```
1985 \cs_set_eq:NN \@@_vline_i: \vline
```

## 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`.

```
1986 \cs_new:Npn \@@_hdottedline:
1987 {
1988   \bool_if:NF \c_@@_draft_bool
1989   {
1990     \noalign{ \skip_vertical:n { 2 \l_@@_radius_dim } }
1991     \@@_hdottedline_i:
1992   }
1993 }
```

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

```
1994 \cs_new_protected:Npn \@@_hdottedline_i:
1995 {
```

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).

```
1996   \tl_gput_right:Nx \g_@@_code_after_tl
1997   { \@@_hdottedline:n { \int_use:N \c@iRow } }
1998 }
```

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.

```
1999 \cs_new_protected:Npn \@@_hdottedline:n #1
2000 {
2001   \dim_zero_new:N \g_@@_x_initial_dim
2002   \dim_zero_new:N \g_@@_y_initial_dim
2003   \dim_zero_new:N \g_@@_x_final_dim
2004   \dim_zero_new:N \g_@@_y_final_dim
2005   \bool_set_true:N \l_@@_initial_open_bool
2006   \bool_set_true:N \l_@@_final_open_bool
2007   \begin { tikzpicture } [ remember=picture ]
2008     \@@_extract_coords:w ( 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).

```
2009     \dim_gset:Nn \g_@@_y_initial_dim { \pgf@y - \l_@@_radius_dim }
2010     \dim_gset:Nn \g_@@_y_final_dim { \pgf@y - \l_@@_radius_dim }
2011     \@@_extract_coords:w ( col - 1 )
2012     \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
2013     \@@_extract_coords:w ( col - \int_eval:n { \c@jCol + 1 } )
2014     \dim_gset:Nn \g_@@_x_final_dim \pgf@x
2015   \end { tikzpicture }
2016   \@@_draw_tikz_line:
2017 }
```

### Vertical dotted lines

```
2018 \cs_new_protected:Npn \@@_vdottedline:n #1
2019 {
```

We should allow the letter “:” in the first position of the preamble but that would need a special programming.

```
2020   \int_compare:nNnTF #1 = 0
2021   { \@@_error:n { Use~of~:~in~first~position } }
2022   { \bool_if:NF \c_@@_draft_bool { \@@_vdottedline_i:n { #1 } } }
2023 }
```



```

2024 \cs_new_protected:Npn \@@_vdottedline_i:n #1
2025 {
2026   \dim_zero_new:N \g_@@_x_initial_dim
2027   \dim_zero_new:N \g_@@_y_initial_dim
2028   \dim_zero_new:N \g_@@_x_final_dim
2029   \dim_zero_new:N \g_@@_y_final_dim
2030   \bool_set_true:N \l_@@_initial_open_bool
2031   \bool_set_true:N \l_@@_final_open_bool
2032   \begin { tikzpicture } [ remember~picture ]
2033     \@@_extract_coords:w ( 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).

```

2034     \dim_gset:Nn \g_@@_x_initial_dim { \pgf@x - \l_@@_radius_dim }
2035     \dim_gset:Nn \g_@@_x_final_dim { \pgf@x - \l_@@_radius_dim }
2036     \@@_extract_coords:w ( 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.

```

2037     \dim_gset:Nn \g_@@_y_initial_dim { \pgf@y - ( \l_@@_inter_dots_dim / 2 ) }
2038     \@@_extract_coords:w ( row - \int_eval:n { \c@iRow + 1 } )
2039     \dim_gset:Nn \g_@@_y_final_dim { \pgf@y + ( \l_@@_inter_dots_dim / 2 ) }
2040   \end { tikzpicture }
2041   \@@_draw_tikz_line:
2042 }

```

## 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.

```

2043 \bool_new:N \l_@@_block_auto_columns_width_bool

```

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

```

2044 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
2045 {
2046   auto-columns-width .code:n =
2047   {
2048     \bool_set_true:N \l_@@_block_auto_columns_width_bool
2049     \dim_gzero_new:N \g_@@_max_cell_width_dim
2050     \bool_set_true:N \l_@@_auto_columns_width_bool
2051   }
2052 }

2053 \NewDocumentEnvironment { NiceMatrixBlock } { ! 0 { } }
2054 {
2055   \int_gincr:N \g_@@_NiceMatrixBlock_int
2056   \dim_zero:N \l_@@_columns_width_dim
2057   \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
2058   \bool_if:NT \l_@@_block_auto_columns_width_bool
2059   {
2060     \cs_if_exist:cT { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
2061     {
2062       \dim_set:Nx \l_@@_columns_width_dim
2063       { \use:c { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int } }
2064     }
2065   }
2066 }

```

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`).

```

2067 {
2068   \bool_if:NT \l_@@_block_auto_columns_width_bool
2069   {
2070     \iow_now:Nn \@mainaux \ExplSyntaxOn
2071     \iow_now:Nx \@mainaux
2072     {
2073       \cs_gset:cpn
2074       { @@ _ max _ cell _ width _ \int_use:N \g_@@_NiceMatrixBlock_int }
2075       { \dim_use:N \g_@@_max_cell_width_dim }
2076     }
2077     \iow_now:Nn \@mainaux \ExplSyntaxOff
2078   }
2079 }

```

## The extra nodes

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

```

2080 \cs_generate_variant:Nn \dim_min:nn { v n }
2081 \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:`. They must *not* be used in the `code-after` because the `code-after` is executed in a scope of `prefix name` of Tikz.

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 `{tikzpicture}`.

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`.

```

2082 \cs_new_protected:Npn \@@_computations_for_medium_nodes:
2083 {
2084   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
2085   {
2086     \dim_zero_new:c { l_@@_row\_@@_i: _min_dim }
2087     \dim_set_eq:cN { l_@@_row\_@@_i: _min_dim } \c_max_dim
2088     \dim_zero_new:c { l_@@_row\_@@_i: _max_dim }
2089     \dim_set:cn { l_@@_row\_@@_i: _max_dim } { - \c_max_dim }
2090   }
2091   \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
2092   {
2093     \dim_zero_new:c { l_@@_column\_@@_j: _min_dim }
2094     \dim_set_eq:cN { l_@@_column\_@@_j: _min_dim } \c_max_dim
2095     \dim_zero_new:c { l_@@_column\_@@_j: _max_dim }
2096     \dim_set:cn { l_@@_column\_@@_j: _max_dim } { - \c_max_dim }
2097   }

```

We begin the two nested loops over the rows and the columns of the array.

```

2098   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
2099   {
2100     \int_step_variable:nnNn
2101     \l_@@_first_col_int \g_@@_col_total_int \@@_j:

```

Maybe the cell  $(i-j)$  is an implicit cell (that is to say a cell after implicit ampersands &). In this case, of course, we don't update the dimensions we want to compute.

```

2102     {
2103         \cs_if_exist:cT
2104         { pgf@sh@ns@nm - \int_use:N \g_@@_env_int - \@@_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`.

```

2105     {
2106         \@@_extract_coords:w
2107         ( nm - \int_use:N \g_@@_env_int
2108           - \@@_i: - \@@_j: .south-west )
2109         \dim_set:cn { l_@@_row _ \@@_i: _min_dim }
2110         { \dim_min:vn { l_@@_row _ \@@_i: _min_dim } \pgf@y }
2111         \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
2112         {
2113             \dim_set:cn { l_@@_column _ \@@_j: _min_dim }
2114             { \dim_min:vn { l_@@_column _ \@@_j: _min_dim } \pgf@x }
2115         }

```

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`.

```

2116         \@@_extract_coords:w
2117         ( nm - \int_use:N \g_@@_env_int - \@@_i: - \@@_j: .north-east )
2118         \dim_set:cn { l_@@_row _ \@@_i: _max_dim }
2119         { \dim_max:vn { l_@@_row _ \@@_i: _max_dim } \pgf@y }
2120         \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
2121         {
2122             \dim_set:cn { l_@@_column _ \@@_j: _max_dim }
2123             { \dim_max:vn { l_@@_column _ \@@_j: _max_dim } \pgf@x }
2124         }
2125     }
2126 }
2127 }
2128 }

```

Here is the command `\@@_create_medium_nodes:`. When this command is used, the “medium nodes” are created. These nodes won't be constructed twice because when used once, this command becomes no-op.

```

2129 \cs_new_protected:Npn \@@_create_medium_nodes:
2130 {
2131     \begin { tikzpicture } [ remember-picture , overlay ]
2132     \@@_computations_for_medium_nodes:
2133     \tikzset { name~suffix = -medium }

```

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” (after changing the value of `name~suffix`).

```

2134     \@@_create_nodes:
2135     \end { tikzpicture }
2136     \cs_set_protected:Npn \@@_create_medium_nodes: { }
2137     \cs_set_protected:Npn \@@_create_medium_and_large_nodes:
2138     { \@@_create_large_nodes: }
2139 }

```

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:`.

```

2140 \cs_new_protected:Npn \@@_create_large_nodes:
2141 {
2142     \begin { tikzpicture } [ remember-picture , overlay ]
2143     \@@_computations_for_medium_nodes:
2144     \@@_computations_for_large_nodes:
2145     \tikzset { name~suffix = -large }

```

```

2146 \@@_create_nodes:
2147 \end { tikzpicture }
2148 \cs_set_protected:Npn \@@_create_large_nodes: { }
2149 \cs_set_protected:Npn \@@_create_medium_and_large_nodes:
2150 { \@@_create_medium_nodes: }
2151 }
2152 \cs_new_protected:Npn \@@_create_medium_and_large_nodes:
2153 {
2154 \begin { tikzpicture } [ remember-picture , overlay ]
2155 \@@_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” (after changing the value of `name-suffix`).

```

2156 \tikzset { name-suffix = -medium }
2157 \@@_create_nodes:
2158 \@@_computations_for_large_nodes:
2159 \tikzset { name-suffix = -large }
2160 \@@_create_nodes:
2161 \end { tikzpicture }
2162 \cs_set_protected:Npn \@@_create_medium_and_large_nodes: { }
2163 \cs_set_protected:Npn \@@_create_medium_nodes: { }
2164 \cs_set_protected:Npn \@@_create_large_nodes: { }
2165 }

```

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.

```

2166 \cs_new_protected:Npn \@@_computations_for_large_nodes:
2167 {
2168 \int_set:Nn \l_@@_first_row_int 1
2169 \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`.

```

2170 \int_step_variable:nNn { \c@iRow - 1 } \@@_i:
2171 {
2172 \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim }
2173 {
2174 (
2175 \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } +
2176 \dim_use:c { l_@@_row _ \int_eval:n { \@@_i: + 1 } _ max _ dim }
2177 )
2178 / 2
2179 }
2180 \dim_set_eq:cc { l_@@_row _ \int_eval:n { \@@_i: + 1 } _ max _ dim }
2181 { l_@@_row _ \@@_i: _ min _ dim }
2182 }
2183 \int_step_variable:nNn { \c@jCol - 1 } \@@_j:
2184 {
2185 \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim }
2186 {
2187 (
2188 \dim_use:c
2189 { l_@@_column _ \@@_j: _ max _ dim } +
2190 \dim_use:c
2191 { l_@@_column _ \int_eval:n { \@@_j: + 1 } _ min _ dim }
2192 )
2193 / 2
2194 }
2195 \dim_set_eq:cc { l_@@_column _ \int_eval:n { \@@_j: + 1 } _ min _ dim }
2196 { l_@@_column _ \@@_j: _ max _ dim }
2197 }
2198 \dim_sub:cn
2199 { l_@@_column _ 1 _ min _ dim }

```

```

2200     \l_@@_left_margin_dim
2201     \dim_add:cn
2202     { l_@@_column _ \int_use:N \c@jCol _ max _ dim }
2203     \l_@@_right_margin_dim
2204 }

```

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.

```

2205 \cs_new_protected:Npn \@@_create_nodes:
2206 {
2207     \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
2208     {
2209         \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:

```

We create two punctual nodes for the extremities of a diagonal of the rectangular node we want to create. These nodes (`@@~south~west`) and (`@@~north~east`) are not available for the user of `nicematrix`. That’s why their names are independent of the row and the column. In the two nested loops, they will be overwritten until the last cell.

```

2210     {
2211         \coordinate ( @@~south~west )
2212             at ( \dim_use:c { l_@@_column_ \@@_j: _min_dim } ,
2213                 \dim_use:c { l_@@_row_ \@@_i: _min_dim } ) ;
2214         \coordinate ( @@~north~east )
2215             at ( \dim_use:c { l_@@_column_ \@@_j: _max_dim } ,
2216                 \dim_use:c { l_@@_row_ \@@_i: _max_dim } ) ;

```

We can eventually draw the rectangular node for the cell (`\@@_i-\@@_j`). This node is created with the Tikz library `fit`. Don’t forget that the Tikz option `name suffix` has been set to `-medium` or `-large`.

```

2217         \node
2218         [
2219             node~contents = { } ,
2220             fit = ( @@~south~west ) ( @@~north~east ) ,
2221             inner~sep = \c_zero_dim ,
2222             name = nm - \int_use:N \g_@@_env_int - \@@_i: - \@@_j: ,
2223             alias =
2224                 \str_if_empty:NF \l_@@_name_str
2225                 { \l_@@_name_str - \@@_i: - \@@_j: }
2226         ]
2227         ;
2228     }
2229 }

```

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  $n$ .

```

2230     \seq_mapthread_function:NNN
2231     \g_@@_multicolumn_cells_seq
2232     \g_@@_multicolumn_sizes_seq
2233     \@@_node_for_multicolumn:nn
2234 }

```

```

2235 \cs_new_protected:Npn \@@_extract_coords_values: #1 - #2 \q_stop
2236 {
2237     \cs_set:Npn \@@_i: { #1 }
2238     \cs_set:Npn \@@_j: { #2 }
2239 }

```

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”).

```

2240 \cs_new_protected:Npn \@@_node_for_multicolumn:nn #1 #2
2241 {

```

```

2242 \@@_extract_coords_values: #1 \q_stop
2243 \coordinate ( @@~south~west ) at
2244 (
2245     \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } ,
2246     \dim_use:c { l_@@_row _ \@@_i: _ min _ dim }
2247 ) ;
2248 \coordinate ( @@~north~east ) at
2249 (
2250     \dim_use:c { l_@@_column _ \int_eval:n { \@@_j: + #2 - 1 } _ max _ dim } ,
2251     \dim_use:c { l_@@_row _ \@@_i: _ max _ dim }
2252 ) ;
2253 \node
2254 [
2255     node~contents = { } ,
2256     fit = ( @@~south~west ) ( @@~north~east ) ,
2257     inner~sep = \c_zero_dim ,
2258     name = nm - \int_use:N \g_@@_env_int - \@@_i: - \@@_j: ,
2259     alias =
2260         \str_if_empty:NF \l_@@_name_str
2261         { \l_@@_name_str - \@@_i: - \@@_j: }
2262 ]
2263 ;
2264 }

```

## 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)

```

2265 \NewDocumentCommand \@@_Block: { 0 { } m D < > { } m }
2266 { \@@_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.

```

2267 \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.

```

2268 \cs_new_protected:Npn \@@_Block_ii:nnnnn #1 #2 #3 #4 #5
2269 {

```

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).

```

2270 \tl_gput_left:Nx \g_@@_code_after_tl
2271 {
2272     \@@_Block_iii:nnnnnn
2273     { \int_use:N \c_iRow }
2274     { \int_use:N \c_jCol }
2275     { \int_eval:n { \c_iRow + #1 - 1 } }
2276     { \int_eval:n { \c_jCol + #2 - 1 } }
2277     { #3 }
2278     \exp_not:n { { #4 $ #5 $ } }
2279 }

```

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.

```

2280 \cs_set_eq:NN \Block \@@_Block_error:nn
2281 }

```

```

2282 \cs_new:Npn \@@_Block_error:nn #1 #2
2283 {
2284   \@@_error:n { Second~Block }
2285   \cs_set_eq:NN \Block \use:nn
2286 }

2287 \keys_define:nn { NiceMatrix / Block }
2288 {
2289   tikz .tl_set:N = \l_@@_tikz_tl ,
2290   tikz .value_required:n = true ,
2291   white .bool_set:N = \l_@@_white_bool ,
2292   white .default:n = true ,
2293   white .value_forbidden:n = true ,
2294 }

```

The following command `\@@_Block_iii:nnnnnn` will be used in the code-after.

```

2295 \cs_new_protected:Npn \@@_Block_iii:nnnnnn #1 #2 #3 #4 #5 #6
2296 {

```

The group is for the keys.

```

2297   \group_begin:
2298   \keys_set:nn { NiceMatrix / Block } { #5 }
2299   \bool_if:nTF
2300   {
2301     \int_compare_p:nNn { #3 } > \c@iRow
2302     || \int_compare_p:nNn { #4 } > \c@jCol
2303   }
2304   { \msg_error:nnnn { nicematrix } { Block~too~large } { #1 } { #2 } }
2305   {

```

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.

```

2306   \hbox_set:Nn \l_@@_cell_box { #6 }

```

The construction of the node corresponding to the merged cells. First, we construct `\coordinate` corresponding to the upper-left and the lower-right corners of the cell. They are called `one` and `two`.

```

2307   \begin { tikzpicture }
2308     \int_set:Nn \l_tmpa_int { #1 }
2309     \int_decr:N \l_tmpa_int
2310     \int_set:Nn \l_tmpb_int { #2 }
2311     \int_decr:N \l_tmpb_int
2312     \@@_extract_coords:w ( row - \int_eval:n { \l_tmpa_int + 1 } )
2313     \dim_set_eq:NN \l_tmpa_dim \pgf@y
2314     \@@_extract_coords:w ( col - \int_eval:n { \l_tmpb_int + 1 } )
2315     \dim_set_eq:NN \l_tmpb_dim \pgf@x
2316     \coordinate ( one ) at ( \l_tmpb_dim , \l_tmpa_dim ) ;
2317     \@@_extract_coords:w ( row - \int_eval:n { #3 + 1 } )
2318     \dim_set_eq:NN \l_tmpa_dim \pgf@y
2319     \@@_extract_coords:w ( col - \int_eval:n { #4 + 1 } )
2320     \dim_set_eq:NN \l_tmpb_dim \pgf@x
2321     \coordinate ( two ) at ( \l_tmpb_dim , \l_tmpa_dim ) ;
2322     \bool_if:NT \l_@@_white_bool
2323     {
2324       \fill [ white , line~width = 0 pt]
2325         ( [ yshift = -\arrayrulewidth ] one )
2326         rectangle
2327         ( [ xshift = -\arrayrulewidth ] two ) ;
2328     }

```

Now, we can construct the real node fitting the nodes `one` and `two`. We have to insert `\l_@@_tikz_tl` in the list of the options and that's why we use `\use:x`.

```

2329     \use:x
2330     {

```

```

2331         \exp_not:N \node
2332         [
2333             fit = ( one ) ( two ) ,
2334             inner-sep = 0 pt ,
2335             \l_@@_tikz_tl
2336         ]
2337     }

```

The node of the block has a name (#1-#2-block) and we will use it internally just after to put the label of the block.

```

2338         (#1-#2-block) { } ;

```

If the creation of the “medium nodes” is required, we create a “medium node” for the block.

```

2339     \bool_if:NT \l_@@_medium_nodes_bool
2340     {
2341         \node
2342         [
2343             fit = ( #1 - #2 - medium . north-west )
2344                 ( #3 - #4 - medium . south-east ) ,
2345             inner-sep = 0 pt ,
2346         ]
2347         (#1-#2-block-medium) { } ;
2348     }

```

Now, we will put the label of the block.

```

2349     \int_compare:nNnTF { #1 } = { #3 }
2350     {

```

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 `\coordinate` 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`.

```

2351         \@@_extract_coords:w (row-#1-base)
2352         \dim_set_eq:NN \l_tmpa_dim \pgf@y

```

We retrieve (in `\pgf@x`) the  $x$ -value of the center of the block.

```

2353         \@@_extract_coords:w (#1-#2-block)

```

We put the label of the block which has been composed in `\l_@@_cell_box`.

```

2354         \path (\pgf@x,\l_tmpa_dim) node [ anchor = base ]
2355             { \box_use_drop:N \l_@@_cell_box } ;
2356     }

```

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`).

```

2357         { \node at (#1-#2-block.center) { \box_use_drop:N \l_@@_cell_box } ; }
2358     \end { tikzpicture }
2359 }
2360 \group_end:
2361 }

```

## How to draw the dotted lines transparently

```

2362 \cs_set_protected:Npn \@@_renew_matrix:
2363 {
2364     \RenewDocumentEnvironment { pmatrix } { { }
2365         { \pNiceMatrix }
2366         { \endpNiceMatrix }
2367     \RenewDocumentEnvironment { vmatrix } { { }
2368         { \vNiceMatrix }
2369         { \endvNiceMatrix }
2370     \RenewDocumentEnvironment { Vmatrix } { { }
2371         { \VNiceMatrix }
2372         { \endVNiceMatrix }

```



```

2373 \RenewDocumentEnvironment { bmatrix } { }
2374 { \bNiceMatrix }
2375 { \endbNiceMatrix }
2376 \RenewDocumentEnvironment { Bmatrix } { }
2377 { \BNiceMatrix }
2378 { \endBNiceMatrix }
2379 }

```

## Automatic arrays

```

2380 \cs_new_protected:Npn \@@_set_size:n #1-#2 \q_stop
2381 {
2382   \int_set:Nn \l_@@_nb_rows_int { #1 }
2383   \int_set:Nn \l_@@_nb_cols_int { #2 }
2384 }
2385 \NewDocumentCommand \AutoNiceMatrixWithDelims { m m O { } m O { } m ! O { } }
2386 {
2387   \int_zero_new:N \l_@@_nb_rows_int
2388   \int_zero_new:N \l_@@_nb_cols_int
2389   \@@_set_size:n #4 \q_stop
2390   \begin { NiceArrayWithDelims } { #1 } { #2 }
2391     { * { \l_@@_nb_cols_int } { C } } [ #3 , #5 , #7 ]
2392     \int_compare:nNnT \l_@@_first_row_int = 0
2393     {
2394       \int_compare:nNnT \l_@@_first_col_int = 0 { & }
2395       \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
2396       \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
2397     }
2398     \prg_replicate:nn \l_@@_nb_rows_int
2399     {
2400       \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).

```

2401     \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { { } #6 & } #6
2402     \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
2403   }
2404   \int_compare:nNnT \l_@@_last_row_int > { -2 }
2405   {
2406     \int_compare:nNnT \l_@@_first_col_int = 0 { & }
2407     \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
2408     \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
2409   }
2410   \end { NiceArrayWithDelims }
2411 }
2412 \cs_set_protected:Npn \@@_define_com:nnn #1 #2 #3
2413 {
2414   \cs_set_protected:cpn { #1 AutoNiceMatrix }
2415   {
2416     \str_gset:Nx \g_@@_name_env_str { #1 AutoNiceMatrix }
2417     \AutoNiceMatrixWithDelims { #2 } { #3 }
2418   }
2419 }
2420 \@@_define_com:nnn p ( )
2421 \@@_define_com:nnn b [ ]
2422 \@@_define_com:nnn v | |
2423 \@@_define_com:nnn V \l \l
2424 \@@_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.

```

2425 \keys_define:nn { NiceMatrix / Package }
2426 {
2427   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
2428   renew-dots .value_forbidden:n = true ,
2429   renew-matrix .code:n = \@@_renew_matrix: ,
2430   renew-matrix .value_forbidden:n = true ,
2431   transparent .meta:n = { renew-dots , renew-matrix } ,
2432   transparent .value_forbidden:n = true,
2433   obsolete-environments .code:n =
2434     \@@_msg_redirect_name:nn { Obsolete-environment } { none }
2435 }
2436 \ProcessKeysOptions { NiceMatrix / Package }

```

## Error messages of the package

The following command converts all the elements of a sequence (which are token lists) into strings.

```

2437 \cs_new_protected:Npn \@@_convert_to_str_seq:N #1
2438 {
2439   \seq_clear:N \l_tmpa_seq
2440   \seq_map_inline:Nn #1
2441     {
2442       \seq_put_left:Nx \l_tmpa_seq { \tl_to_str:n { ##1 } }
2443     }
2444   \seq_set_eq:NN #1 \l_tmpa_seq
2445 }

```

The following command creates a sequence of strings (`str`) from a `clist`.

```

2446 \cs_new_protected:Npn \@@_set_seq_of_str_from_clist:Nn #1 #2
2447 {
2448   \seq_set_from_clist:Nn #1 { #2 }
2449   \@@_convert_to_str_seq:N #1
2450 }
2451 \@@_set_seq_of_str_from_clist:Nn \c_@@_types_of_matrix_seq
2452 {
2453   NiceMatrix ,
2454   pNiceMatrix , bNiceMatrix , vNiceMatrix, BNiceMatrix, VNiceMatrix
2455 }

```

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`.

```

2456 \cs_new_protected:Npn \@@_error_too_much_cols:
2457 {
2458   \seq_if_in:NVTF \c_@@_types_of_matrix_seq \g_@@_name_env_str
2459     {
2460       \int_compare:nNnTF \l_@@_last_col_int = { -1 }
2461       { \@@_fatal:n { too-much-cols-for-matrix } }
2462       { \@@_fatal:n { too-much-cols-for-matrix-with-last-col } }
2463     }
2464     { \@@_fatal:n { too-much-cols-for-array } }
2465 }
2466 \@@_msg_new:nn { too-much-cols-for-matrix-with-last-col }
2467 {
2468   You-try-to-use-more-columns-than-allowed-by-your-
2469   \@@_full_name_env:~The-maximal-number-of-columns-is-
2470   \int_eval:n { \l_@@_last_col_int - 1 }~(plus-the-potential-

```

```

2471     exterior~ones).~This~error~is~fatal.
2472 }
2473 \@@_msg_new:nn { too~much~cols~for~matrix }
2474 {
2475     You~try~to~use~more~columns~than~allowed~by~your~
2476     \@@_full_name_env:~ Recall~that~the~maximal~number~of~columns~
2477     for~a~matrix~is~fixed~by~the~LaTeX~counter~'MaxMatrixCols'.~
2478     Its~actual~value~is~\int_use:N \c@MaxMatrixCols.~This~error~is~fatal.
2479 }
2480 \@@_msg_new:nn { too~much~cols~for~array }
2481 {
2482     You~try~to~use~more~columns~than~allowed~by~your~
2483     \@@_full_name_env:~The~maximal~number~of~columns~is~
2484     \int_use:N \c@jCol\space (plus~the~potential~exterior~ones).~
2485     This~error~is~fatal.
2486 }
2487 \@@_msg_new:nn { bad~value~for~baseline }
2488 {
2489     The~value~you~gave~to~'baseline'~(\int_use:N \l_tmpa_int)~is~not~
2490     valid.\\
2491     If~you~go~on,~a~value~of~1~will~be~used.
2492 }
2493 \@@_msg_new:nn { Second~Block }
2494 {
2495     You~can't~use~\token_to_str:N \Block\ twice~in~the~same~cell~of~the~array.\\
2496     If~you~go~on,~this~command~(and~the~other)~will~be~ignored.
2497 }
2498 \@@_msg_new:nn { empty~environment }
2499 { Your~\@@_full_name_env:~ is~empty.~This~error~is~fatal. }
2500 \@@_msg_new:nn { unknown~cell~for~line~in~code~after }
2501 {
2502     Your~command~\token_to_str:N\line\{#1\}\{#2\}~in~the~'code~after'~
2503     can't~be~executed~because~a~Tikz~node~doesn't~exist.\\
2504     If~you~go~on~this~command~will~be~ignored.
2505 }
2506 \@@_msg_new:nn { last~col~non~empty~for~NiceArray }
2507 {
2508     In~the~\@@_full_name_env:,~you~must~use~the~option~
2509     'last~col'~without~value.\\
2510     However,~you~can~go~on~for~this~time~
2511     (the~value~'\l_keys_value_tl'~will~be~ignored).
2512 }
2513 \@@_msg_new:nn { last~col~empty~for~NiceMatrix }
2514 {
2515     In~the~\@@_full_name_env:,~you~can't~use~the~option~
2516     'last~col'~without~value.~You~must~give~the~number~of~that~last~column.\\
2517     If~you~go~on~this~option~will~be~ignored.
2518 }
2519 \@@_msg_new:nn { Block~too~large }
2520 {
2521     You~try~to~draw~a~block~in~the~cell~#1~#2~of~your~matrix~but~the~matrix~is~
2522     too~small~for~that~block. \\
2523     If~you~go~on,~this~command~will~be~ignored.
2524 }
2525 \@@_msg_new:nn { Impossible~line }
2526 {
2527     A~dotted~line~can't~be~drawn~because~you~have~not~put~
2528     all~the~ampersands~required~on~the~row~#1.\\
2529     If~you~go~on,~this~dotted~line~will~be~ignored.
2530 }

```

```

2531 \@@_msg_new:nn { Wrong-last-row }
2532 {
2533   You~have~used~'last-row=\int_use:N \l_@@_last_row_int'~but~your~
2534   \@@_full_name_env:\ seems~to~have~\int_use:N \c@iRow \ rows.~
2535   If~you~go~on,~the~value~of~\int_use:N \c@iRow \ will~be~used~for~
2536   last-row.~You~can~avoid~this~problem~by~using~'last-row'~
2537   without~value~(more~compilations~might~be~necessary).
2538 }
2539 \@@_msg_new:nn { Yet-in-env }
2540 {
2541   Environments~\{NiceArray\}~(or~\{NiceMatrix\},~etc.)~can't~be~nested.\\
2542   This~error~is~fatal.
2543 }
2544 \@@_msg_new:nn { Outside-math-mode }
2545 {
2546   The~\@@_full_name_env:\ can~be~used~only~in~math-mode~
2547   (and~not~in~\token_to_str:N \vcenter).\\
2548   This~error~is~fatal.
2549 }
2550 \@@_msg_new:nn { Bad-value-for-letter-for-dotted-lines }
2551 {
2552   The~value~of~key~'\tl_use:N\l_keys_key_str'~must~be~of~length~1.\\
2553   If~you~go~on,~it~will~be~ignored.
2554 }
2555 \@@_msg_new:nnn { Unknown-key-for-NiceMatrixOptions }
2556 {
2557   The~key~'\tl_use:N\l_keys_key_str'~is~unknown~for~the~command~
2558   \token_to_str:N \NiceMatrixOptions. \\
2559   If~you~go~on,~it~will~be~ignored. \\
2560   For~a~list~of~the~available~keys,~type~H<return>.
2561 }
2562 {
2563   The~available~options~are~(in~alphabetic~order):~
2564   allow-duplicate-names,~
2565   code-for-first-col,~
2566   code-for-first-row,~
2567   code-for-last-col,~
2568   code-for-last-row,~
2569   create-extra-nodes,~
2570   create-medium-nodes,~
2571   create-large-nodes,~
2572   dotted-lines-margin,~
2573   end-of-row,~
2574   exterior-arraycolsep,~
2575   hlines,~
2576   hvlines,~
2577   left-margin,~
2578   letter-for-dotted-lines,~
2579   light-syntax,~
2580   nullify-dots,~
2581   parallelize-diags,~
2582   renew-dots,~
2583   renew-matrix,~
2584   right-margin,~
2585   small,~
2586   transparent~
2587   and~vlines.
2588 }
2589 \@@_msg_new:nnn { Unknown-option-for-NiceArray }
2590 {
2591   The~option~'\tl_use:N\l_keys_key_str'~is~unknown~for~the~environment~
2592   \{NiceArray\}. \\

```

```

2593 If~you~go~on,~it~will~be~ignored. \\
2594 For~a~list~of~the~available~options,~type~H~<return>.
2595 }
2596 {
2597 The~available~options~are~(in~alphabetic~order):~
2598 b,~
2599 baseline,~
2600 c,~
2601 code~after,~
2602 code~for~first~col,~
2603 code~for~first~row,~
2604 code~for~last~col,~
2605 code~for~last~row,~
2606 columns~width,~
2607 create~extra~nodes,~
2608 create~medium~nodes,~
2609 create~large~nodes,~
2610 dotted~lines~margin,~
2611 end~of~row,~
2612 extra~left~margin,~
2613 extra~right~margin,~
2614 first~col,~
2615 first~row,~
2616 hlines,~
2617 hvlines,~
2618 last~col,~
2619 last~row,~
2620 left~margin,~
2621 light~syntax,~
2622 name,~
2623 nullify~dots,~
2624 parallelize~diags,~
2625 renew~dots,~
2626 right~margin,~
2627 small,~
2628 t,~
2629 and~vlines.
2630 }

```

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).

```

2631 \@@_msg_new:nnn { Unknown~option~for~NiceMatrix }
2632 {
2633 The~option~'\tl_use:N\l_keys_key_str'~is~unknown~for~the~
2634 \@@_full_name_env:. \\
2635 If~you~go~on,~it~will~be~ignored. \\
2636 For~a~list~of~the~available~options,~type~H~<return>.
2637 }
2638 {
2639 The~available~options~are~(in~alphabetic~order):~
2640 code~after,~
2641 code~for~first~col,~
2642 code~for~first~row,~
2643 code~for~last~col,~
2644 code~for~last~row,~
2645 columns~width,~
2646 create~extra~nodes,~
2647 create~medium~nodes,~
2648 create~large~nodes,~
2649 dotted~lines~margin,~
2650 end~of~row,~
2651 extra~left~margin,~
2652 extra~right~margin,~
2653 first~col,~

```

```

2654     first-row,~
2655     hlines,~
2656     hvlines,~
2657     last-col,~
2658     last-row,~
2659     left-margin,~
2660     light-syntax,~
2661     name,~
2662     nullify-dots,~
2663     parallelize-diags,~
2664     renew-dots,~
2665     right-margin,~
2666     small,~
2667     and~vlines.
2668 }

2669 \@@_msg_new:nnn { Duplicate-name }
2670 {
2671     The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
2672     the~same~environment~name~twice.~You~can~go~on,~but,~
2673     maybe,~you~will~have~incorrect~results~especially~
2674     if~you~use~'columns-width=auto'.~If~you~don't~want~to~see~this~
2675     message~again,~use~the~option~'allow-duplicate-names'.\\
2676     For~a~list~of~the~names~already~used,~type~H~<return>. \\
2677 }
2678 {
2679     The~names~already~defined~in~this~document~are:~
2680     \seq_use:Nnnn \g_@@_names_seq { ,~ } { ,~ } { ~and~ }.
2681 }

2682 \@@_msg_new:nn { Option-auto-for-columns-width }
2683 {
2684     You~can't~give~the~value~'auto'~to~the~option~'columns-width'~here.~
2685     If~you~go~on,~the~option~will~be~ignored.
2686 }

2687 \@@_msg_new:nn { Zero~row }
2688 {
2689     There~is~a~problem.~Maybe~you~have~used~l,~c~and~r~instead~of~L,~C~
2690     and~R~in~the~preamble~of~your~environment. \\
2691     This~error~is~fatal.
2692 }

2693 \@@_msg_new:nn { Use-of-:~in-first-position }
2694 {
2695     You~can't~use~the~column~specifier~'\l_@@_letter_for_dotted_lines_str'~in~the~
2696     first~position~of~the~preamble~of~the~\@@_full_name_env:. \\
2697     If~you~go~on,~this~dotted~line~will~be~ignored.
2698 }

```

## Obsolete environments

```

2699 \@@_msg_new:nn { Obsolete~environment }
2700 {
2701     The~environment~\{\@currenvir\}~is~obsolete.~You~should~use~#1~instead.~
2702     However,~it's~still~possible~to~use~the~environment~\{\@currenvir\}~(for~
2703     a~few~months)~by~loading~'nicematrix'~with~the~option~
2704     'obsolete-environments'.
2705 }

2706 \NewDocumentEnvironment { pNiceArrayC } { }
2707 {
2708     \@@_fatal:nn { Obsolete~environment } { the~option~'last-col' }
2709     \int_zero:N \l_@@_last_col_int

```

```

2710 \pNiceArray
2711 }
2712 { \endpNiceArray }

2713 \NewDocumentEnvironment { bNiceArrayC } { }
2714 {
2715   \@@_fatal:nn { Obsolete-environment } { the-option~'last-col' }
2716   \int_zero:N \l_@@_last_col_int
2717   \bNiceArray
2718 }
2719 { \endbNiceArray }

2720 \NewDocumentEnvironment { BNiceArrayC } { }
2721 {
2722   \@@_fatal:nn { Obsolete-environment } { the-option~'last-col' }
2723   \int_zero:N \l_@@_last_col_int
2724   \BNiceArray
2725 }
2726 { \endBNiceArray }

2727 \NewDocumentEnvironment { vNiceArrayC } { }
2728 {
2729   \@@_fatal:nn { Obsolete-environment } { the-option~'last-col' }
2730   \int_zero:N \l_@@_last_col_int
2731   \vNiceArray
2732 }
2733 { \endvNiceArray }

2734 \NewDocumentEnvironment { VNiceArrayC } { }
2735 {
2736   \@@_fatal:nn { Obsolete-environment } { the-option~'last-col' }
2737   \int_zero:N \l_@@_last_col_int
2738   \VNiceArray
2739 }
2740 { \endVNiceArray }

2741 \NewDocumentEnvironment { pNiceArrayRC } { }
2742 {
2743   \@@_fatal:nn { Obsolete-environment }
2744     { the-options~'last-col'~and~'first-row' }
2745   \int_zero:N \l_@@_last_col_int
2746   \int_zero:N \l_@@_first_row_int
2747   \pNiceArray
2748 }
2749 { \endpNiceArray }

2750 \NewDocumentEnvironment { bNiceArrayRC } { }
2751 {
2752   \@@_fatal:nn { Obsolete-environment }
2753     { the-options~'last-col'~and~'first-row' }
2754   \int_zero:N \l_@@_last_col_int
2755   \int_zero:N \l_@@_first_row_int
2756   \bNiceArray
2757 }
2758 { \endbNiceArray }

2759 \NewDocumentEnvironment { BNiceArrayRC } { }
2760 {
2761   \@@_fatal:nn { Obsolete-environment }
2762     { the-options~'last-col'~and~'first-row' }
2763   \int_zero:N \l_@@_last_col_int
2764   \int_zero:N \l_@@_first_row_int
2765   \BNiceArray
2766 }
2767 { \endBNiceArray }

2768 \NewDocumentEnvironment { vNiceArrayRC } { }
2769 {

```

```

2770 \@@_fatal:nn { Obsolete-environment }
2771 { the~options~'last-col'~and~'first-row' }
2772 \int_zero:N \l_@@_last_col_int
2773 \int_zero:N \l_@@_first_row_int
2774 \vNiceArray
2775 }
2776 { \endvNiceArray }

2777 \NewDocumentEnvironment { VNiceArrayRC } { }
2778 {
2779 \@@_fatal:nn { Obsolete-environment }
2780 { the~options~'last-col'~and~'first-row' }
2781 \int_zero:N \l_@@_last_col_int
2782 \int_zero:N \l_@@_first_row_int
2783 \VNiceArray
2784 }
2785 { \endVNiceArray }

2786 \NewDocumentEnvironment { NiceArrayCwithDelims } { }
2787 {
2788 \@@_fatal:nn { Obsolete-environment }
2789 { the~option~'last-col' }
2790 \int_zero:N \l_@@_last_col_int
2791 \NiceArrayWithDelims
2792 }
2793 { \endNiceArrayWithDelims }

2794 \NewDocumentEnvironment { NiceArrayRCwithDelims } { }
2795 {
2796 \@@_fatal:nn { Obsolete-environment }
2797 { the~options~'last-col'~and~'first-row' }
2798 \int_zero:N \l_@@_last_col_int
2799 \int_zero:N \l_@@_first_row_int
2800 \NiceArrayWithDelims
2801 }
2802 { \endNiceArrayWithDelims }

```

## 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>37</sup>, Tikz externalization is now deactivated in the environments of the extension `nicematrix`.<sup>38</sup>

## Changes between version 2.1 and 2.1.2

Option `draft`: with this option, the dotted lines are not drawn (quicker).

## Changes between version 2.1.2 and 2.1.3

When searching the end of a dotted line from a command like `\Cdots` issued in the “main matrix” (not in the exterior column), the cells in the exterior column are considered as outside the matrix. That means that it’s possible to do the following matrix with only a `\Cdots` command (and a single `\Vdots`).

$$\left( \begin{array}{ccc} & C_j & \\ 0 & \vdots & 0 \\ & a \cdots \cdots & \\ 0 & & 0 \end{array} \right)_{L_i}$$

## Changes between version 2.1.3 and 2.1.4

Replacement of some options `0 { }` in commands and environments defined with `xparse` by `! 0 { }` (because a recent version of `xparse` introduced the specifier `!` and modified the default behaviour of the last optional arguments).

See [www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end](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 `\dashline` 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`).

---

<sup>37</sup>cf. [tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package](https://tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package)

<sup>38</sup>Before this version, there was an error when using `nicematrix` with Tikz externalization. In any case, it’s not possible to externalize the Tikz elements constructed by `nicematrix` because they use the options `overlay` and `remember picture`.

## Changes between version 2.2 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>39</sup>, optimization of Tikz externalization is disabled in the environments of `nicematrix` when the class `standalone` or the package `standalone` is used.

## Changes between version 3.4 and 3.5

Correction on a bug on the two previous versions where the `code-after` was not executed.

---

<sup>39</sup>cf. [tex.stackexchange.com/questions/510841/nicematrix-and-tikz-external-optimize](https://tex.stackexchange.com/questions/510841/nicematrix-and-tikz-external-optimize)

## Changes between version 3.5 and 3.6

LaTeX counters `iRow` and `jCol` available in the cells of the array.

Addition of `\normalbaselines` before the construction of the array: in environments like `{align}` of `amsmath` the value of `\baselineskip` is changed and if the options `first-row` and `last-row` were used in an environment of `nicematrix`, the position of the delimiters was wrong.

A warning is written in the `.log` file if an obsolete environment is used.

There is no longer artificial errors `Duplicate-name` in the environments of `amsmath`.

## Changes between version 3.6 and 3.7

The four “corners” of the matrix are correctly protected against the four codes: `code-for-first-col`, `code-for-last-col`, `code-for-first-row` and `code-for-last-row`.

New command `\pAutoNiceMatrix` and its variants (suggestion of Christophe Bal).

## Changes between version 3.7 and 3.8

New programming for the command `\Block` when the block has only one row. With this programming, the vertical rules drawn by the specifier “|” at the end of the block is actually drawn. In previous versions, they were not because the block of one row was constructed with `\multicolumn`.

An error is raised when an obsolete environment is used.

## Changes between version 3.8 and 3.9

New commands `\NiceMatrixLastEnv` and `\OnlyMainNiceMatrix`.

New options `create-medium-nodes` and `create-large-nodes`.

## Changes between version 3.9 and 3.10

New option `light-syntax` (and `end-of-row`).

New option `dotted-lines-margin` for fine tuning of the dotted lines.

## Changes between versions 3.10 and 3.11

Correction of a bug linked to `first-row` and `last-row`.

## Changes between versions 3.11 and 3.12

Command `\rotate` in the cells of the array.

Options `vlines`, `hlines` and `hvlines`.

Option `baseline` pour `{NiceArray}` (uniquement).

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.

# Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols	
@@ commands:	<code>\@@_Block_ii:nnnnn</code> ..... 2267, 2268
<code>\@@_Block:</code> ..... 524, 2265	<code>\@@_Block_iii:nnnnnn</code> ..... 2272, 2295
<code>\@@_Block_error:nn</code> ..... 2280, 2282	<code>\@@_Cdots</code> ..... 516, 530, 1809
<code>\@@_Block_i</code> ..... 2266, 2267	<code>\g_@@_Cdots_lines_tl</code> ..... 570, 1274
	<code>\@@_Cell:</code> ..... 145, 319, 402, 510, 511, 512

<code>\@@_Ddots</code>	518, 532, 1821	<code>\@@_create_medium_and_large_nodes:</code>	1255, 2137, 2149, 2152, 2162
<code>\g_@@_Ddots_lines_tl</code>	573, 1272	<code>\@@_create_medium_nodes:</code>	1256, 1456, 1457, 1908, 2129, 2136, 2150, 2163
<code>\@@_Hdotsfor:</code>	522, 535, 1850	<code>\@@_create_nodes:</code>	2134, 2146, 2157, 2160, 2205
<code>\@@_Hdotsfor:nnm</code>	1865, 1873	<code>\@@_ddots</code>	1795, 1824
<code>\@@_Hdotsfor_i</code>	1853, 1857, 1861	<code>\l_@@_ddots_int</code>	1245, 1626, 1627
<code>\g_@@_Hdotsfor_lines_tl</code>	575, 1270, 1863	<code>\@@_define_com:nnn</code>	2412, 2420, 2421, 2422, 2423, 2424
<code>\@@_Hspace:</code>	521, 1833	<code>\@@_define_env:n</code>	1145, 1164, 1165, 1166, 1167, 1168, 1169
<code>\@@_Iddots</code>	519, 533, 1827	<code>\l_@@_delta_x_one_dim</code>	1247, 1629, 1639
<code>\g_@@_Iddots_lines_tl</code>	574, 1273	<code>\l_@@_delta_x_two_dim</code>	1249, 1671, 1681
<code>\@@_Ldots</code>	515, 529, 534, 1803	<code>\l_@@_delta_y_one_dim</code>	1248, 1631, 1639
<code>\g_@@_Ldots_lines_tl</code>	571, 1275	<code>\l_@@_delta_y_two_dim</code>	1250, 1673, 1681
<code>\l_@@_NiceArray_bool</code>	64, 596, 632, 662, 673, 722, 1099, 1314, 1315	<code>\l_@@_dotted_lines_margin_dim</code>	86, 87, 179, 1723, 1734, 1742
<code>\g_@@_NiceMatrixBlock_int</code>	60, 2055, 2060, 2063, 2074	<code>\@@_double_int_eval:n</code>	1920, 1927, 1928
<code>\@@_OnlyMainNiceMatrix:n</code>	526, 1966, 1984	<code>\g_@@_dp_ante_last_row_dim</code>	342, 502
<code>\@@_OnlyMainNiceMatrix_i:n</code>	1969, 1976, 1979	<code>\g_@@_dp_last_row_dim</code>	342, 343, 505, 506, 606, 607, 790, 1306
<code>\@@_Vdots</code>	517, 531, 1815	<code>\g_@@_dp_row_zero_dim</code>	352, 353, 496, 497, 739, 773, 784
<code>\g_@@_Vdots_lines_tl</code>	572, 1271	<code>\c_@@_draft_bool</code>	18, 19, 43, 413, 1855, 1933, 1988, 2022
<code>\@@_actually_draw_Ldots:</code>	1505, 1508, 1911	<code>\@@_draw_Cdots:nn</code>	1530
<code>\@@_adapt_S_column:</code>	120, 135, 582	<code>\@@_draw_Ddots:nn</code>	1605
<code>\@@_add_to_empty_cells:</code>	1797, 1807, 1813, 1819, 1825, 1831, 1835	<code>\@@_draw_Iddots:nn</code>	1647
<code>\@@_adjust_with_col_nodes:</code>	1481, 1521, 1549, 1910	<code>\@@_draw_Ldots:nn</code>	1499
<code>\@@_after_array:</code>	817, 1220	<code>\@@_draw_Vdots:nn</code>	1558
<code>\@@_analyze_end:Nn</code>	863, 898	<code>\@@_draw_tikz_line:</code>	1528, 1554, 1601, 1643, 1685, 1691, 1957, 2016, 2041
<code>\@@_array:</code>	427, 864, 873	<code>\@@_draw_vlines:</code>	1285, 1293
<code>\l_@@_auto_columns_width_bool</code>	161, 220, 915, 921, 2050	<code>\@@_end_Cell:</code>	147, 365, 406, 510, 511, 512
<code>\l_@@_baseline_str</code>	153, 154, 298, 299, 300, 301, 438, 724, 744, 747, 748, 749	<code>\l_@@_end_of_row_tl</code>	173, 174, 183, 875, 876
<code>\@@_begin_of_row:</code>	325, 339, 962	<code>\g_@@_env_int</code>	59, 346, 382, 449, 592, 615, 618, 727, 731, 761, 765, 912, 938, 952, 1004, 1082, 1180, 1193, 1203, 1230, 1282, 1383, 1444, 1467, 1473, 1486, 1494, 1690, 1801, 1943, 1945, 2104, 2107, 2117, 2222, 2258
<code>\l_@@_block_auto_columns_width_bool</code>	593, 922, 2043, 2048, 2058, 2068	<code>\@@_error:n</code>	25, 271, 280, 283, 292, 294, 304, 306, 312, 317, 708, 756, 2021, 2284
<code>\@@_cdots</code>	1793, 1812	<code>\@@_error:nn</code>	26, 228
<code>\l_@@_cell_box</code>	329, 353, 355, 361, 370, 393, 401, 409, 469, 605, 607, 963, 988, 1015, 1034, 1059, 1093, 1919, 2306, 2355, 2357	<code>\@@_error:nnn</code>	27, 1960
<code>\@@_cell_with_light_syntax:n</code>	890, 897	<code>\@@_error_too_much_cols:</code>	679, 2456
<code>\g_@@_cells_seq</code>	886, 887, 888, 890	<code>\@@_everycr:</code>	441, 491, 494
<code>\g_@@_code_after_tl</code>	99, 233, 561, 1287, 1288, 1996, 2270	<code>\@@_everycr_i:</code>	441, 442
<code>\l_@@_code_for_first_col_tl</code>	185, 976	<code>\l_@@_exterior_arraycolsep_bool</code>	155, 268, 664, 675
<code>\l_@@_code_for_first_row_tl</code>	189, 333	<code>\l_@@_extra_left_margin_dim</code>	171, 211, 687, 1020
<code>\l_@@_code_for_last_col_tl</code>	187, 1047	<code>\l_@@_extra_right_margin_dim</code>	172, 212, 699, 1068
<code>\l_@@_code_for_last_row_tl</code>	191, 336	<code>\@@_extract_coords:w</code>	82, 727, 729, 760, 763, 1298, 1300, 1307, 1309, 1466, 1472, 1486, 1493, 1950, 1953, 2008, 2011, 2013, 2033, 2036, 2038, 2106, 2116, 2312, 2314, 2317, 2319, 2351, 2353
<code>\g_@@_col_total_int</code>	327, 328, 541, 713, 942, 943, 1032, 1033, 2091, 2101, 2209	<code>\@@_extract_coords_values:</code>	2235, 2242
<code>\c_@@_colortbl_loaded_bool</code>	72, 77, 459, 486, 1296	<code>\@@_fatal:n</code>	28, 68, 584, 872, 901, 907, 2461, 2462, 2464
<code>\l_@@_columns_width_dim</code>	61, 221, 272, 916, 930, 2056, 2062	<code>\@@_fatal:nn</code>	29, 2708, 2715, 2722, 2729, 2736, 2743, 2752, 2761, 2770, 2779, 2788, 2796
<code>\g_@@_com_or_env_str</code>	91, 92, 95		
<code>\@@_computations_for_large_nodes:</code>	2144, 2158, 2166		
<code>\@@_computations_for_medium_nodes:</code>	2082, 2132, 2143, 2155		
<code>\@@_convert_to_str_seq:N</code>	2437, 2449		
<code>\@@_create_col_nodes:</code>	867, 880, 904		
<code>\@@_create_large_nodes:</code>	1258, 2138, 2140, 2148, 2164		

<code>\l_@@_final_i_int</code>	1261, 1333, 1338, 1341, 1362, 1367, 1373, 1384, 1391, 1474, 1878, 1900	2185, 2189, 2191, 2195, 2196, 2209, 2212, 2215, 2222, 2225, 2238, 2245, 2250, 2258, 2261
<code>\l_@@_final_j_int</code>	..... 1262, 1334, 1339, 1347, 1353, 1363, 1367, 1374, 1385, 1475, 1895, 1901, 1903	<code>\l_@@_l_dim</code> . 1693, 1694, 1710, 1717, 1723, 1734, 1742, 1750, 1755, 1767, 1768, 1775, 1776
<code>\l_@@_final_open_bool</code>	..... 1264, 1340, 1344, 1350, 1356, 1360, 1388, 1457, 1490, 1517, 1524, 1545, 1552, 1573, 1586, 1620, 1662, 1714, 1729, 1760, 1761, 1876, 1896, 1904, 1907, 1940, 2006, 2031	<code>\g_@@_large_nodes_bool</code> 166, 483, 1254, 1258 <code>\l_@@_large_nodes_bool</code> ..... 165, 202, 483 <code>\g_@@_last_col_found_bool</code> ..... ..... 112, 569, 714, 815, 941, 1030
<code>\@@_find_extremities_of_line:nnnn</code>	..... 1328, 1504, 1535, 1563, 1610, 1652	<code>\l_@@_last_col_int</code> ..... 110, 111, 293, 305, 313, 668, 1155, 1157, 2396, 2402, 2408, 2460, 2470, 2709, 2716, 2723, 2730, 2737, 2745, 2754, 2763, 2772, 2781, 2790, 2798
<code>\l_@@_first_col_int</code>	.. 105, 106, 235, 310, 324, 657, 717, 908, 1968, 2091, 2101, 2169, 2209, 2394, 2400, 2406	<code>\l_@@_last_row_int</code> ..... 107, 108, 237, 315, 335, 457, 555, 600, 610, 617, 624, 702, 706, 709, 716, 787, 971, 973, 1042, 1044, 1303, 1974, 1982, 2404, 2533
<code>\l_@@_first_row_int</code>	..... 103, 104, 236, 314, 539, 736, 770, 781, 2084, 2098, 2168, 2207, 2392, 2746, 2755, 2764, 2773, 2782, 2799	<code>\l_@@_last_row_without_value_bool</code> ... ..... 109, 612, 704, 1225
<code>\@@_full_name_env:</code>	..... 93, 2469, 2476, 2483, 2499, 2508, 2515, 2534, 2546, 2634, 2696	<code>\g_@@_last_vdotted_col_int</code> 558, 560, 566, 568 <code>\@@_ldots</code> ..... 1792, 1806
<code>\@@_hdottedline:</code>	..... 520, 1986	<code>\l_@@_left_delim_dim</code> 630, 634, 644, 856, 1018
<code>\@@_hdottedline:n</code>	..... 1997, 1999	<code>\l_@@_left_margin_dim</code> ..... ..... 167, 205, 686, 1019, 2200
<code>\@@_hdottedline:i:</code>	..... 1991, 1994	<code>\l_@@_letter_for_dotted_lines_str</code> ... ..... 279, 285, 286, 548, 549, 2695
<code>\l_@@_hlines_bool</code>	..... 158, 194, 451	<code>\l_@@_light_syntax_bool</code> . 152, 181, 689, 694
<code>\g_@@_ht_last_row_dim</code>	..... 344, 503, 504, 604, 605, 790, 1306	<code>\@@_line:nn</code> ..... 1286, 1922
<code>\g_@@_ht_row_one_dim</code>	..... 360, 361, 500, 501	<code>\@@_line_i:nn</code> ..... 1926, 1931
<code>\g_@@_ht_row_zero_dim</code>	..... 354, 355, 498, 499, 739, 773, 784	<code>\@@_line_with_light_syntax:n</code> .... 879, 892
<code>\@@_i:</code>	..... 2084, 2086, 2087, 2088, 2089, 2098, 2104, 2108, 2109, 2110, 2111, 2117, 2118, 2119, 2120, 2170, 2172, 2175, 2176, 2180, 2181, 2207, 2213, 2216, 2222, 2225, 2237, 2246, 2251, 2258, 2261	<code>\@@_line_with_light_syntax:i:n</code> 878, 884, 895
<code>\@@_iddots</code>	..... 1796, 1830	<code>\g_@@_max_cell_width_dim</code> ..... ..... 369, 370, 594, 926, 2049, 2075
<code>\l_@@_iddots_int</code>	..... 1246, 1668, 1669	<code>\l_@@_max_delimiter_width_bool</code> 175, 178, 811
<code>\@@_if_not_empty_cell:nn</code>	..... 1170	<code>\g_@@_medium_nodes_bool</code> .... 164, 482, 1252
<code>\@@_if_not_empty_cell:nnTF</code>	..... 1367, 1427, 1886, 1900	<code>\l_@@_medium_nodes_bool</code> 163, 201, 482, 2339
<code>\l_@@_impossible_line_bool</code>	..... 71, 1392, 1453, 1503, 1505, 1534, 1536, 1562, 1564, 1609, 1611, 1651, 1653	<code>\@@_msg_new:nn</code> ..... ..... 30, 41, 2466, 2473, 2480, 2487, 2493, 2498, 2500, 2506, 2513, 2519, 2525, 2531, 2539, 2544, 2550, 2682, 2687, 2693, 2699
<code>\l_@@_in_env_bool</code>	..... 63, 584, 585	<code>\@@_msg_new:nnn</code> .. 31, 2555, 2589, 2631, 2669
<code>\l_@@_initial_i_int</code>	..... 1259, 1331, 1398, 1401, 1422, 1428, 1434, 1445, 1452, 1468, 1877, 1886	<code>\@@_msg_redirect_name:nn</code> .... 32, 274, 2434
<code>\l_@@_initial_j_int</code>	..... 1260, 1332, 1399, 1407, 1413, 1423, 1428, 1435, 1446, 1469, 1881, 1887, 1889	<code>\@@_multicolumn:nnn</code> ..... 523, 1839
<code>\l_@@_initial_open_bool</code>	..... 1263, 1400, 1404, 1410, 1416, 1420, 1449, 1456, 1483, 1512, 1522, 1540, 1550, 1568, 1581, 1615, 1657, 1712, 1759, 1875, 1882, 1890, 1907, 1939, 2005, 2030	<code>\g_@@_multicolumn_cells_seq</code> ..... ..... 537, 1844, 2111, 2120, 2231
<code>\@@_instruction_of_type:n</code>	..... 414, 416, 1805, 1811, 1817, 1823, 1829	<code>\g_@@_multicolumn_sizes_seq</code> 538, 1846, 2232
<code>\l_@@_inter_dots_dim</code>	..... 84, 85, 1268, 1717, 1724, 1735, 1743, 1750, 1755, 1767, 1775, 2037, 2039	<code>\g_@@_name_env_str</code> ..... ..... 90, 96, 97, 580, 581, 900, 1100, 1101, 1107, 1108, 1115, 1116, 1123, 1124, 1131, 1132, 1139, 1140, 1149, 1290, 2416, 2458
<code>\@@_j:</code>	..... 2091, 2093, 2094, 2095, 2096, 2101, 2104, 2108, 2111, 2113, 2114, 2117, 2120, 2122, 2123, 2183,	<code>\l_@@_name_str</code> ..... 162, 230, 386, 388, 613, 622, 625, 1008, 1010, 1086, 1088, 1233, 1237, 2224, 2225, 2260, 2261
		<code>\g_@@_names_seq</code> ..... 62, 227, 229, 2680
		<code>\l_@@_nb_cols_int</code> ..... ..... 2383, 2388, 2391, 2395, 2401, 2407
		<code>\l_@@_nb_rows_int</code> ..... 2382, 2387, 2398
		<code>\@@_node_for_multicolumn:nn</code> .... 2233, 2240
		<code>\l_@@_nullify_dots_bool</code> ..... ..... 160, 200, 1806, 1812, 1818, 1824, 1830
		<code>\@@_old_multicolumn</code> ..... 1838, 1841
		<code>\l_@@_parallelize_diags_bool</code> ..... ..... 156, 157, 197, 1243, 1624, 1666
		<code>\@@_pre_array:</code> ..... 467, 629

<code>\c_@@_preamble_first_col_tl</code> . . . . .	658, 958
<code>\c_@@_preamble_last_col_tl</code> . . . . .	669, 1026
<code>\@@_put_box_in_flow:</code> . . . . .	813, 819, 857
<code>\@@_put_box_in_flow_bis:nn</code> . . . . .	812, 825
<code>\l_@@_radius_dim</code> . . . . .	88, 89, 556, 1267, 1784, 1990, 2009, 2010, 2034, 2035
<code>\l_@@_real_left_delim_dim</code> . . .	827, 842, 856
<code>\l_@@_real_right_delim_dim</code> . .	828, 854, 858
<code>\@@_renew_NC@rewrite@S:</code> . . . . .	138, 567
<code>\l_@@_renew_dots_bool</code> . . . . .	198, 527, 2427
<code>\@@_renew_matrix:</code> . . . . .	264, 2362, 2429
<code>\@@_renewcolumnmtype:nn</code> . . . . .	396, 545, 546
<code>\@@_restore_iRow_jCol:</code> . . . . .	1291, 1323
<code>\@@_retrieve_coords:nn</code> . . . . .	1459, 1480, 1510, 1538, 1566, 1579, 1613, 1655
<code>\c_@@_revtex_bool</code> . . . . .	34, 36, 39, 429
<code>\l_@@_right_delim_dim</code> 631, 635, 652, 858, 1066	
<code>\l_@@_right_margin_dim</code> . . . . .	168, 207, 698, 1067, 2203
<code>\@@_rotate:</code> . . . . .	525, 1916
<code>\@@_rotate_i:</code> . . . . .	1916, 1917
<code>\@@_rotate_ii:</code> . . . . .	1917, 1918
<code>\@@_rotate_iii:</code> . . . . .	1918, 1919
<code>\g_@@_row_of_col_done_bool</code> 102, 455, 579, 910	
<code>\g_@@_row_total_int</code> . . . . .	540, 715, 753, 1231, 1238, 1309, 2084, 2098, 2207
<code>\g_@@_rows_seq</code> . . . . .	874, 876, 877, 879
<code>\l_@@_save_iRow_int</code> . . . . .	100, 471, 1325
<code>\l_@@_save_jCol_int</code> . . . . .	101, 474, 1326
<code>\@@_set_seq_of_str_from_clist:Nn</code> 2446, 2451	
<code>\@@_set_size:n</code> . . . . .	2380, 2389
<code>\c_@@_siunitx_loaded_bool</code> 113, 117, 122, 567	
<code>\l_@@_small_bool</code> 193, 331, 477, 965, 1036, 1265	
<code>\@@_standard_ialign:</code> . . . . .	440, 507
<code>\l_@@_stop_loop_bool</code> . . . . .	1335, 1336, 1364, 1368, 1395, 1396, 1424, 1429
<code>\c_@@_table_collect_begin_tl</code> .	130, 132, 145
<code>\c_@@_table_print_tl</code> . . . . .	133, 134, 147
<code>\@@_test_if_math_mode:</code> . . . . .	65, 583, 1109, 1117, 1125, 1133, 1141
<code>\l_@@_the_array_box</code> . . . . .	655, 680, 741, 745, 776, 803
<code>\l_@@_tikz_tl</code> . . . . .	2289, 2335
<code>\c_@@_types_of_matrix_seq</code> . . . .	2451, 2458
<code>\@@_update_for_first_and_last_row:</code> . .	348, 371, 602, 983, 1054
<code>\@@_vdots</code> . . . . .	1794, 1818
<code>\@@_vdottedline:n</code> . . . . .	562, 2018
<code>\@@_vdottedline_i:n</code> . . . . .	2022, 2024
<code>\@@_vline:</code> . . . . .	595, 1984
<code>\@@_vline_i:</code> . . . . .	78, 1984, 1985
<code>\l_@@_vlines_bool</code> . . . . .	159, 195, 663, 674, 681, 700, 1285
<code>\l_@@_white_bool</code> . . . . .	2291, 2322
<code>\g_@@_width_first_col_dim</code> 170, 720, 984, 987	
<code>\g_@@_width_last_col_dim</code> 169, 816, 1055, 1058	
<code>\g_@@_x_final_dim</code> . . . . .	1463, 1476, 1495, 1578, 1591, 1597, 1599, 1630, 1638, 1672, 1680, 1700, 1749, 1765, 1937, 1954, 2003, 2014, 2028, 2035
<code>\g_@@_x_initial_dim</code> . . . . .	1461, 1470, 1487, 1578, 1591, 1594, 1597, 1599, 1630, 1638, 1672, 1680, 1701, 1749, 1763, 1765, 1783, 1786, 1935, 1951, 2001, 2012, 2026, 2034
<code>\g_@@_y_final_dim</code> . . . . .	1464, 1477, 1523, 1525, 1527, 1551, 1553, 1632, 1635, 1674, 1677, 1704, 1754, 1773, 1938, 1955, 2004, 2010, 2029, 2039
<code>\g_@@_y_initial_dim</code> . . . . .	1462, 1471, 1523, 1525, 1526, 1551, 1553, 1632, 1637, 1674, 1679, 1705, 1754, 1771, 1773, 1783, 1787, 1936, 1952, 2002, 2009, 2027, 2037
<code>\</code> . . . . .	895, 2396, 2402, 2408, 2490, 2495, 2503, 2509, 2516, 2522, 2528, 2541, 2547, 2552, 2558, 2559, 2592, 2593, 2634, 2635, 2675, 2676, 2690, 2696
<code>\{</code> . . . 97, 1126, 2424, 2502, 2541, 2592, 2701, 2702	
<code>\}</code> . . . 97, 1126, 2424, 2502, 2541, 2592, 2701, 2702	
<code>\ </code> . . . . .	1142, 2423
<code>\sqcup</code> . . . . .	2495, 2499, 2534, 2535, 2546
<b>A</b>	
<code>\array</code> . . . . .	437
<code>\arraycolsep</code> . . . 206, 208, 210, 480, 634, 635, 683, 719, 802, 804, 816, 926, 930, 1023, 1061	
<code>\arrayrulewidth</code> . . . . .	460, 461, 683, 684, 700, 1297, 2325, 2327
<code>\arraystretch</code> . . . . .	479
<code>\AtBeginDocument</code> . . . . .	73, 114
<code>\AutoNiceMatrixWithDelims</code> . . . . .	2385, 2417
<b>B</b>	
<code>\begin</code> . . . . .	690, 691, 726, 759, 1151, 1199, 1297, 1465, 1485, 1492, 1779, 1948, 2007, 2032, 2131, 2142, 2154, 2307, 2390
<code>\Block</code> . . . . .	524, 2280, 2285, 2495
<code>\BNiceArray</code> . . . . .	2724, 2765
<code>\bNiceArray</code> . . . . .	2717, 2756
<code>\BNiceMatrix</code> . . . . .	2377
<code>\bNiceMatrix</code> . . . . .	2374
bool commands:	
<code>\bool_do_until:Nn</code> . . . . .	1336, 1396
<code>\bool_gset_eq:NN</code> . . . . .	482, 483
<code>\bool_gset_false:N</code> . . . . .	569, 579
<code>\bool_gset_true:N</code> . . . . .	910, 1030
<code>\bool_if:NTF</code> . . . . .	43, 122, 331, 413, 429, 451, 455, 459, 477, 486, 527, 567, 584, 593, 596, 632, 681, 689, 694, 700, 704, 722, 811, 815, 941, 965, 1036, 1196, 1225, 1243, 1254, 1258, 1265, 1285, 1296, 1314, 1315, 1360, 1388, 1420, 1449, 1456, 1457, 1483, 1490, 1505, 1512, 1517, 1522, 1524, 1536, 1540, 1545, 1550, 1552, 1564, 1568, 1573, 1581, 1586, 1592, 1596, 1611, 1615, 1620, 1624, 1653, 1657, 1662, 1666, 1712, 1714, 1729, 1759, 1760, 1761, 1806, 1812, 1818, 1824, 1830, 1855, 1933, 1988, 2022, 2058, 2068, 2322, 2339
<code>\bool_if:nTF</code> . . . . .	660, 671, 714, 750, 913, 919, 966, 1037, 1252, 1805, 1811, 1817, 1823, 1829, 1907, 1941, 2299
<code>\bool_new:N</code> . . . . .	18, 34, 63, 64, 71, 72, 102, 109, 112, 113, 152, 155, 156, 158, 159, 160, 161, 163, 164, 165, 166, 175, 2043



<code>\bool_set:Nn</code> .....	1577, 1590
<code>\bool_set_false:N</code> .....	1172, 1186, 1263, 1264, 1335, 1340, 1395, 1400, 1503, 1534, 1562, 1609, 1651, 1875, 1876, 1939, 1940
<code>\bool_set_true:N</code> .....	19, 36, 39, 77, 117, 157, 220, 585, 612, 1099, 1194, 1344, 1350, 1356, 1364, 1368, 1392, 1404, 1410, 1416, 1424, 1429, 1453, 1882, 1890, 1896, 1904, 2005, 2006, 2030, 2031, 2048, 2050
<code>\l_tmpa_bool</code> .....	1172, 1186, 1194, 1196, 1577, 1596
<code>\l_tmpb_bool</code> .....	1590, 1592
box commands:	
<code>\box_clear_new:N</code> .....	469, 655
<code>\box_dp:N</code> .....	343, 353, 497, 506, 607, 822, 836, 849
<code>\box_ht:N</code> .....	344, 355, 361, 499, 501, 504, 605, 821, 836, 849
<code>\box_move_up:nn</code> .....	50, 52, 54, 741, 775
<code>\box_rotate:Nn</code> .....	1919
<code>\box_set_dp:Nn</code> .....	822
<code>\box_set_ht:Nn</code> .....	821
<code>\box_use_drop:N</code> .....	393, 409, 741, 745, 776, 803, 823, 1015, 1093, 2355, 2357
<code>\box_wd:N</code> .....	370, 645, 653, 843, 855, 988, 1059
<code>\l_tmpa_box</code> .....	638, 645, 646, 653, 793, 821, 822, 823, 836, 849
<code>\l_tmpb_box</code> .....	829, 843, 844, 855
<b>C</b>	
<code>\Cdots</code> .....	516
<code>\cdots</code> .....	530, 1793
<code>\coordinate</code> .....	345, 448, 912, 938, 950, 2211, 2214, 2243, 2248, 2316, 2321
<code>\cr</code> .....	956
<code>\crr</code> .....	906
cs commands:	
<code>\cs_generate_variant:Nn</code> .....	395, 1480, 2080, 2081
<code>\cs_gset:Npn</code> .....	1230, 1237, 2073
<code>\cs_gset:Npx</code> .....	1799
<code>\cs_gset_eq:NN</code> .....	135, 490
<code>\cs_if_exist:NTF</code> .....	16, 470, 473, 586, 589, 615, 622, 1173, 1187, 1223, 1325, 1326, 2060, 2103
<code>\cs_if_exist_p:N</code> .....	1943, 1945
<code>\cs_if_free:NTF</code> .....	1177, 1380, 1441, 1501, 1532, 1560, 1607, 1649
<code>\cs_new:Npn</code> .....	93, 441, 904, 1839, 1850, 1920, 1986, 2267, 2282
<code>\cs_new_protected:Npn</code> .....	25, 26, 27, 28, 29, 30, 31, 32, 65, 138, 319, 339, 348, 365, 396, 416, 427, 442, 467, 819, 825, 884, 892, 897, 898, 1145, 1220, 1293, 1323, 1328, 1459, 1481, 1499, 1508, 1530, 1558, 1605, 1647, 1691, 1797, 1833, 1873, 1916, 1917, 1918, 1919, 1922, 1931, 1966, 1979, 1984, 1994, 1999, 2018, 2024, 2082, 2129, 2140, 2152, 2166, 2205, 2235, 2240, 2268, 2295, 2380, 2437, 2446, 2456
<code>\cs_set:Npn</code> .....	434, 479, 484, 1330, 1370, 1431, 1910, 1914, 2237, 2238
<code>\cs_set_eq:NN</code> .....	126, 431, 432, 433, 440, 507, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 529, 530, 531, 532, 533, 534,

535, 542, 543, 544, 546, 1286, 1792, 1793, 1794, 1795, 1796, 1838, 1965, 1985, 2280, 2285	
<code>\cs_set_protected:Npn</code> .....	78, 82, 120, 414, 595, 2136, 2137, 2148, 2149, 2162, 2163, 2164, 2362, 2412, 2414
<b>D</b>	
<code>\Ddots</code> .....	518
<code>\ddots</code> .....	532, 1795
<code>\DeclareOption</code> .....	19, 20
dim commands:	
<code>\dim_abs:n</code> .....	1213
<code>\dim_add:Nn</code> .....	683, 1308, 2201
<code>\dim_compare:nNnTF</code> .....	1212, 1710
<code>\dim_compare_p:nNn</code> .....	916, 1578, 1591
<code>\dim_gadd:Nn</code> .....	738, 772, 1526, 1527, 1763, 1771, 1786, 1787
<code>\dim_gset:Nn</code> .....	343, 344, 352, 354, 360, 369, 497, 499, 501, 504, 506, 604, 606, 634, 635, 984, 1055, 1208, 1210, 1470, 1471, 1476, 1477, 1487, 1495, 1594, 1635, 1677, 1951, 1952, 1954, 1955, 2009, 2010, 2012, 2014, 2034, 2035, 2037, 2039
<code>\dim_gset_eq:NN</code> .....	342, 728, 762, 1523, 1525, 1551, 1553, 1599
<code>\dim_gsub:Nn</code> .....	734, 768
<code>\dim_gzero_new:N</code> .....	496, 498, 500, 502, 503, 505, 594, 1461, 1462, 1463, 1464, 2049
<code>\dim_max:nn</code> .....	353, 355, 361, 370, 605, 607, 986, 1057, 1596, 2081, 2119, 2123
<code>\dim_min:nn</code> .....	1596, 2080, 2110, 2114
<code>\dim_new:N</code> .....	61, 84, 86, 88, 167, 168, 169, 170, 171, 172
<code>\dim_ratio:nn</code> .....	1639, 1681, 1717, 1722, 1733, 1741, 1750, 1755, 1766, 1774
<code>\dim_set:Nn</code> .....	85, 87, 89, 221, 272, 395, 480, 644, 652, 783, 789, 842, 854, 1267, 1268, 1301, 1305, 1629, 1631, 1671, 1673, 1694, 1747, 1752, 2062, 2089, 2096, 2109, 2113, 2118, 2122, 2172, 2185
<code>\dim_set_eq:NN</code> .....	1299, 2087, 2094, 2180, 2195, 2313, 2315, 2318, 2320, 2352
<code>\dim_sub:Nn</code> .....	1310, 1311, 2198
<code>\dim_use:N</code> .....	1317, 1318, 1700, 1701, 1704, 1705, 2075, 2175, 2176, 2188, 2190, 2212, 2213, 2215, 2216, 2245, 2246, 2250, 2251
<code>\dim_zero:N</code> .....	786, 792, 1302, 2056
<code>\dim_zero_new:N</code> .....	630, 631, 827, 828, 1247, 1248, 1249, 1250, 1693, 1935, 1936, 1937, 1938, 2001, 2002, 2003, 2004, 2026, 2027, 2028, 2029, 2086, 2088, 2093, 2095
<code>\c_max_dim</code> .....	2087, 2089, 2094, 2096
<code>\g_tmpa_dim</code> .....	728, 734, 738, 741, 762, 768, 772, 775, 1208, 1213
<code>\l_tmpa_dim</code> .....	783, 786, 799, 821, 1299, 1301, 1308, 1310, 1311, 1318, 1747, 1786, 2313, 2316, 2318, 2321, 2352, 2354
<code>\g_tmpb_dim</code> .....	1210, 1213
<code>\l_tmpb_dim</code> .....	789, 792, 806, 822, 1302, 1305, 1311, 1317, 1752, 1787, 2315, 2316, 2320, 2321
<code>\c_zero_dim</code> .....	375, 376, 916, 995, 996, 1073, 1074, 1710, 2221, 2257
<code>\dots</code> .....	534
<code>\draw</code> .....	1317

<b>E</b>		
else commands:		
\else: .....	67	
\end .....	695, 696, 735, 769, 862, 902, 1162, 1211, 1320, 1478, 1488, 1496, 1789, 1956, 2015, 2040, 2135, 2147, 2161, 2358, 2410	
\endarray .....	868, 881	
\endBNiceArray .....	2726, 2767	
\endbNiceArray .....	2719, 2758	
\endBNiceMatrix .....	2378	
\endbNiceMatrix .....	2375	
\endNiceArrayWithDelims .....	1104, 1112, 1120, 1128, 1136, 1144, 2793, 2802	
\endpNiceArray .....	2712, 2749	
\endpNiceMatrix .....	2366	
\endVNiceArray .....	2740, 2785	
\endvNiceArray .....	2733, 2776	
\endVNiceMatrix .....	2372	
\endvNiceMatrix .....	2369	
\everycr .....	494	
exp commands:		
\exp_after:wN .....	142	
\exp_args:NNV .....	876	
\exp_args:NV .....	549, 864, 873, 878	
\exp_not:N .....	2331	
\exp_not:n .....	2278	
\ExplSyntaxOff .....	1241, 2077	
\ExplSyntaxOn .....	1227, 2070	
<b>F</b>		
fi commands:		
\fi: .....	69	
\fill .....	2324	
\firstline .....	513	
fp commands:		
\fp_to_dim:n .....	1696	
<b>G</b>		
group commands:		
\group_begin: ...	124, 1222, 1295, 1909, 2297	
\group_end: ....	129, 1289, 1321, 1912, 2360	
\group_insert_after:N ....	1916, 1917, 1918	
<b>H</b>		
\halign .....	508	
hbox commands:		
\hbox:n .....	50, 52, 55, 445, 800, 935	
\hbox_overlap_left:n .....	990	
\hbox_overlap_right:n .....	1062	
\hbox_set:Nn ...	638, 646, 793, 829, 844, 2306	
\hbox_set:Nw .....	329, 401, 680, 963, 1034	
\hbox_set_end: ....	368, 407, 701, 982, 1053	
\Hdotsfor .....	522	
\hdotsfor .....	535	
\hdottedline .....	520	
\hfil .....	546	
\hline .....	513, 514	
\hrule .....	460, 461	
\Hspace .....	521	
\hspace .....	1836	
\hss .....	546	
<b>I</b>		
\ialign .....	440, 484, 507	
\Iddots .....	519	
\iddots .....	45, 533, 1796	
if commands:		
\if_mode_math: .....	67	
\ifstandalone .....	589	
int commands:		
\int_add:Nn .....	1338, 1339, 1422, 1423	
\int_compare:nNnTF .....	322, 324, 332, 333, 335, 350, 358, 453, 457, 553, 555, 558, 600, 610, 657, 668, 702, 706, 716, 717, 736, 770, 781, 787, 907, 908, 1155, 1190, 1303, 1341, 1343, 1347, 1349, 1353, 1355, 1401, 1403, 1407, 1409, 1413, 1415, 1627, 1669, 1842, 1879, 1893, 1968, 1971, 1973, 1974, 1981, 1982, 2020, 2349, 2392, 2394, 2396, 2400, 2402, 2404, 2406, 2408, 2460	
\int_compare:nTF .....	278	
\int_compare_p:nNn .....	752, 753, 968, 971, 973, 1039, 1042, 1044, 2301, 2302	
\int_decr:N .....	2309, 2311	
\int_eval:n .....	449, 953, 1157, 1300, 1307, 1309, 1494, 1845, 1921, 2013, 2033, 2038, 2176, 2180, 2191, 2195, 2250, 2275, 2276, 2312, 2314, 2317, 2319, 2470	
\int_gadd:Nn .....	1848	
\int_gdecr:N .....	714, 716	
\int_gincr:N ..	321, 341, 592, 947, 1031, 2055	
\int_gset:Nn .....	327, 539, 568, 940, 1032	
\int_gset_eq:NN ..	560, 709, 713, 715, 1325, 1326	
\int_gzero:N .....	444	
\int_gzero_new:N ....	472, 475, 540, 541, 566	
\int_incr:N .....	1626, 1668	
\int_max:nn .....	328, 1033	
\int_new:N ..	59, 60, 100, 101, 103, 105, 107, 110	
\int_set:Nn .....	104, 106, 108, 111, 293, 617, 624, 749, 757, 1331, 1332, 1333, 1334, 1716, 1720, 1731, 1739, 1757, 1877, 1878, 1881, 1885, 1887, 1889, 1895, 1899, 1901, 1903, 2168, 2169, 2308, 2310, 2382, 2383	
\int_set_eq:NN .....	471, 474	
\int_step_inline:nnn ....	1313, 1780, 1913	
\int_step_variable:nNn .....	2170, 2183	
\int_step_variable:nnNn .....	2084, 2091, 2098, 2100, 2207, 2209	
\int_sub:Nn .....	1362, 1363, 1398, 1399	
\int_use:N ..	346, 382, 383, 384, 389, 390, 422, 423, 449, 562, 615, 618, 727, 731, 732, 761, 765, 766, 912, 938, 952, 1004, 1005, 1011, 1082, 1083, 1084, 1089, 1090, 1174, 1180, 1181, 1182, 1188, 1191, 1203, 1204, 1205, 1230, 1231, 1238, 1282, 1373, 1374, 1383, 1384, 1385, 1391, 1434, 1435, 1444, 1445, 1446, 1452, 1467, 1468, 1469, 1473, 1474, 1475, 1486, 1494, 1690, 1800, 1801, 1845, 1866, 1867, 1943, 1945, 1997, 2060, 2063, 2074, 2104, 2107, 2117, 2202, 2222, 2258, 2273, 2274, 2478, 2484, 2489, 2533, 2534, 2535	
\int_zero:N .....	235, 236, 305, 310, 313, 314, 2709, 2716, 2723, 2730, 2737, 2745, 2746, 2754, 2755, 2763, 2764, 2772, 2773, 2781, 2782, 2790, 2798, 2799	
\int_zero_new:N .....	1245, 1246, 1259, 1260, 1261, 1262, 2387, 2388	



\g_tmpa_int . . . . .	940, 947, 953	\nulldelimiterspace . . . . .	645, 653, 843, 855
\l_tmpa_int . . . . .	749, 752, 753, 757, 766, 1716, 1720, 1731, 1739, 1767, 1775, 1780, 1885, 1886, 1899, 1900, 2308, 2309, 2312, 2489	<b>O</b>	
\l_tmpb_int . . . . .	1757, 1769, 1777, 2310, 2311, 2314	\omit . . . . .	908, 909, 946
iow commands:		\OnlyMainNiceMatrix . . . . .	526, 1965
\iow_now:Nn . . . . .		<b>P</b>	
. . . . .	1227, 1228, 1235, 1241, 2070, 2071, 2077	\path . . . . .	1949, 2354
<b>K</b>		peek commands:	
\kern . . . . .	55	\peek_meaning_ignore_spaces:Ntf . . . . .	862
keys commands:		\pgfpathcircle . . . . .	1782
\keys_define:nn . . . . .	176, 216, 240, 262, 289, 296, 308, 2044, 2287, 2425	\pgfpoint . . . . .	1783
\l_keys_key_str . . . . .	16, 2552, 2557, 2591, 2633	\pgfpointanchor . . . . .	1207, 1209
\keys_set:nn . . . . .	288, 597, 598, 1150, 2057, 2298	\pgfusepath . . . . .	1785
\l_keys_value_tl . . . . .	2511, 2671	\phantom . . . . .	1806, 1812, 1818, 1824, 1830
<b>L</b>		\pNiceArray . . . . .	2710, 2747
\lasthline . . . . .	514	\pNiceMatrix . . . . .	2365
\Ldots . . . . .	515	prg commands:	
\ldots . . . . .	529, 1792	\prg_do_nothing: . . . . .	126, 135, 490
\left . . . . .	641, 649, 796, 832, 847	\prg_replicate:nn . . . . .	942, 943, 1858, 1870, 2395, 2398, 2401, 2407
legacy commands:		\prg_return_false: . . . . .	1184, 1197, 1214
\legacy_if:Ntf . . . . .	224	\prg_return_true: . . . . .	1175, 1215
\line . . . . .	1286, 2502	\prg_set_conditional:Npnn . . . . .	1170
<b>M</b>		\ProcessKeysOptions . . . . .	2436
\makebox . . . . .	409	\ProcessOptions . . . . .	21
math commands:		\ProvideDocumentCommand . . . . .	45
\c_math_toggle_token . . . . .	330, 367, 640, 642, 648, 650, 688, 697, 795, 809, 831, 840, 846, 852, 964, 981, 1035, 1052	\ProvidesExplPackage . . . . .	5
\mathinner . . . . .	47	<b>Q</b>	
msg commands:		quark commands:	
\msg_error:nn . . . . .	17, 25	\q_stop . . . . .	1920, 1927, 1928, 2235, 2242, 2266, 2267, 2380, 2389
\msg_error:nnn . . . . .	26, 1390, 1451	<b>R</b>	
\msg_error:nnnn . . . . .	27, 2304	\relax . . . . .	21, 543, 544
\msg_fatal:nn . . . . .	28	\renewcommand . . . . .	140
\msg_fatal:nnn . . . . .	29	\RenewDocumentEnvironment . . . . .	2364, 2367, 2370, 2373, 2376
\msg_new:nnn . . . . .	10, 30	\RequirePackage . . . . .	1, 3, 4, 22, 23, 24
\msg_new:nnnn . . . . .	31	\right . . . . .	641, 649, 808, 839, 851
\msg_redirect_name:nnn . . . . .	33	\rotate . . . . .	525
\msg_warning:nn . . . . .	44	<b>S</b>	
\multicolumn . . . . .	523, 1838, 1852, 1858, 1870	\scriptstyle . . . . .	331, 965, 1036
\myfiledate . . . . .	7	seq commands:	
\myfileversion . . . . .	8	\seq_clear:N . . . . .	2439
<b>N</b>		\seq_gclear_new:N . . . . .	537, 538, 874, 886
\newcolumntype . . . . .	398, 510, 511, 512, 549	\seq_gpop_left:NN . . . . .	877, 888
\NewDocumentCommand . . . . .	287, 1803, 1809, 1815, 1821, 1827, 1857, 1861, 2265, 2385	\seq_gput_left:Nn . . . . .	229, 1844, 1846
\NewDocumentEnvironment . . . . .	577, 860, 870, 1097, 1105, 1113, 1121, 1129, 1137, 1147, 2053, 2706, 2713, 2720, 2727, 2734, 2741, 2750, 2759, 2768, 2777, 2786, 2794	\seq_gset_split:Nnn . . . . .	876, 887
\NewExpandableDocumentCommand . . . . .	1689	\seq_if_in:NnTF . . . . .	227, 2111, 2120, 2458
\NiceArrayWithDelims . . . . .	1102, 1110, 1118, 1126, 1134, 1142, 2791, 2800	\seq_map_function:NN . . . . .	879, 890
\NiceMatrixLastEnv . . . . .	1689	\seq_map_inline:Nn . . . . .	2440
\NiceMatrixOptions . . . . .	287, 2558	\seq_mapthread_function:NNN . . . . .	2230
\noalign . . . . .	441, 490, 1990	\seq_new:N . . . . .	62
\node . . . . .	379, 999, 1077, 2217, 2253, 2331, 2341, 2357	\seq_put_left:Nn . . . . .	2442
\normalbaselines . . . . .	476	\seq_set_eq:NN . . . . .	2444
		\seq_set_from_clist:Nn . . . . .	2448
		\seq_use:Nnnn . . . . .	2680
		\l_tmpa_seq . . . . .	2439, 2442, 2444
		skip commands:	
		\skip_gset:Nn . . . . .	925, 929, 933
		\skip_horizontal:N . . . . .	934, 948

<code>\skip_horizontal:n</code> . . . . .	556, 684, 686, 687, 698, 699, 700, 719, 720, 802, 804, 816, 856, 858, 1016, 1023, 1061, 1064
<code>\skip_vertical:n</code> . . . . .	799, 806, 1990
<code>\g_tmpa_skip</code> . . . . .	925, 929, 933, 934, 948
<code>\c_zero_skip</code> . . . . .	495
<code>\space</code> . . . . .	96, 97, 2484
str commands:	
<code>\c_backslash_str</code> . . . . .	96
<code>\c_colon_str</code> . . . . .	286
<code>\str_gclear:N</code> . . . . .	1290
<code>\str_gset:Nn</code> . . . . .	581, 1101, 1108, 1116, 1124, 1132, 1140, 1149, 2416
<code>\str_if_empty:NTF</code> . . . . .	386, 580, 613, 1008, 1086, 1100, 1107, 1115, 1123, 1131, 1139, 1233, 2224, 2260
<code>\str_if_eq:nnTF</code> . . . . .	95, 219, 270, 438, 724, 744, 747, 900
<code>\str_new:N</code> . . . . .	90, 91, 153, 162, 285
<code>\str_set:Nn</code> 92, 154, 226, 279, 298, 299, 300, 748	
<code>\str_set_eq:NN</code> . . . . .	230, 286
<code>\l_tmpa_str</code> . . . . .	226, 227, 229, 230
<b>T</b>	
<code>\tabskip</code> . . . . .	495
T <sub>E</sub> X and L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> commands:	
<code>\@acol</code> . . . . .	433
<code>\@acoll</code> . . . . .	431
<code>\@acolr</code> . . . . .	432
<code>\@addtopreamble</code> . . . . .	595
<code>\@array@array</code> . . . . .	435
<code>\@arrayacol</code> . . . . .	431, 432, 433
<code>\@arrayrule</code> . . . . .	595
<code>\@arstrutbox</code> 343, 344, 497, 499, 501, 504, 506	
<code>\@currenvir</code> . . . . .	2701, 2702
<code>\@halignto</code> . . . . .	434
<code>\@ifclassloaded</code> . . . . .	35, 38
<code>\@ifnextchar</code> . . . . .	542
<code>\@ifpackageloaded</code> . . . . .	75, 116
<code>\@mainaux</code> . . . . .	1227, 1228, 1235, 1241, 2070, 2071, 2077
<code>\@temptokena</code> . . . . .	125, 128, 142, 144
<code>\c@iRow</code> . . . . .	332, 335, 341, 346, 350, 358, 383, 389, 422, 449, 453, 457, 471, 472, 539, 553, 555, 706, 709, 715, 716, 732, 907, 968, 973, 1005, 1011, 1039, 1044, 1083, 1089, 1300, 1307, 1325, 1341, 1800, 1845, 1866, 1973, 1974, 1981, 1982, 1997, 2038, 2170, 2273, 2275, 2301, 2534, 2535
<code>\c@jCol</code> . . . . .	321, 322, 328, 333, 384, 390, 423, 444, 474, 475, 558, 560, 562, 713, 714, 1031, 1033, 1084, 1090, 1315, 1326, 1353, 1413, 1494, 1800, 1845, 1848, 1867, 1893, 1971, 2013, 2183, 2202, 2274, 2276, 2302, 2484
<code>\c@MaxMatrixCols</code> . . . . .	1156, 2478
<code>\CT@arc@</code> . . . . .	78, 460, 1296
<code>\CT@everycr</code> . . . . .	488
<code>\CT@row@color</code> . . . . .	490
<code>\NC@find</code> . . . . .	126, 149
<code>\NC@find@W</code> . . . . .	544
<code>\NC@find@w</code> . . . . .	543
<code>\NC@rewrite@S</code> . . . . .	127, 140
<code>\new@ifnextchar</code> . . . . .	542
<code>\pgf@x</code> . . . . .	1208, 1210, 1470, 1476, 1487, 1495, 1951, 1954, 2012, 2014, 2034, 2035, 2114, 2123, 2315, 2320, 2354
<code>\pgf@y</code> . . . . .	728, 734, 762, 768, 1299, 1301, 1308, 1310, 1471, 1477, 1952, 1955, 2009, 2010, 2037, 2039, 2110, 2119, 2313, 2318, 2352
<code>\pgfutil@firstofone</code> . . . . .	83
<code>\tikz@library@external@loaded</code> . . . . .	586, 1223
<code>\tikz@parse@node</code> . . . . .	83
tex commands:	
<code>\tex_mkern:D</code> . . . . .	49, 51, 53, 56
<code>\tex_the:D</code> . . . . .	144
<code>\theiRow</code> . . . . .	470, 1325
<code>\thejCol</code> . . . . .	473, 1326
<code>\tikz</code> . . . . .	345, 372, 447, 911, 937, 949, 992, 1070
<code>\tikzset</code> . . . . .	588, 590, 1224, 1276, 2133, 2145, 2156, 2159
tl commands:	
<code>\tl_const:Nn</code> . . . . .	958, 1026
<code>\tl_count:n</code> . . . . .	278
<code>\tl_gclear:N</code> . . . . .	1288
<code>\tl_gclear_new:N</code> 570, 571, 572, 573, 574, 575	
<code>\tl_gput_left:Nn</code> . . . . .	2270
<code>\tl_gput_right:Nn</code> . . . . .	418, 561, 1863, 1996
<code>\tl_gset:Nn</code> . . . . .	128, 132, 134
<code>\tl_if_empty:nTF</code> . . . . .	291, 303, 311, 872, 894
<code>\tl_item:Nn</code> . . . . .	131, 132, 134
<code>\tl_new:N</code> . . . . .	99, 130, 133, 173
<code>\tl_put_left:Nn</code> . . . . .	658, 666
<code>\tl_put_right:Nn</code> . . . . .	602, 669, 677, 679
<code>\tl_set:Nn</code> . . . . .	131, 174, 656, 1200
<code>\tl_set_rescan:Nnn</code> . . . . .	547, 875
<code>\tl_to_str:n</code> . . . . .	2442
<code>\tl_use:N</code> . . . . .	2552, 2557, 2591, 2633
<code>\g_tmpa_tl</code> . . . . .	128, 131, 134
<code>\l_tmpa_tl</code> . . . . .	131, 132, 656, 658, 666, 669, 677, 679, 864, 873, 877, 878, 888, 889, 1200, 1207, 1209
token commands:	
<code>\token_to_str:N</code> . . . . .	2495, 2502, 2547, 2558
<b>U</b>	
use commands:	
<code>\use:N</code> . . . . .	421, 618, 625, 1191, 2063
<code>\use:n</code> . . . . .	1924, 1965, 2329
<code>\use:nn</code> . . . . .	2285
<code>\usetikzlibrary</code> . . . . .	2
<b>V</b>	
vbox commands:	
<code>\vbox:n</code> . . . . .	55
<code>\vbox_to_ht:nn</code> . . . . .	835, 848
<code>\vcenter</code> . . . . .	641, 649, 797, 833, 2547
<code>\Vdots</code> . . . . .	517
<code>\vdots</code> . . . . .	531, 1794
<code>\vline</code> . . . . .	78, 1985
<code>\VNiceArray</code> . . . . .	2738, 2783
<code>\vNiceArray</code> . . . . .	2731, 2774
<code>\VNiceMatrix</code> . . . . .	2371
<code>\vNiceMatrix</code> . . . . .	2368

# 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 . . . . .	5
3.4	Fine tuning of the dotted lines . . . . .	5
<b>4</b>	<b>The Tikz nodes created by nicematrix</b>	<b>5</b>
<b>5</b>	<b>The code-after</b>	<b>7</b>
<b>6</b>	<b>The environment {NiceArray}</b>	<b>7</b>
<b>7</b>	<b>The exterior rows and columns</b>	<b>8</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>10</b>
<b>10</b>	<b>Block matrices</b>	<b>11</b>
<b>11</b>	<b>Advanced features</b>	<b>12</b>
11.1	The command \rotate . . . . .	12
11.2	The option small . . . . .	12
11.3	The counters iRow and jCol . . . . .	13
11.4	The options hlines, vlines and hvlines . . . . .	13
11.5	The option light-syntax . . . . .	14
11.6	Use of the column type S of siunitx . . . . .	14
<b>12</b>	<b>Technical remarks</b>	<b>15</b>
12.1	Definition of new column types . . . . .	15
12.2	Intersections of dotted lines . . . . .	15
12.3	The names of the Tikz nodes created by nicematrix . . . . .	15
12.4	Diagonal lines . . . . .	16
12.5	The “empty” cells . . . . .	16
12.6	The option exterior-arraycolsep . . . . .	17
12.7	The class option draft . . . . .	17
12.8	A technical problem with the argument of \\. . . . .	17
12.9	Obsolete environments . . . . .	17
<b>13</b>	<b>Examples</b>	<b>18</b>
13.1	Dotted lines . . . . .	18
13.2	Width of the columns . . . . .	20
13.3	How to highlight cells of the matrix . . . . .	20
13.4	Direct use of the Tikz nodes . . . . .	23
<b>14</b>	<b>Implementation</b>	<b>24</b>
<b>15</b>	<b>History</b>	<b>88</b>
	<b>Index</b>	<b>91</b>