

# The package `nicematrix`\*

F. Pantigny  
fpantigny@wanadoo.fr

July 18, 2019

## 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`.

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

- continuous dotted lines<sup>1</sup>;
- a first row and a last column for labels;
- a control of the width of the columns.

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

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`.

```
$A = \begin{pmatrix}
1 & & \cdots & & \cdots & & 1 & \\
0 & & \ddots & & & & & \vdots \\
\vdots & & \ddots & & \ddots & & \vdots & \\
0 & & \cdots & & 0 & & & 1
\end{pmatrix}$
\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 2.3 of `nicematrix`, at the date of 2019/07/18.

<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>{pNiceArrayC}</code>	<code>{pNiceArrayRC}</code>
<code>{pNiceMatrix}</code>		<code>{bNiceArrayC}</code>	<code>{bNiceArrayRC}</code>
<code>{bNiceMatrix}</code>		<code>{BNiceArrayC}</code>	<code>{BNiceArrayRC}</code>
<code>{BNiceMatrix}</code>		<code>{vNiceArrayC}</code>	<code>{vNiceArrayRC}</code>
<code>{vNiceMatrix}</code>		<code>{VNiceArrayC}</code>	<code>{VNiceArrayRC}</code>
<code>{VNiceMatrix}</code>		<code>{NiceArrayCwithDelims}</code>	<code>{NiceArrayRCwithDelims}</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{...}{...}`, `|`, `>{...}`, `<{...}`, `@{...}`, `!{...}` and `*{n}{...}` but the letters `p`, `m` and `b` should not be used. See p. 7 the section relating to `{NiceArray}`.

The environments with `C` at the end of their name, `{pNiceArrayC}`, `{bNiceArrayC}`, `{BNiceArrayC}`, `{vNiceArrayC}` and `{VNiceArrayC}` are similar to the environment `{NiceArray}` (especially the special letters `L`, `C` and `R`) but create an exterior column (on the right of the closing delimiter). See p. 8 the section relating to `{pNiceArrayC}`.

The environments with `RC`, `{pNiceArrayRC}`, `{bNiceArrayRC}`, `{BNiceArrayRC}`, `{vNiceArrayRC}`, `{VNiceArrayRC}` are similar to the environment `{NiceArray}` but create an exterior row (above the main matrix) and an exterior column. See p. 9 the section relating to `{pNiceArrayRC}`.

## 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>2</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>3</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.

```
\begin{bNiceMatrix}
a_1      & \Cdots &      & & a_1 \\
\Vdots   & a_2      & \Cdots & & a_2 \\
          & \Vdots & \Ddots \\
\\
a_1      & a_2      &      & & a_n \\
\end{bNiceMatrix}
```

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

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

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

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

<sup>2</sup>The command `\iddots`, defined in `nicematrix`, is a variant of `\ddots` with dots going forward:  $\cdots$ . If `mathdots` is loaded, the version of `mathdots` is used. It corresponds to the command `\adots` of `unicode-math`.

<sup>3</sup>The precise definition of a “non-empty cell” is given below (cf. p. 12).

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`:

<pre> \begin{bNiceMatrix} 0      &amp; \Cdots &amp; \Cdots &amp; 0      &amp; \\ \Vdots &amp;        &amp;        &amp; \Vdots &amp; \\ \Vdots &amp;        &amp;        &amp; \Vdots &amp; \\ 0      &amp; \Cdots &amp; \Cdots &amp; 0      &amp;  \end{bNiceMatrix} </pre>	$\begin{bmatrix} 0 & \cdots & \cdots & 0 \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix}$
--	--

In the first column of this example, there are two instructions `\Vdots` but only one dotted line is drawn (there is no overlapping graphic objects in the resulting PDF<sup>4</sup>).

However, useless computations are performed by TeX before detecting that both instructions would eventually yield the same dotted line. That's why the package `nicematrix` provides starred versions of `\Ldots`, `\Cdots`, etc.: `\Ldots*`, `\Cdots*`, etc. These versions are simply equivalent to `\hphantom{\ldots}`, `\hphantom{\cdots}`, etc. The user should use these starred versions whenever a classical version has already been used for the same dotted line.

<pre> \begin{bNiceMatrix} 0      &amp; \Cdots &amp; \Cdots* &amp; 0      &amp; \\ \Vdots &amp;        &amp;        &amp; \Vdots &amp; \\ \Vdots* &amp;        &amp;        &amp; \Vdots* &amp; \\ 0      &amp; \Cdots &amp; \Cdots* &amp; 0      &amp;  \end{bNiceMatrix} </pre>	$\begin{bmatrix} 0 & \cdots & \cdots & 0 \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix}$
--	--

In fact, in this example, it would be possible to draw the same matrix without starred commands with the following code:

<pre> \begin{bNiceMatrix} 0      &amp; \Cdots &amp;        &amp; 0      &amp; \\ \Vdots &amp;        &amp;        &amp; \Vdots &amp; \\         &amp;        &amp;        &amp; \Vdots &amp; \\ 0      &amp;        &amp; \Cdots &amp; 0      &amp;  \end{bNiceMatrix} </pre>	$\begin{bmatrix} 0 & \cdots & \cdots & 0 \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix}$
---	--

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>5</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`.

<pre> \begin{bNiceMatrix} 0      &amp; \Cdots &amp; \Hspace*{1cm} &amp; 0      &amp; \\ \Vdots &amp;        &amp;                &amp; \Vdots &amp; \\ 0      &amp; \Cdots &amp;                &amp; 0      &amp;  \end{bNiceMatrix} </pre>	$\begin{bmatrix} 0 & \cdots & \cdots & 0 \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix}$
--	--

<sup>4</sup>And it's not possible to draw a `\Ldots` and a `\Cdots` line between the same cells.

<sup>5</sup>Nevertheless, the best way to fix the width of a column is to use the environment `{NiceArray}` with a column of type `w` (or `W`).

### 3.1 The option `nullify-dots`

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

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

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

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

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

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

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).

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

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.

```

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

```

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

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

```

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

```

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

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` is compatible with the extension `colortbl`.

### 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 `{matrix}`. This option can be set as option of `\usepackage` or with the command `\NiceMatrixOptions`.

In fact, this option is an alias for the conjunction of two options: `renew-dots` and `renew-matrix`.

- The option `renew-dots`

With this option, the commands `\ldots`, `\cdots`, `\vdots`, `\ddots`, `\iddots`<sup>6</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}

```

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

<sup>6</sup>The command `\iddots` is not a command of LaTeX but is defined by the package `nicematrix`. If `mathdots` is loaded, the version of `mathdots` is used.

## 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.

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

In fact, the package `nicematrix` can create “extra nodes”. These new nodes are created if the option `create-extra-nodes` is used. There are two series of extra nodes: the “medium nodes” and the “large nodes”.

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

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

The names of the “large nodes” are constructed by adding the suffix “`-large`” to the names of the “normal nodes”. In the following example, we have underlined the “large nodes”. We consider that this example is self-explanatory.<sup>7</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>8</sup>

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

<sup>7</sup>In the environments like `{pNiceArrayC}` and `{pNiceArrayRC}`, there is not “large nodes” created in the exterior row and column.

<sup>8</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`.

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”. In the following example, we have used `extra-left-margin` and `extra-right-margin` with the value 3 pt.

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

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.

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

We explain below how to fill the nodes created by `nicematrix` (cf. p. 16).

## 5 The code-after

The option `code-after` may be used to give some code that will be excuted after the construction of the matrix (and, 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>9</sup> instead of `l`, `c` and `r`. It's possible to use the constructions `w{...}{...}`, `W{...}{...}`, `l`, `>{...}`, `<{...}`, `@{...}`, `!{...}` and `*{n}{...}` but the letters `p`, `m` and `b` should not be used.<sup>10</sup>

The environment `{NiceArray}` accepts the classical options `t`, `c` and `b` of `{array}` but also other options defined by `nicematrix` (`renew-dots`, `columns-width`, etc.).

An example with a linear system (we need `{NiceArray}` for the vertical rule):

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

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

<sup>9</sup>The column types `L`, `C` and `R` are defined locally inside `{NiceArray}` with `\newcolumnntype` of `array`. This definition overrides an eventual previous definition. In fact, the column types `w` and `W` are also redefined.

<sup>10</sup>In a command `\multicolumn`, one should also use the letters `L`, `C`, `R`.

An example where we use `{NiceArray}` because we want to use the types L and R for the columns:

```

 $\left(\begin{NiceArray}{LCR}
a_{11} & \cdots & a_{1n} \\
a_{21} & & a_{2n} \\
\vdots & & \vdots \\
a_{n-1,1} & \cdots & a_{n-1,n}
\end{NiceArray}\right)$ 

```

## 7 The environment `{pNiceArrayC}` and its variants

The environment `{pNiceArrayC}` composes a matrix with an exterior column.

The environment `{pNiceArrayC}` takes a mandatory argument which is the preamble of the array. The types of columns available are the same as for the environment `{NiceArray}`. **However, no specification must be given for the last column.** It will automatically (and necessarily) be a L column.

A special option, called `code-for-last-col`, specifies tokens that will be inserted before each cell of the last column. The option `columns-width` doesn't apply to this external column.

```

 $\begin{pNiceArrayC}{*6C|C}[nullify-dots,code-for-last-col={\scriptstyle}]
1 & 1 & 1 & \cdots & 1 & 0 & \scriptstyle L_2 \leftarrow L_2 - L_1 \\
0 & 1 & 0 & \cdots & 0 & & \scriptstyle L_3 \leftarrow L_3 - L_1 \\
0 & 0 & 1 & \ddots & \vdots & & \\
& & & \ddots & \vdots & & \\
\vdots & & & \ddots & 0 & & \\
0 & & & \cdots & 0 & 1 & \scriptstyle L_n \leftarrow L_n - L_1
\end{pNiceArrayC}$ 

```

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

Note that an horizontal line drawn with `\hline` does *not* extend in the last column.

```

 $\begin{pNiceArrayC}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33} \\
\hline
S_1 & S_2 & S_3
\end{pNiceArrayC}$ 

```

In fact, the environment `{pNiceArrayC}` and its variants are based upon a more general environment, called `{NiceArrayCwithDelims}`. 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 `{NiceArrayCwithDelims}` if we want to use atypical delimiters.

```

 $\begin{NiceArrayCwithDelims}
{\downarrow}{\downarrow}{CCC}
1 & 2 & 3 & L_1 \\
4 & 5 & 6 & L_2 \\
7 & 8 & 9 & L_3
\end{NiceArrayCwithDelims}$ 

```



## 8 The environment `{pNiceArrayRC}` and its variants

The environment `{pNiceArrayRC}` composes a matrix with an exterior row and an exterior column. This environment `{pNiceArrayRC}` takes a mandatory argument which is the preamble of the array. As for the environment `{pNiceArrayC}`, no specification must be given for the last column (it will automatically be a L column).

A special option, called `code-for-first-row`, specifies tokens that will be inserted before each cell of the first row.

```

 $\begin{pNiceArrayRC}{CCC}% (here, % is mandatory)
  [columns-width = auto,
  code-for-first-row = \color{blue},
  code-for-last-col = \color{blue}]
C_1 & C_2 & C_3 \\
1 & 2 & 3 & L_1 \\
4 & 5 & 6 & L_2 \\
7 & 8 & 9 & L_3 \\
\end{pNiceArrayRC}$ 

```

$$\begin{pmatrix} C_1 & C_2 & C_3 \\ 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \begin{matrix} L_1 \\ L_2 \\ L_3 \end{matrix}$$

The first row of an environment `{pNiceArrayRC}` has the number 0, and not 1. This number is used for the names of the Tikz nodes (the names of these nodes are used, for example, by the command `\line` in `code-after`).

For technical reasons, it's not possible to use the option of the command `\line` after the first row (the placement of the delimiters would be wrong).

If we want to write a linear system, we can use the following code, with a preamble `CCC|C`:

```

 $\begin{pNiceArrayRC}{CCC|C}
C_1 & \cdots & C_n \\
a_{11} & \cdots & a_{1n} & b_1 \\
\vdots & & \vdots & \vdots \\
a_{n1} & \cdots & a_{nn} & b_n \\
\end{pNiceArrayRC}$ 

```

$$\begin{pmatrix} C_1 & \cdots & C_n \\ a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{matrix} b_1 \\ \vdots \\ b_n \end{matrix}$$

We remark that the vertical rule doesn't extend in the first row.<sup>11</sup>

In fact, the environment `{pNiceArrayRC}` and its variants are based upon an more general environment, called `{NiceArrayRCwithDelims}`. 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 `{NiceArrayRCwithDelims}` if we want to use atypical delimiters.

```

 $\begin{NiceArrayRCwithDelims}
  {\downarrow}{\downarrow}{CCC}[columns-width=auto]
C_1 & C_2 & C_3 \\
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{NiceArrayRCwithDelims}$ 

```

$$\begin{matrix} C_1 & C_2 & C_3 \\ \left| \begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix} \right| \end{matrix}$$

## 9 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` (of `arydshln`).

<sup>11</sup>This is a feature of the version 2.2.1 of `nicematrix`. Before that version, the `vrule` extended in the first row. We must remark that, if the extension `arydshln` is loaded, the line will extend even if any functionality of `arydshln` is used in the array (that's because `arydshln` redefine many internals of `array`).

```

\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 “first row” and the “last column” of the environments for the environments with such features (e.g. `{pNiceArrayRC}`).

```

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

```

$$\begin{pmatrix} C_1 & C_2 & C_3 & C_4 \\ 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ \hdottedline 13 & 14 & 15 & 16 \end{pmatrix} \begin{matrix} L_1 \\ L_2 \\ L_3 \\ L_4 \end{matrix}$$

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)

```

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

## 10 The width of the columns

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

```

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

```

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

It's also possible to fix the width of all the columns of a matrix directly with the option `columns-width` (in all the environments of `nicematrix`).

```

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

```

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

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 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. **Two or three compilations may be necessary.**

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

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

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>12</sup>

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

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

## 11 The option `hlines`

Of course, you can add horizontal rules between rows in the environments of `nicematrix` with the command `\hline`. But you can also specify that all horizontal lines must be drawn with the option `hlines`.

```
\begin{NiceArray}{|*{4}{C|}}[hlines]
e & a & b & c \\
a & e & c & b \\
b & c & e & a \\
c & b & a & e
\end{NiceArray}
```

<i>e</i>	<i>a</i>	<i>b</i>	<i>c</i>
<i>a</i>	<i>e</i>	<i>c</i>	<i>b</i>
<i>b</i>	<i>c</i>	<i>e</i>	<i>a</i>
<i>c</i>	<i>b</i>	<i>a</i>	<i>e</i>

## 12 Utilisation 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`.

<sup>12</sup>At this time, this is the only usage of the environment `{NiceMatrixBlock}` but it may have other usages in the future.

```

 $\begin{pNiceArrayRC}{SCwc{1cm}C}[nullify-dots]
{C_1} & \Cdots & & C_n \\
2.3 & 0 & \Cdots & 0 \\
12.4 & \Vdots & & \Vdots \\
1.45 & \\
7.2 & 0 & \Cdots & 0 \\
\end{pNiceArrayRC}$ 

```

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

## 13 Technical remarks

### 13.1 Diagonal lines

By default, all the diagonal lines<sup>13</sup> of a same array are “parallelized”. That means that the first diagonal line is drawn and, then, the other lines are drawn parallel to the first one (by rotation around the left-most extremity of the line). That’s why the position of the instructions `\Ddots` in the array can have a marked effect on the final result.

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

Example with parallelization (default):

```

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

```

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

```

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

```

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

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

The same example without parallelization:

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

### 13.2 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:

<sup>13</sup>We speak of the lines created by `\Ddots` and not the lines created by a command `\line` in `code-after`.

```

\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` and their starred versions 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`.

### 13.3 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>14</sup>

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

However, the user can change this behaviour with the boolean option `exterior-arraycolsep` of the command `\NiceMatrixOptions`. With this option, `{NiceArray}` will insert the same horizontal spaces as the environment `{array}`.

This option is only for “compatibility” since the package `nicematrix` provides a more precise control with the options `left-margin`, `right-margin`, `extra-left-margin` and `extra-right-margin`.

### 13.4 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>15</sup>

That’s why, when the class option `draft` is used, the dotted lines are not drawn, for a faster compilation.

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

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

```

\begin{pNiceMatrix}
a & \frac{A}{B} \\[2mm]
b & c
\end{pNiceMatrix}

```

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

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

<sup>14</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@{}}`.

<sup>15</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.

```

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

```

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

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

```

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

```

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

## 14 Examples

### 14.1 Dotted lines

A tridiagonal matrix:

```

$\begin{pNiceMatrix}[nullify-dots]
a & b & 0 & & \cdots & 0 & \\
b & a & b & \ddots & & \vdots & \\
0 & b & a & \ddots & & & \\
& \ddots & \ddots & \ddots & & 0 & \\
\vdots & & & & & b & \\
0 & \cdots & & 0 & b & a & \\
\end{pNiceMatrix}$

```

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

A permutation matrix:

```

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

```

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

An example with `\Iddots`:

```

$\begin{pNiceMatrix}
1 & \cdots & & 1 & \\
\vdots & & & 0 & \\
& \Iddots & \Iddots & \vdots & \\
1 & 0 & \cdots & 0 & \\
\end{pNiceMatrix}$

```

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

An example with `\multicolumn`:

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

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

An example with `\Hdotsfor`:

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

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

An example for the resultant of two polynoms:

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

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

## 14.2 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 $\begin{pNiceArrayC}{CCCC:C}
1&1&1&1&1&\backslash\backslash
2&4&8&16&9&\backslash\backslash
3&9&27&81&36&\backslash\backslash
4&16&64&256&100&\backslash\backslash
\end{pNiceArrayC}$
...
\end{NiceMatrixBlock}
```

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

## 14.3 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`.

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



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

The package `nicematrix` is constructed upon the environment `{array}` and, therefore, it's possible to use the package `colortbl` in the environments of `nicematrix`.

```
\begin{bNiceMatrix}
0 & \Cdots & 0 \\
\rowcolor{red!15} 1 & \Cdots & 1 \\
0 & \Cdots & 0 \\
\end{bNiceMatrix}
```

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

The result may be disappointing. We therefore 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`. Warning: some PDF readers are not able to render transparency correctly.

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

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

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

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

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

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

```
\begin{pNiceArrayC}{CCC}[name=example,create-extra-nodes]
a & a + b & a + b + c & L_1 \\
a & a & a + b & L_2 \\
a & a & a & L_3 \\
\end{pNiceArrayC}
```

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

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

```
\tikzset{myoptions/.style={remember picture,
    overlay,
    name prefix = example-,
    every node/.style = {fill = red!15,
        blend mode = multiply,
        inner sep = 0pt}}

\begin{tikzpicture}[myoptions]
\node [fit = (1-1) (1-3)] {} ;
\node [fit = (2-1) (2-3)] {} ;
\node [fit = (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}[myoptions, name suffix = -medium]
\node [fit = (1-1) (1-3)] {} ;
\node [fit = (2-1) (2-3)] {} ;
\node [fit = (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.

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

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

## 14.4 Block matrices

In the following example, we use the “large nodes” to construct a block matrix (the dashed lines have been drawn with `arydshln`).

```
\NiceMatrixOptions{letter-for-dotted-lines = V}
\left(\begin{NiceArray}{CC:CC}%
  [create-extra-nodes,
   code-after = { \tikz \node [fit = (1-1-large) (2-2-large), inner sep = 0 pt]
                  {$0_{22}$} ; } ]
    & & a_{13} & a_{14} \\
    & & a_{23} & a_{24} \\
\hdashline
a_{31} & a_{32} & a_{33} & a_{34} \\
a_{41} & a_{42} & a_{43} & a_{44}
\end{NiceArray}\right)
```

$$D = \begin{pmatrix} & & & & a_{13} & a_{14} \\ & 0_{22} & & & a_{23} & a_{24} \\ & & & & a_{33} & a_{34} \\ a_{31} & a_{32} & & & a_{33} & a_{34} \\ a_{41} & a_{42} & & & a_{34} & a_{44} \end{pmatrix}$$

## 15 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 independant of its implementation. Unfortunately, it was not possible to be strictly independant: the package `nicematrix` relies upon the fact that the package `{array}` uses `\ialign` to begin the `\halign`.

The desire to do no modification to existing code leads to complications in the code of this extension.

### 15.1 Declaration of the package and extensions loaded

First, `tikz` and the Tikz library `fit` are loaded before the `\ProvidesExplPackage`. They are loaded this way because `\usetikzlibrary` in `expl3` code fails.<sup>16</sup>

```
1 \RequirePackage{tikz}
2 \usetikzlibrary{fit}
3 \RequirePackage{expl3}[2019/02/15]
```

<sup>16</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 give the traditionnal declaration of a package written with `expl3`:

```

4 \RequirePackage{l3keys2e}
5 \ProvidesExplPackage
6   {nicematrix}
7   {\myfiledate}
8   {\myfileversion}
9   {Several features to improve the typesetting of mathematical matrices with TikZ}

```

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

```

10 \bool_new:N \c_@@_draft_bool
11 \DeclareOption { draft } { \bool_set_true:N \c_@@_draft_bool }
12 \DeclareOption* { }
13 \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`.

```

14 \RequirePackage { array }
15 \RequirePackage { amsmath }
16 \RequirePackage { xparse } [ 2018-10-17 ]

17 \cs_new_protected:Nn \@@_error:n { \msg_error:nn { nicematrix } { #1 } }
18 \cs_new_protected:Nn \@@_error:nn { \msg_error:nn { nicematrix } { #1 } { #2 } }
19 \cs_new_protected:Npn \@@_msg_new:nn { \msg_new:nnn { nicematrix } }
20 \cs_new_protected:Npn \@@_msg_new:nnn { \msg_new:nnnn { nicematrix } }
21 \cs_new_protected:Npn \@@_msg_redirect_name:nn
22   { \msg_redirect_name:nnn { nicematrix } }

```

## 15.2 Technical definitions

We test whether the current class is `revtex4-1` or `revtex4-2`.

```

23 \bool_new:N \c_@@_revtex_bool
24 \@ifclassloaded { revtex4-1 }
25   { \bool_set_true:N \c_@@_revtex_bool }
26   { }
27 \@ifclassloaded { revtex4-2 }
28   { \bool_set_true:N \c_@@_revtex_bool }
29   { }

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

32 \bool_if:NT \c_@@_draft_bool
33   { \msg_warning:nn { nicematrix } { Draft-mode } }

```

We create booleans in order to know if some packages are loaded. for the package `siunitx`, the boolean is called `\c_@@_siunitx_loaded_bool`.<sup>17</sup>

```

34 \AtBeginDocument
35   {
36     \clist_map_inline:nn
37       {
38         siunitx
39       }
40       {
41         \bool_new:c { c_@@_#1_loaded_bool }

```

<sup>17</sup>It's not possible to use `\@ifpackageloaded` in the core of the functions because `\@ifpackageloaded` is available only in the preamble.

```

42     \ifpackageloaded { #1 }
43     { \bool_set_true:c { c_@@_#1_loaded_bool } }
44     { }
45 }
46 }

```

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.

```

47 \ProvideDocumentCommand \iddots { }
48 {
49   \mathinner
50   {
51     \mkern 1 mu
52     \raise \p@ \hbox:n { . }
53     \mkern 2 mu
54     \raise 4 \p@ \hbox:n { . }
55     \mkern 2 mu
56     \raise 7 \p@ \vbox { \kern 7 pt \hbox:n { . } } \mkern 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

```

The dimension `\l_@@_columns_width_dim` will be used when the options specify that all the columns must have the same width.

```

60 \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`.

```

61 \seq_new:N \g_@@_names_seq

```

The integer `\l_@@_nb_first_row_int` is the number of the first row of the array. The default value is 1, but, in the environments like `{pNiceArrayRC}`, the value will be 0.

```

62 \int_new:N \l_@@_nb_first_row_int
63 \int_set:Nn \l_@@_nb_first_row_int 1

```

The flag `\l_@@_exterior_column_bool` will indicate if we are in an environment of the type of `{pNiceArrayC}` or `{pNiceArrayRC}`. It will be used for the creation of the “large nodes”.

```

64 \bool_new:N \l_@@_exterior_column_bool

```

Consider the following code :

```

\begin{pNiceArrayC}{CC}
1 & 2
3 & 4
\end{pNiceArrayC}

```

In such a code, the last column of the environment `{pNiceArrayC}` is not used. We want to be able to detect such a situation. We create a boolean for that job.

```

65 \bool_new:N \g_@@_exterior_column_found_bool

```

This boolean will be raised in the last column of environments like `{pNiceArrayC}`.

We want to know if we are in an environment `{NiceArray}` because we want to raise an error if the user tries to use nested environments `{NiceArray}`.

```

66 \bool_new:N \l_@@_in_NiceArray_bool

```

### 15.3 The column S of siunitx

The command `\NC@rewrite@S` is a LaTeX command created by siunitx in connection with the S column. In the code of siunitx, this command is defined by:

```
\renewcommand*{\NC@rewrite@S}[1] []
{
  \@temptokena \exp_after:wN
  {
    \tex_the:D \@temptokena
    > { \__siunitx_table_collect_begin: S {#1} }
    c
    < { \__siunitx_table_print: }
  }
  \NC@find
}
```

We want to patch this command (in the environments of nicematrix) in order to have:

```
\renewcommand*{\NC@rewrite@S}[1] []
{
  \@temptokena \exp_after:wN
  {
    \tex_the:D \@temptokena
    > { \@@_Cell: \__siunitx_table_collect_begin: S {#1} }
    c
    < { \__siunitx_table_print: \@@_end_Cell: }
  }
  \NC@find
}
```

However, we don't want to use explicitly any private command of siunitx. That's why we will extract the name of the two `\__siunitx...` commands by their position in the code of `\NC@rewrite@S`. Since the command `\NC@rewrite@S` appends some tokens to the *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 utilisation of an environment of nicematrix with the command `\@@_adapt_S_column:`. This command becomes globally no-op when used once.

```
67 \cs_set_protected:Npn \@@_adapt_S_column:
68 {
69   \bool_if:NT \c_@@_siunitx_loaded_bool
70   {
71     \group_begin:
72     \@temptokena = { }
```

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

```
73     \cs_set_eq:NN \NC@find \prg_do_nothing:
74     \NC@rewrite@S { }
```

Conversion of the *toks* `\@temptokena` in a token list of `expl3` (the *toks* are not supported by `expl3` and that's why we use `\tex_the:D`). The conversion is global (`gset`) because we have to exit the group.

```
75     \exp_args:NNo \tl_gset:Nn \g_tmpa_tl { \tex_the:D \@temptokena }
76     \group_end:
77     \tl_new:N \c_@@_table_collect_begin_tl
78     \tl_set:Nx \l_tmpa_tl { \tl_item:Nn \g_tmpa_tl 2 }
79     \tl_gset:Nx \c_@@_table_collect_begin_tl { \tl_item:Nn \l_tmpa_tl 1 }
80     \tl_new:N \c_@@_table_print_tl
81     \tl_gset:Nx \c_@@_table_print_tl { \tl_item:Nn \g_tmpa_tl { -1 } }
82   }
```

When, used once, the command `\@@_adapt_S_column:` becomes no-op (globally).

```
83     \cs_gset_eq:NN \@@_adapt_S_column: \prg_do_nothing:
84   }
```

The token lists `\c_@@_table_collect_begin_tl` and `\c_@@_table_print_tl` contain now the two commands of `siunitx`.

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).

```

85 \cs_new_protected:Npn \@@_renew_NC@rewrite@S:
86 {
87   \renewcommand*{\NC@rewrite@S}[1] []
88   {
89     \@temptokena \exp_after:wN
90     {
91       \tex_the:D \@temptokena
92       > { \@@_Cell: \c_@@_table_collect_begin_tl S {##1} }
93       c
94       < { \c_@@_table_print_tl \@@_end_Cell: }
95     }
96     \NC@find
97   }
98 }

```

## 15.4 The options

```

99 \@@_msg_new:nn { Option-Transparent~suppressed }
100 {
101   The~option~'Transparent'~has~been~renamed~'transparent'.\\
102   However,~you~can~go~on~for~this~time.
103 }
104 \@@_msg_new:nn { Option-RenewMatrix~suppressed }
105 {
106   The~option~'RenewMatrix'~has~been~renamed~'renew-matrix'.\\
107   However,~you~can~go~on~for~this~time.
108 }

```

The token list `\l_@@_pos_env_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}`, `{pNiceArrayC}`, `{pNiceArrayRC}` 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.

```

109 \str_new:N \l_@@_pos_env_str
110 \str_set:Nn \l_@@_pos_env_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}` (but neither for `{NiceMatrix}`, `{pNiceArrayC}`, `{pNiceArrayRC}` and their variants even if these environments rely upon `{NiceArray}`).

```

111 \bool_new:N \l_@@_exterior_arraycolsep_bool

```

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

```

112 \bool_new:N \l_@@_parallelize_diags_bool
113 \bool_set_true:N \l_@@_parallelize_diags_bool

```

The flag `\l_@@_hlines_bool` correspond to the option `\hlines`.

```

114 \bool_new:N \l_@@_hlines_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.).

```
115 \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 cell of the “exterior column” of an environment of the kind of `{pNiceArrayC}`).

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

The token list `\l_@@_code_for_last_col_tl` will contain code inserted at the beginning of each cell of the last column in the environment `{pNiceArrayC}` (and its variants). It corresponds to the option `code-for-last-col`.

```
117 \tl_new:N \l_@@_code_for_last_col_tl
```

We don’t want to patch any existing code. That’s why some code must be executed in a `\group_insert_after:N`. That’s why the parameters used in that code must be transferred outside the current group. To do this, we copy those quantities in global variables just before the `\group_insert_after:N`. Therefore, for those quantities, we have two parameters, one local and one global. For example, we have `\l_@@_name_str` and `\g_@@_name_str`.

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.

```
118 \str_new:N \g_@@_name_str
```

```
119 \str_new:N \l_@@_name_str
```

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

```
120 \bool_new:N \l_@@_extra_nodes_bool
```

```
121 \bool_new:N \g_@@_extra_nodes_bool
```

The dimensions `\l_@@_left_margin_dim` and `\l_@@_right_margin_dim` correspond to the options `left-margin` and `right-margin`.

```
122 \dim_new:N \l_@@_left_margin_dim
```

```
123 \dim_new:N \l_@@_right_margin_dim
```

```
124 \dim_new:N \g_@@_left_margin_dim
```

```
125 \dim_new:N \g_@@_right_margin_dim
```

The dimensions `\l_@@_extra_left_margin_dim` and `\l_@@_extra_right_margin_dim` correspond to the options `extra-left-margin` and `extra-right-margin`.

```
126 \dim_new:N \l_@@_extra_left_margin_dim
```

```
127 \dim_new:N \l_@@_extra_right_margin_dim
```

```
128 \dim_new:N \g_@@_extra_right_margin_dim
```

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

```
129 \keys_define:nn { NiceMatrix / Global }
```

```
130 {
```

```
131   hlines .bool_set:N = \l_@@_hlines_bool ,
```

```
132   parallelize-diags .bool_set:N = \l_@@_parallelize_diags_bool ,
```

```
133   parallelize-diags .default:n = true ,
```

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

```
134   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
```

```
135   renew-dots .default:n = true ,
```

```
136   nullify-dots .bool_set:N = \l_@@_nullify_dots_bool ,
```

```
137   nullify-dots .default:n = true ,
```



An option to test whether the extra nodes will be created (these nodes are the “medium nodes” and “large nodes”). In some circumstances, the extra nodes are created automatically, for example when a dotted line has an “open” extremity.

```

138   create-extra-nodes .bool_set:N = \l_@@_extra_nodes_bool ,
139   create-extra-nodes .default:n = true,
140   left-margin .dim_set:N = \l_@@_left_margin_dim ,
141   left-margin .default:n = \arraycolsep ,
142   right-margin .dim_set:N = \l_@@_right_margin_dim ,
143   right-margin .default:n = \arraycolsep ,
144   extra-left-margin .dim_set:N = \l_@@_extra_left_margin_dim ,
145   extra-right-margin .dim_set:N = \l_@@_extra_right_margin_dim
146 }

```

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

```

147 \keys_define:nn { NiceMatrix / Env }
148 {
149   columns-width .code:n =
150     \str_if_eq:nnTF { #1 } { auto }
151     { \bool_set_true:N \l_@@_auto_columns_width_bool }
152     { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,
153   columns-width .value_required:n = true ,
154   name .code:n =
155     \seq_if_in:NnTF \g_@@_names_seq { #1 }
156     { \@@_error:nn { Duplicate~name } { #1 } }
157     { \seq_gput_left:Nn \g_@@_names_seq { #1 } }
158     \str_set:Nn \l_@@_name_str { #1 } ,
159   name .value_required:n = true ,
160   code-after .tl_gset:N = \g_@@_code_after_tl ,
161   code-after .initial:n = \c_empty_tl ,
162   code-after .value_required:n = true ,
163 }

```

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

```

164 \keys_define:nn { NiceMatrix }
165 {
166   NiceMatrixOptions .inherit:n = NiceMatrix / Global ,
167   NiceMatrix .inherit:n =
168     {
169       NiceMatrix / Global ,
170       NiceMatrix / Env
171     } ,
172   NiceArray .inherit:n =
173     {
174       NiceMatrix / Global ,
175       NiceMatrix / Env
176     } ,
177   NiceArrayC .inherit:n =
178     {
179       NiceMatrix / Global ,
180       NiceMatrix / Env
181     } ,
182   NiceArrayRC .inherit:n =
183     {
184       NiceMatrix / Global ,
185       NiceMatrix / Env
186     }
187 }

```

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

```

188 \keys_define:nn { NiceMatrix / NiceMatrixOptions }

```

189 {

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

```
190   renew-matrix .code:n = \@@_renew_matrix: ,
191   renew-matrix .value_forbidden:n = true ,
192   RenewMatrix .code:n = \@@_error:n { Option~RenewMatrix~suppressed }
193                   \@@_renew_matrix: ,
194   transparent .meta:n = { renew-dots , renew-matrix } ,
195   transparent .value_forbidden:n = true,
196   Transparent .code:n = \@@_error:n { Option~Transparent~suppressed }
197                   \@@_renew_matrix:
198                   \bool_set_true:N \l_@@_renew_dots_bool ,
```

The following option is only for the environment `{pNiceArrayC}` and its variants. It will contain code inserted at the beginning of each cell of the last column.<sup>18</sup>

```
199   code-for-last-col .tl_set:N = \l_@@_code_for_last_col_tl ,
200   code-for-last-col .value_required:n = true ,
```

Idem for the first row in environments like `{pNiceArrayRC}`.

```
201   code-for-first-row .tl_set:N = \l_@@_code_for_first_row_tl ,
202   code-for-first-row .value_required: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}`.

```
203   exterior-arraycolsep .bool_set:N = \l_@@_exterior_arraycolsep_bool ,
204   exterior-arraycolsep .default:n = true ,
```

If the option `columns-width` is used, all the columns will have the same width.

In `\NiceMatrixOptions`, the special value `auto` is not available.

```
205   columns-width .code:n =
206       \str_if_eq:nnTF { #1 } { auto }
207       { \@@_error:n { Option~auto~for~columns~width } }
208       { \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.

```
209   allow-duplicate-names .code:n =
210       \@@_msg_redirect_name:nn { Duplicate~name } { none } ,
211   allow-duplicate-names .value_forbidden:n = true ,
```

By default, the specifier used in the preamble of the array (for example in `{pNiceArrayC}` 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`).

```
212   letter-for-dotted-lines .tl_set:N = \l_@@_letter_for_dotted_lines_str ,
213   letter-for-dotted-lines .value_required:n = true ,
214   letter-for-dotted-lines .initial:n = \c_colon_str ,
```

```
215   unknown .code:n = \@@_error:n { Unknown~key~for~NiceMatrixOptions } }
```

```
216 \@@_msg_new:nnn { Unknown~key~for~NiceMatrixOptions }
217 {
218   The~key~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~command~
219   \token_to_str:N \NiceMatrixOptions. \\
220   If~you~go~on,~it~will~be~ignored. \\
221   For~a~list~of~the~available~keys,~type~H~<return>.
222 }
```

---

<sup>18</sup>In an environment `{pNiceArrayC}`, the last column is composed outside the parentheses of the array.

```

223 {
224   The~available~keys~are~(in~alphabetic~order):~
225   allow~duplicate~names,~
226   code~for~last~col,~
227   exterior~arraycolsep,~
228   hlines,~
229   left~margin,~
230   letter~for~dotted~lines,~
231   nullify~dots,~
232   parallelize~diags,~
233   renew~dots,~
234   renew~matrix,~
235   right~margin,~
236   and~transparent
237 }
238 \@@_msg_new:nn { Option~auto~for~columns~width }
239 {
240   You~can't~give~the~value~'auto'~to~the~option~'columns~width'~here.~
241   If~you~go~on,~the~option~will~be~ignored.
242 }

```

`\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.

```

243 \NewDocumentCommand \NiceMatrixOptions { m }
244 { \keys_set:nn { NiceMatrix / NiceMatrixOptions } { #1 } }

```

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

```

245 \keys_define:nn { NiceMatrix / NiceMatrix }
246 { unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix } }
247 \@@_msg_new:nnn { Unknown~option~for~NiceMatrix }
248 {
249   The~option~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~environment~
250   \{NiceMatrix\}~and~its~variants. \\\
251   If~you~go~on,~it~will~be~ignored. \\\
252   For~a~list~of~the~available~options,~type~H~<return>.
253 }
254 {
255   The~available~options~are~(in~alphabetic~order):~
256   code~after,~
257   columns~width,~
258   create~extra~nodes,~
259   extra~left~margin,~
260   extra~right~margin,~
261   hlines,~
262   left~margin,~
263   name,~
264   nullify~dots,~
265   parallelize~diags,~
266   renew~dots~
267   and~right~margin.
268 }
269 \@@_msg_new:nnn { Duplicate~name }
270 {
271   The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
272   the~same~environment~name~twice.~You~can~go~on,~but,~
273   maybe,~you~will~have~incorrect~results~especially~
274   if~you~use~'columns~width=auto'. \\\
275   For~a~list~of~the~names~already~used,~type~H~<return>. \\\
276   If~you~don't~want~to~see~this~message~again,~use~the~option~

```

```

277   'allow-duplicate-names'.
278 }
279 {
280   The~names~already~defined~in~this~document~are:~
281   \seq_use:Nnnn \g_@@_names_seq { ,~ } { ,~ } { ~and~ }.
282 }

```

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

```

283 \keys_define:nn { NiceMatrix / NiceArray }
284 {

```

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

```

285   c .code:n = \str_set:Nn \l_@@_pos_env_str c ,
286   t .code:n = \str_set:Nn \l_@@_pos_env_str t ,
287   b .code:n = \str_set:Nn \l_@@_pos_env_str b ,
288   unknown .code:n = \@@_error:n { Unknown~option~for~NiceArray }
289 }
290 \@@_msg_new:nnn { Unknown~option~for~NiceArray }
291 {
292   The~option~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~environment~
293   \{NiceArray\}. \\
294   If~you~go~on,~it~will~be~ignored. \\
295   For~a~list~of~the~available~options,~type~H~<return>.
296 }
297 {
298   The~available~options~are~(in~alphabetic~order):~
299   b,~
300   c,~
301   code-after,~
302   create-extra-nodes,~
303   columns-width,~
304   extra-left-margin,~
305   extra-right-margin,~
306   hlines,~
307   left-margin,~
308   name,~
309   nullify-dots,~
310   parallelize-diags,~
311   renew-dots,~
312   right-margin,~
313   and~t.
314 }

```

## 15.5 The environments {NiceArray} and {NiceMatrix}

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}).

```

315 \cs_new_protected:Nn \@@_Cell:
316 {

```

We increment \g\_@@\_column\_int, which is the counter of the columns.

```

317   \int_gincr:N \g_@@_column_int

```

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.

```

318   \int_compare:nNnT \g_@@_column_int = 1 { \int_gincr:N \g_@@_row_int }
319   \int_gset:Nn \g_@@_column_total_int

```

```

320     { \int_max:nn \g_@@_column_total_int \g_@@_column_int }
321     \hbox_set:Nw \l_tmpa_box
322     \c_math_toggle_token
323     \int_compare:nNnT \g_@@_row_int = \c_zero_int
324     \l_@@_code_for_first_row_tl
325 }
326 \cs_new_protected:Nn \@@_end_Cell:
327 {
328     \c_math_toggle_token
329     \hbox_set_end:

```

We want to compute in `\l_@@_max_cell_width_dim` the width of the widest cell of the array (except the cells of the last column of an environment of the kind of `{pNiceArrayC}`).

```

330     \dim_gset:Nn \g_@@_max_cell_width_dim
331     { \dim_max:nn \g_@@_max_cell_width_dim { \box_wd:N \l_tmpa_box } }
332     \int_compare:nNnT \g_@@_row_int = \c_zero_int
333     {
334         \dim_gset:Nn \g_@@_max_dp_row_zero_dim
335         { \dim_max:nn \g_@@_max_dp_row_zero_dim { \box_dp:N \l_tmpa_box } }
336         \dim_gset:Nn \g_@@_max_ht_row_zero_dim
337         { \dim_max:nn \g_@@_max_ht_row_zero_dim { \box_ht:N \l_tmpa_box } }
338     }
339     \int_compare:nNnT \g_@@_row_int = \c_one_int
340     {
341         \dim_gset:Nn \g_@@_max_ht_row_one_dim
342         { \dim_max:nn \g_@@_max_ht_row_one_dim { \box_ht:N \l_tmpa_box } }
343     }

```

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

```

344     \tikz
345     [
346         remember~picture ,
347         inner~sep = \c_zero_dim ,
348         minimum~width = \c_zero_dim ,
349         baseline
350     ]
351     \node
352     [
353         anchor = base ,
354         name = nm - \int_use:N \g_@@_env_int -
355                 \int_use:N \g_@@_row_int -
356                 \int_use:N \g_@@_column_int ,
357         alias =
358             \str_if_empty:NF \l_@@_name_str
359             {
360                 \l_@@_name_str -
361                 \int_use:N \g_@@_row_int -
362                 \int_use:N \g_@@_column_int
363             }
364     ]
365     \bgroup
366     \box_use:N \l_tmpa_box
367     \egroup ;
368 }

```

The environment `{NiceArray}` is the main environment of the extension `nicematrix`. In order to clarify the explanations, we will first give the definition of the environment `{NiceMatrix}`.

Our environment `{NiceMatrix}` must have the same second part as the environment `{matrix}` of `amsmath` (because of the programming of the option `renew-matrix`). Hence, this second part is the following:

```

\endarray
\skip_horizontal:n {-\arraycolsep}

```

That's why, in the definition of `{NiceMatrix}`, we must use `\NiceArray` and not `\begin{NiceArray}` (and, in the definition of `{NiceArray}`, we will have to use `\array`, and not `\begin{array}`: see below).

Here's the definition of `{NiceMatrix}`:

```

369 \NewDocumentEnvironment { NiceMatrix } { ! O { } }
370 {
371   \keys_set:nn { NiceMatrix / NiceMatrix } { #1 }
372   \str_set:Nn \l_@@_pos_env_str c
373   \bool_set_false:N \l_@@_exterior_arraycolsep_bool
374   \NiceArray { * \c@MaxMatrixCols C }
375 }
376 {
377   \endarray
378   \skip_horizontal:n
379     { \g_@@_right_margin_dim + \g_@@_extra_right_margin_dim - \arraycolsep }
380 }

```

For the definition of `{NiceArray}` (just below), we have the following constraints:

- we must use `\array` in the first part of `{NiceArray}` and, therefore, `\endarray` in the second part;
- we have to put a `\group_insert_after:N \@@_after_array:` in the first part of `{NiceArray}` so that `\@@_draw_lines` will be executed at the end of the current environment (either `{NiceArray}` or `{NiceMatrix}`).

```

381 \cs_generate_variant:Nn \dim_set:Nn { N x }

382 \@@_msg_new:nn { Yet-in-NiceArray }
383 {
384   Environments~\{NiceArray\}~(or~\{NiceMatrix\},~etc.)~can't~be~
385   nested.~You~can~go~on,~but,~maybe,~you~will~have~errors~or~an~incorrect~
386   result.
387 }
388 \@@_msg_new:nn { Outside-math-mode }
389 {
390   The~environment~\{@@_currenvir\}~can~be~used~only~in~math~mode~
391   (and~not~in~\token_to_str:N \vcenter).\_
392   If~you~go~on,~you~will~have~other~errors.
393 }

```

In the environment `{NiceArray}`, we will have to redefine the column types `w` and `W`. These definitions are rather long because we have to construct the `w`-nodes in these columns. The redefinition of these two column types are very close and that's why we use a macro `\@@_renewcolumnstype: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.

```

394 \cs_new_protected:Nn \@@_renewcolumnstype:nn
395 {
396   \newcolumnstype #1 [ 2 ]
397   {
398     > {
399       \hbox_set:Nw \l_tmpa_box
400       \@@_Cell:
401     }
402     c
403     < {

```

```

404 \@@_end_Cell:
405 \hbox_set_end:
406 #2
407 \hbox_set:Nn \l_tmpb_box
408 { \makebox [ ##2 ] [ ##1 ] { \box_use:N \l_tmpa_box } }
409 \dim_set:Nn \l_tmpa_dim { \box_dp:N \l_tmpb_box }
410 \box_move_down:nn \l_tmpa_dim
411 {
412   \vbox:n
413   {
414     \hbox_to_wd:nn { \box_wd:N \l_tmpb_box }
415     {
416       \hfil
417       \tikz [ remember~picture , overlay ]
418       \coordinate (@@~north~east) ;
419     }
420     \hbox:n
421     {
422       \tikz [ remember~picture , overlay ]
423       \coordinate (@@~south~west) ;
424       \box_move_up:nn \l_tmpa_dim { \box_use:N \l_tmpb_box }
425     }
426   }
427 }

```

The `w`-node is created using the Tikz library `fit` after construction of the nodes `(@@~south~west)` and `(@@~north~east)`. It's not possible to construct by a standard `node` instruction because such a construction give a erroneous result with some engines (XeTeX, LuaTeX) although the result is good with pdf<sub>l</sub>atex (why?).

```

428 \tikz [ remember~picture , overlay ]
429 \node
430 [
431   node~contents = { } ,
432   name = nm - \int_use:N \g_@@_env_int -
433             \int_use:N \g_@@_row_int -
434             \int_use:N \g_@@_column_int - w,
435   alias =
436     \str_if_empty:NF \l_@@_name_str
437     {
438       \l_@@_name_str -
439       \int_use:N \g_@@_row_int -
440       \int_use:N \g_@@_column_int - w
441     } ,
442   inner~sep = \c_zero_dim ,
443   fit = (@@~south~west) (@@~north~east)
444 ]
445 ;
446 }
447 }
448 }
449 \cs_new_protected:Npn \@@_test_if_math_mode:
450 {
451   \ifmmode \else
452     \@@_error:n { Outside~math~mode }
453   \fi
454 }

```

First, we test if we are yet in an environment `{NiceArray}` (nested environments are forbidden).

```

455 \NewDocumentEnvironment { NiceArray } { 0 { } m ! 0 { } }
456 {
457   \@@_adapt_S_column:
458   \@@_test_if_math_mode:
459   \bool_if:NT \l_@@_in_NiceArray_bool

```

```

460 { \@@_error:n { Yet~in~NiceArray } }
461 \bool_set_true:N \l_@@_in_NiceArray_bool

```

We deactivate Tikz externalization (since we use Tikz pictures with the options `overlay` and `remember picture`, there would be errors).

```

462 \cs_if_exist:NT \tikz@library@external@loaded
463 { \tikzset { external / export = false } }
464 \group_insert_after:N \@@_after_array:
465 \tl_gclear_new:N \g_@@_lines_to_draw_tl

```

We increment the counter `\g_@@_env_int` which counts the environments `{NiceArray}`.

```

466 \int_gincr:N \g_@@_env_int
467 \bool_if:NF \l_@@_block_auto_columns_width_bool
468 { \dim_gzero_new:N \g_@@_max_cell_width_dim }

```

For the following variables, maybe we should create it only if we use the environment `{pNiceArrayRC}` or its variants.

```

469 \dim_gzero_new:N \g_@@_max_dp_row_zero_dim
470 \dim_gzero_new:N \g_@@_max_ht_row_zero_dim
471 \dim_gzero_new:N \g_@@_max_ht_row_one_dim
472 \keys_set:nn { NiceMatrix / NiceArray } { #1 , #3 }

```

If the user requires all the columns to have a width equal to the widest cell of the array, we read this length in the file `.aux` (of, course, this is possible only on the second run of LaTeX : on the first run, the dimension `\l_@@_columns_width_dim` will be set to zero — and the columns will have their natural width).

```

473 \bool_if:NT \l_@@_auto_columns_width_bool
474 {
475   \group_insert_after:N \@@_write_max_cell_width:
476   \cs_if_free:cTF { _@@_max_cell_width_ \int_use:N \g_@@_env_int }
477   { \dim_zero:N \l_@@_columns_width_dim }
478   {
479     \dim_set:Nx \l_@@_columns_width_dim
480     { \use:c { _@@_max_cell_width_ \int_use:N \g_@@_env_int } }
481   }

```

If the environment has a name, we read the value of the maximal value of the columns from `_@@_name_cell_widthname` (the value will be the correct value even if the number of the environment has changed (for example because the user has created or deleted an environment before the current one)).

```

482 \str_if_empty:NF \l_@@_name_str
483 {
484   \cs_if_free:cF { _@@_max_cell_width_ \l_@@_name_str }
485   {
486     \dim_set:Nx \l_@@_columns_width_dim
487     { \use:c { _@@_max_cell_width_ \l_@@_name_str } }
488   }
489 }
490 }

```

We don't want to patch any code and that's why some code is executed in a `\group_insert_after:N`. In particular, in this `\group_insert_after:N`, we will have to know the value of some parameters like `\l_@@_extra_nodes_bool`. That's why we transit via a global version for some variables.

```

491 \bool_gset_eq:NN \g_@@_extra_nodes_bool \l_@@_extra_nodes_bool
492 \dim_gset_eq:NN \g_@@_left_margin_dim \l_@@_left_margin_dim
493 \dim_gset_eq:NN \g_@@_right_margin_dim \l_@@_right_margin_dim
494 \dim_gset_eq:NN \g_@@_extra_right_margin_dim \l_@@_extra_right_margin_dim
495 \tl_gset_eq:NN \g_@@_name_str \l_@@_name_str

```

The environment `{array}` uses internally the command `\ialign` and, in particular, this command `\ialign` sets `\everycr` to `{}`. However, we want to use `\everycr` in our array. The solution is to give



to `\ialign` a new definition (giving to `\everycr` the value we want) that will revert automatically to its default definition after the first utilisation.<sup>19</sup>

```

496   \cs_set:Npn \ialign
497   {
498     \everycr
499     {
500       \noalign
501       {
502         \int_gzero:N \g_@@_column_int
503         \bool_if:NT \l_@@_hlines_bool
504         {
505           \int_compare:nNnT \g_@@_row_int > { -1 }
506           {
507             \hrule \@height \arrayrulewidth
508             \skip_vertical:n { - \arrayrulewidth }
509           }
510         }
511       }
512     }
513     \tabskip = \c_zero_skip
514     \cs_set:Npn \ialign
515     {
516       \everycr { }
517       \tabskip = \c_zero_skip
518       \halign
519     }
520     \halign
521   }

```

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}`.

```

522   \dim_compare:nNnTF \l_@@_columns_width_dim = \c_zero_dim
523   {
524     \newcolumntype L { > \@@_Cell: l < \@@_end_Cell: }
525     \newcolumntype C { > \@@_Cell: c < \@@_end_Cell: }
526     \newcolumntype R { > \@@_Cell: r < \@@_end_Cell: }
527   }

```

If there is an option that specify that all the columns must have the same width, the column types L, C and R are in fact defined upon the column type w of `array` which is, in fact, redefined below.

```

528   {
529     \newcolumntype L { w l { \dim_use:N \l_@@_columns_width_dim } }
530     \newcolumntype C { w c { \dim_use:N \l_@@_columns_width_dim } }
531     \newcolumntype R { w r { \dim_use:N \l_@@_columns_width_dim } }
532   }

```

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:`.

```

533   \cs_set_eq:NN \NC@find@w \relax
534   \cs_set_eq:NN \NC@find@W \relax

```

We redefine the column types w and W of the package `array`.

```

535   \@@_renewcolumntype:nn w { }
536   \@@_renewcolumntype: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`.

<sup>19</sup>With this programming, we will have, in the cells of the array, a clean version of `\ialign`. That's necessary: the user will probably not employ directly `\ialign` in the array... but more likely environments that utilize `\ialign` internally (e.g.: `{substack}`)

```

537 \exp_args:Nx \newcolumntype \l_@@_letter_for_dotted_lines_str
538 {
539   !
540   {
541     \skip_horizontal:n { 0.53 pt }
542     \bool_gset_true:N \g_@@_extra_nodes_bool

```

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.

```

543 \int_compare:nNnT \g_@@_column_int > \g_@@_last_vdotted_col_int
544 {
545   \int_gset_eq:NN \g_@@_last_vdotted_col_int \g_@@_column_int
546   \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`.

```

547 { \@@_vdottedline:n { \int_use:N \g_@@_column_int } }
548 }
549 }
550 }

```

The commands `\Ldots`, `\Cdots`, etc. will be defined only in the environment `{NiceArray}`.

```

551 \cs_set_eq:NN \Ldots \@@_Ldots
552 \cs_set_eq:NN \Cdots \@@_Cdots
553 \cs_set_eq:NN \Vdots \@@_Vdots
554 \cs_set_eq:NN \Ddots \@@_Ddots
555 \cs_set_eq:NN \Iddots \@@_Iddots
556 \cs_set_eq:NN \hdottedline \@@_hdottedline:
557 \cs_set_eq:NN \Hspace \@@_Hspace:
558 \cs_set_eq:NN \Hdotsfor \@@_Hdotsfor
559 \cs_set_eq:NN \multicolumn \@@_multicolumn:nnn
560 \bool_if:NT \l_@@_renew_dots_bool
561 {
562   \cs_set_eq:NN \ldots \@@_Ldots
563   \cs_set_eq:NN \cdots \@@_Cdots
564   \cs_set_eq:NN \vdots \@@_Vdots
565   \cs_set_eq:NN \ddots \@@_Ddots
566   \cs_set_eq:NN \iddots \@@_Iddots
567   \cs_set_eq:NN \dots \@@_Ldots
568   \cs_set_eq:NN \hdotsfor \@@_Hdotsfor
569 }
570 \bool_if:NT \c_@@_siunitx_loaded_bool \@@_renew_NC@rewrite@S:

```

The sequence `\g_@@_empty_cells_seq` will contain a list of “empty” cells (not all the empty cells of the matrix). If we want to indicate that the cell in row  $i$  and column  $j$  must be considered as empty, the token list “ $i-j$ ” will be put in this sequence.

```

571 \seq_gclear_new:N \g_@@_empty_cells_seq

```

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).

```

572 \seq_gclear_new:N \g_@@_multicolumn_cells_seq
573 \seq_gclear_new:N \g_@@_multicolumn_sizes_seq

```

The counter `\g_@@_row_int` will be used to count the rows of the array (its incrementation will be in the first cell of the row). At the end of the environment `{array}`, this counter will give the total number of rows of the matrix.

```

574 \int_gzero_new:N \g_@@_row_int
575 \int_gset:Nn \g_@@_row_int { \l_@@_nb_first_row_int - 1 }

```

The counter `\g_@@_column_int` 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_@@_column_total_int`. These counters are updated in the command `\@@_Cell:` executed at the beginning of each cell.

```

576 \int_gzero_new:N \g_@@_column_int
577 \int_gzero_new:N \g_@@_column_total_int

578 \int_gzero_new:N \g_@@_last_vdotted_col_int
579 \int_gset:Nn \g_@@_last_vdotted_col_int { -1 }
580 \cs_set_eq:NN \@@_ifnextchar \new@ifnextchar

```

The extra horizontal spaces on both sides of an environment `{array}` should be considered as a bad idea of standard LaTeX. In the environment `{matrix}` the package `amsmath` prefers to suppress these spaces with instructions “`\hskip -\arraycolsep`”. In the same way, we decide to suppress them in `{NiceArray}`. However, for better compatibility, we give an option `exterior-arraycolsep` to control this feature.

```

581 \bool_if:NF \l_@@_exterior_arraycolsep_bool
582 { \skip_horizontal:n { - \arraycolsep } }

583 \skip_horizontal:n \l_@@_left_margin_dim
584 \skip_horizontal:n \l_@@_extra_left_margin_dim

```

Eventually, the environment `{NiceArray}` is defined upon the environment `{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`.

```

585 \bool_if:NTF \c_@@_revtex_bool
586 {
587     \cs_set_eq:NN \@acoll \@arrayacol
588     \cs_set_eq:NN \@acolr \@arrayacol
589     \cs_set_eq:NN \@acol \@arrayacol
590     \cs_set:Npn \@halignto { }
591     \@array@array
592 }
593 \array

```

The token list `\l_@@_pos_env_str`, which is the argument of `\array` (or `\@array@array`) will contain one of the values `t`, `c` or `b`.

```

594 [ \l_@@_pos_env_str ] { #2 }
595 }

596 { \endarray
597     \bool_if:NF \l_@@_exterior_arraycolsep_bool
598     { \skip_horizontal:n { - \arraycolsep } }
599     \skip_horizontal:n \g_@@_right_margin_dim
600     \skip_horizontal:n \g_@@_extra_right_margin_dim
601 }

```

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

```

602 \NewDocumentEnvironment { pNiceMatrix } { }
603 {
604     \@@_test_if_math_mode:
605     \left( \begin{NiceMatrix}
606 }
607 { \end{NiceMatrix} \right) }

```

```

608 \NewDocumentEnvironment { bNiceMatrix } { }
609 {
610   \@@_test_if_math_mode:
611   \left[ \begin{NiceMatrix}
612   }
613   { \end{NiceMatrix} \right] }
614 \NewDocumentEnvironment { BNiceMatrix } { }
615 {
616   \@@_test_if_math_mode:
617   \left\{ \begin{NiceMatrix}
618   }
619   { \end{NiceMatrix} \right\} }
620 \NewDocumentEnvironment { vNiceMatrix } { }
621 {
622   \@@_test_if_math_mode:
623   \left\lvert \begin{NiceMatrix}
624   }
625   { \end{NiceMatrix} \right\rvert }
626 \NewDocumentEnvironment { VNiceMatrix } {}
627 {
628   \@@_test_if_math_mode:
629   \left\lVert \begin{NiceMatrix}
630   }
631   { \end{NiceMatrix} \right\rVert }

```

For the option `columns-width=auto` (or the option `auto-columns-width` of the environment `{NiceMatrixBlock}`), we want to know the maximal width of the cells of the array (except the cells of the “exterior” column of an environment of the kind of `{pNiceAccayC}`). This length can be known only after the end of the construction of the array (or at the end of the environment `{NiceMatrixBlock}`). That’s why we store this value in the main `.aux` file and it will be available in the next run. We write a dedicated command for this because it will be called in a `\group_insert_after:N`.

```

632 \cs_new_protected:Nn \@@_write_max_cell_width:
633 {
634   \bool_if:NF \l_@@_block_auto_columns_width_bool
635   {
636     \iow_now:Nn \@mainaux \ExplSyntaxOn
637     \iow_now:Nx \@mainaux
638     {
639       \cs_gset:cpn { @@_max_cell_width_ \int_use:N \g_@@_env_int }
640       { \dim_use:N \g_@@_max_cell_width_dim }
641     }
642   }

```

If the environment has a name, we also create an alias named `\@@_max_cell_width_name`.

```

642   \str_if_empty:NF \g_@@_name_str
643   {
644     \iow_now:Nx \@mainaux
645     {
646       \cs_gset:cpn { @@_max_cell_width_ \g_@@_name_str }
647       { \dim_use:N \g_@@_max_cell_width_dim }
648     }
649   }
650   \iow_now:Nn \@mainaux \ExplSyntaxOff
651 }
652 }

```

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.

```

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

```

If the cell is an implicit cell (that is after the symbol `\\` of end of row), the cell must, of course, be considered as empty. It's easy to check whether we are in this situation considering the correspondent Tikz node.

```

654 {
655   \cs_if_free:cTF
656   { pgf@sh@ns@nm -\int_use:N \g_@@_env_int - \int_use:N #1 - \int_use:N #2 }
657   \prg_return_false:

```

We manage a list of “empty cells” called `\g_@@_empty_cells_seq`. In fact, this list is not a list of all the empty cells of the array but only those explicitly declared empty for some reason. It's easy to check if the current cell is in this list.

```

658 {
659   \seq_if_in:NxTF \g_@@_empty_cells_seq { \int_use:N #1 - \int_use:N #2 }
660   \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.

```

661 {
662   \begin { pgfpicture }

```

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

```

663   \tl_set:Nx \l_tmpa_tl
664   { nm - \int_use:N \g_@@_env_int - \int_use:N #1 - \int_use:N #2 }
665   \pgfpointanchor \l_tmpa_tl { east }
666   \dim_gset:Nn \g_tmpa_dim \pgf@x
667   \pgfpointanchor \l_tmpa_tl { west }
668   \dim_gset:Nn \g_tmpb_dim \pgf@x
669   \end { pgfpicture }
670   \dim_compare:nNnTF
671   { \dim_abs:n { \g_tmpb_dim - \g_tmpa_dim } } < { 0.5 pt }
672   \prg_return_false:
673   \prg_return_true:
674 }
675 }
676 }

```

The argument of the following command `\@@_instruction_of_type:n` is the type of the instruction (`Cdots`, `Vdots`, `Ddots`, etc.). This command writes in `\g_@@_lines_to_draw_tl` the instruction that will really 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 & \Hdotsfor{2} & & \\
\end{pNiceMatrix}

```

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

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

```

\@@_draw_Cdots:nn {2}{2}
\@@_draw_Hdotsfor:nnn {3}{2}{2}
677 \bool_if:NTF \c_@@_draft_bool
678 { \cs_set_protected:Npn \@@_instruction_of_type:n #1 { } }
679 {
680   \cs_new_protected:Npn \@@_instruction_of_type:n #1
681   {
682     \tl_gput_right:Nx \g_@@_lines_to_draw_tl
683     {
684       \exp_not:c { @@ _ draw _ #1 : nn }
685       { \int_use:N \g_@@_row_int }
686       { \int_use:N \g_@@_column_int }
687     }
688   }
689 }

```

## 15.6 After the construction of the array

```

690 \cs_new_protected:Nn \@@_after_array:
691 {
692   \int_compare:nNnTF \g_@@_row_int > \c_zero_int
693     \@@_after_array_i:
694     { \@@_error:n { Zero~row } }
695 }
696 \@@_msg_new:nn { Zero~row }
697 {
698   There-is-a-problem.-Maybe-your-environment-{\@currentenv}\-is-empty.-
699   Maybe-you-have-used-L,~C-and-R-instead-of~l,~c-and-r-in-the-preamble-
700   of-your-environment. \\\
701   If-you-go-on,~the-result-may-be-incorrect.
702 }

```

We deactivate Tikz externalization (since we use Tikz pictures with the options `overlay` and `remember picture`, there would be errors).

```

703 \cs_new_protected:Nn \@@_after_array_i:
704 {
705   \group_begin:
706   \cs_if_exist:NT \tikz@library@external@loaded
707   { \tikzset { external / export = false } }

```

Now, the definition of the counters `\g_@@_column_int` and `\g_@@_column_total_int` change: `\g_@@_column_int` will be the number of columns without the exterior column (in an environment like `{pNiceArrayC}`) and `\g_@@_column_total_int` will be the number of columns with this exterior column.

```

708   \int_gset_eq:NN \g_@@_column_int \g_@@_column_total_int
709   \bool_if:nT { \l_@@_exterior_column_bool && \g_@@_exterior_column_found_bool }
710   { \int_gdecr:N \g_@@_column_int }

```

The sequence `\g_@@_yet_drawn_seq` contains a list of lines which have been drawn previously in the matrix. We maintain this sequence because we don't want to draw two overlapping lines.

```

711   \seq_gclear_new:N \g_@@_yet_drawn_seq

```

By default, the diagonal lines will be parallelized<sup>20</sup>. There are two types of diagonals lines: the `\Ddots` diagonals and the `\Iddots` diagonals. We have to count both types in order to know whether a diagonal is the first of its type in the current `{NiceArray}` environment.

```

712   \bool_if:NT \l_@@_parallelize_diags_bool
713   {
714     \int_zero_new:N \l_@@_ddots_int
715     \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 `\Iddots` diagonal.

```

716     \dim_zero_new:N \l_@@_delta_x_one_dim
717     \dim_zero_new:N \l_@@_delta_y_one_dim
718     \dim_zero_new:N \l_@@_delta_x_two_dim
719     \dim_zero_new:N \l_@@_delta_y_two_dim
720   }

```

If the user has used the option `create-extra-nodes`, the “medium nodes” and “large nodes” are created. We recall that the command `\@@_create_extra_nodes:`, when used once, becomes `no-op` (in the current TeX group).

```

721   \bool_if:NT \g_@@_extra_nodes_bool \@@_create_extra_nodes:

```

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

```

722   \tl_if_empty:NF \g_@@_lines_to_draw_tl
723   {

```

---

<sup>20</sup>It's possible to use the option `parallelize-diags` to disable this parallelization.

```

724 \int_zero_new:N \l_@@_initial_i_int
725 \int_zero_new:N \l_@@_initial_j_int
726 \int_zero_new:N \l_@@_final_i_int
727 \int_zero_new:N \l_@@_final_j_int
728 \bool_set_false:N \l_@@_initial_open_bool
729 \bool_set_false:N \l_@@_final_open_bool
730 \g_@@_lines_to_draw_tl
731 }
732 \tl_gclear:N \g_@@_lines_to_draw_tl

```

Now, the code-after.

```

733 \tikzset
734 {
735   every-picture / .style =
736   {
737     overlay ,
738     remember-picture ,
739     name-prefix = nm - \int_use:N \g_@@_env_int -
740   }
741 }
742 \cs_set_eq:NN \line \l_@@_line:nn
743 \g_@@_code_after_tl
744 \tl_gclear:N \g_@@_code_after_tl
745 \group_end:
746 }

```

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 a open one, we use the “medium node” (the medium and large nodes are created with `\@@_create_extra_nodes:` if they have not been created yet).

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

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

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

```

747 \cs_new_protected:Nn \l_@@_find_extremities_of_line:nnnn
748 {
749   \int_set:Nn \l_@@_initial_i_int { #1 }
750   \int_set:Nn \l_@@_initial_j_int { #2 }
751   \int_set:Nn \l_@@_final_i_int { #1 }
752   \int_set:Nn \l_@@_final_j_int { #2 }
753   \bool_set_false:N \l_@@_initial_open_bool
754   \bool_set_false:N \l_@@_final_open_bool

```

We will do two loops: one when determinating the initial cell and the other when determinating the final cell. The boolean `\l_@@_stop_loop_bool` will be used to control these loops.

```

755   \bool_set_false:N \l_@@_stop_loop_bool
756   \bool_do_until:Nn \l_@@_stop_loop_bool
757   {
758     \int_add:Nn \l_@@_final_i_int { #3 }
759     \int_add:Nn \l_@@_final_j_int { #4 }

```

We test if we are still in the matrix.

```

760     \bool_if:nTF
761     {
762       \int_compare_p:nNn
763       \l_@@_final_i_int < { \l_@@_nb_first_row_int - 1 }
764       || \int_compare_p:nNn \l_@@_final_i_int > \g_@@_row_int
765       || \int_compare_p:nNn \l_@@_final_j_int < \c_one_int
766       || \int_compare_p:nNn \l_@@_final_j_int > \g_@@_column_total_int

```

If you arrive in the column `C` of an environment with such columns (like `{pNiceArrayC}`), you must consider that we are *outside* the matrix except if we are drawing a vertical line (included in the column `C`).

```

767       || \int_compare_p:nNn \l_@@_final_j_int > \g_@@_column_int
768       && \int_compare_p:nNn { #4 } > \c_zero_int
769     }

```

If we are outside the matrix, we have found the extremity of the dotted line and it's a *open* extremity.

```

770     {
771       \bool_set_true:N \l_@@_final_open_bool

```

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).

```

772       \int_sub:Nn \l_@@_final_i_int { #3 }
773       \int_sub:Nn \l_@@_final_j_int { #4 }
774       \bool_set_true:N \l_@@_stop_loop_bool
775     }

```

If we are in the matrix, we test if 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`.

```

776     {
777       \@@_if_not_empty_cell:nnT \l_@@_final_i_int \l_@@_final_j_int
778       { \bool_set_true:N \l_@@_stop_loop_bool }
779     }
780   }

```

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

```

781   \bool_set_false:N \l_@@_stop_loop_bool
782   \bool_do_until:Nn \l_@@_stop_loop_bool
783   {
784     \int_sub:Nn \l_@@_initial_i_int { #3 }
785     \int_sub:Nn \l_@@_initial_j_int { #4 }
786     \bool_if:nTF
787     {
788       \int_compare_p:nNn \l_@@_initial_i_int < \l_@@_nb_first_row_int
789       ||
790       \int_compare_p:nNn \l_@@_initial_i_int > \g_@@_row_int

```



```

791         ||
792         \int_compare_p:nNn \l_@@_initial_j_int < 1
793         ||
794         \int_compare_p:nNn \l_@@_initial_j_int > \g_@@_column_total_int
795     }
796     {
797         \bool_set_true:N \l_@@_initial_open_bool
798         \int_add:Nn \l_@@_initial_i_int { #3 }
799         \int_add:Nn \l_@@_initial_j_int { #4 }
800         \bool_set_true:N \l_@@_stop_loop_bool
801     }
802     {
803         \@@_if_not_empty_cell:nnT
804         \l_@@_initial_i_int \l_@@_initial_j_int
805         { \bool_set_true:N \l_@@_stop_loop_bool }
806     }
807 }

```

If we have at least one open extremity, we create the “medium nodes” in the matrix (in the case of an open extremity, the dotted line uses the “medium node” of the last empty cell). We remind that, when used once, the command `\@@_create_extra_nodes:` becomes no-op in the current TeX group.

```

808     \bool_if:nT { \l_@@_initial_open_bool || \l_@@_final_open_bool }
809     \@@_create_extra_nodes:
810 }

```

If the dotted line to draw is in the list of the previously drawn lines (`\g_@@_yet_drawn_seq`), we don’t draw (so, we won’t have overlapping lines in the PDF). The token list `\l_tmpa_tl` is the 4-list characteristic of the line.

```

811 \prg_set_conditional:Npnn \@@_if_yet_drawn: { F }
812 {
813     \tl_set:Nx \l_tmpa_tl
814     {
815         \int_use:N \l_@@_initial_i_int -
816         \int_use:N \l_@@_initial_j_int -
817         \int_use:N \l_@@_final_i_int -
818         \int_use:N \l_@@_final_j_int
819     }
820     \seq_if_in:NVTF \g_@@_yet_drawn_seq \l_tmpa_tl

```

If the dotted line to draw is not in the list, we add it to the list `\g_@@_yet_drawn_seq`.

```

821     \prg_return_true:
822     {
823         \seq_gput_left:NV \g_@@_yet_drawn_seq \l_tmpa_tl
824         \prg_return_false:
825     }
826 }

```

The command `\@@_retrieve_coords:nn` retrieves the Tikz coordinates of the two extremities of the dotted line we will have to draw <sup>21</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 prefix and the anchor that must be used for the two nodes.

The coordinates are stored in `\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}`.

```

827 \cs_new_protected:Nn \@@_retrieve_coords:nn
828 {
829     \dim_gzero_new:N \g_@@_x_initial_dim

```

<sup>21</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.

```

830 \dim_gzero_new:N \g_@@_y_initial_dim
831 \dim_gzero_new:N \g_@@_x_final_dim
832 \dim_gzero_new:N \g_@@_y_final_dim
833 \begin { tikzpicture } [ remember-picture ]
834 \tikz@parse@node \pgfutil@firstofone
835 ( nm - \int_use:N \g_@@_env_int -
836 \int_use:N \l_@@_initial_i_int -
837 \int_use:N \l_@@_initial_j_int #1 )
838 \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
839 \dim_gset:Nn \g_@@_y_initial_dim \pgf@y
840 \tikz@parse@node \pgfutil@firstofone
841 ( nm - \int_use:N \g_@@_env_int -
842 \int_use:N \l_@@_final_i_int -
843 \int_use:N \l_@@_final_j_int #2 )
844 \dim_gset:Nn \g_@@_x_final_dim \pgf@x
845 \dim_gset:Nn \g_@@_y_final_dim \pgf@y
846 \end { tikzpicture }
847 }
848 \cs_generate_variant:Nn \@@_retrieve_coords:nn { x x }

849 \cs_new_protected:Nn \@@_draw_Ldots:nn
850 {
851 \@@_find_extremities_of_line:nnnn { #1 } { #2 } \c_zero_int \c_one_int
852 \@@_if_yet_drawn:F \@@_actually_draw_Ldots:
853 }

```

The command `\@@_actually_draw_Ldots:` actually draws the `Ldots` line using `\l_@@_initial_i_int`, `\l_@@_initial_j_int`, `\l_@@_initial_open_bool`, `\l_@@_final_i_int`, `\l_@@_final_j_int` and `\l_@@_final_open_bool`. We have a dedicated command because it is used also by `\Hdotsfor`.

```

854 \cs_new_protected:Nn \@@_actually_draw_Ldots:
855 {
856 \@@_retrieve_coords:xx
857 {
858 \bool_if:NTF \l_@@_initial_open_bool
859 { - medium.base~west }
860 { .base~east }
861 }
862 {
863 \bool_if:NTF \l_@@_final_open_bool
864 { - medium.base~east }
865 { . base~west }
866 }
867 \bool_if:NT \l_@@_initial_open_bool
868 { \dim_gset_eq:NN \g_@@_y_initial_dim \g_@@_y_final_dim }
869 \bool_if:NT \l_@@_final_open_bool
870 { \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 text.

```

871 \dim_gadd:Nn \g_@@_y_initial_dim { 0.53 pt }
872 \dim_gadd:Nn \g_@@_y_final_dim { 0.53 pt }
873 \@@_draw_tikz_line:
874 }

875 \cs_new_protected:Nn \@@_draw_Cdots:nn
876 {
877 \@@_find_extremities_of_line:nnnn { #1 } { #2 } \c_zero_int \c_one_int
878 \@@_if_yet_drawn:F
879 {
880 \@@_retrieve_coords:xx
881 {
882 \bool_if:NTF \l_@@_initial_open_bool

```

```

883         { - medium.mid-west }
884         { .mid-east }
885     }
886     {
887         \bool_if:NTF \l_@@_final_open_bool
888         { - medium.mid-east }
889         { .mid-west }
890     }
891     \bool_if:NT \l_@@_initial_open_bool
892     { \dim_gset_eq:NN \g_@@_y_initial_dim \g_@@_y_final_dim }
893     \bool_if:NT \l_@@_final_open_bool
894     { \dim_gset_eq:NN \g_@@_y_final_dim \g_@@_y_initial_dim }
895     \@@_draw_tikz_line:
896 }
897 }

```

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?).

```

898 \cs_new_protected:Nn \@@_draw_Vdots:nn
899 {
900     \@@_find_extremities_of_line:nnnn { #1 } { #2 } \c_one_int \c_zero_int
901     \@@_if_yet_drawn:F
902     { \@@_retrieve_coords:xx
903         {
904             \bool_if:NTF \l_@@_initial_open_bool
905             { - medium.north-west }
906             { .south-west }
907         }
908         {
909             \bool_if:NTF \l_@@_final_open_bool
910             { - medium.south-west }
911             { .north-west }
912         }
913     }
914 }

```

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

```

913     \bool_set:Nn \l_tmpa_bool
914     { \dim_compare_p:nNn \g_@@_x_initial_dim = \g_@@_x_final_dim }
915     \@@_retrieve_coords:xx
916     {
917         \bool_if:NTF \l_@@_initial_open_bool
918         { - medium.north }
919         { .south }
920     }
921     {
922         \bool_if:NTF \l_@@_final_open_bool
923         { - medium.south }
924         { .north }
925     }
926 }

```

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

```

926     \bool_set:Nn \l_tmpb_bool
927     { \dim_compare_p:nNn \g_@@_x_initial_dim = \g_@@_x_final_dim }
928     \bool_if:NF \l_tmpb_bool
929     {
930         \dim_gset:Nn \g_@@_x_initial_dim
931         {
932             \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
933             \g_@@_x_initial_dim \g_@@_x_final_dim
934         }
935         \dim_gset_eq:NN \g_@@_x_final_dim \g_@@_x_initial_dim
936     }

```

```

936     }
937     \@@_draw_tikz_line:
938 }
939 }

```

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.

```

940 \cs_new_protected:Nn \@@_draw_Ddots:nn
941 {
942   \@@_find_extremities_of_line:nnnn { #1 } { #2 } \c_one_int \c_one_int
943   \@@_if_yet_drawn:F
944   {
945     \@@_retrieve_coords:xx
946     {
947       \bool_if:NTF \l_@@_initial_open_bool
948       { - medium.north-west }
949       { .south-east }
950     }
951     {
952       \bool_if:NTF \l_@@_final_open_bool
953       { - medium.south-east }
954       { .north-west }
955     }
956   }
957 }

```

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.

```

956   \bool_if:NT \l_@@_parallelize_diags_bool
957   {
958     \int_incr:N \l_@@_ddots_int
959   }

```

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

```

959   \int_compare:nNnTF \l_@@_ddots_int = \c_one_int

```

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.

```

960   {
961     \dim_set:Nn \l_@@_delta_x_one_dim
962     { \g_@@_x_final_dim - \g_@@_x_initial_dim }
963     \dim_set:Nn \l_@@_delta_y_one_dim
964     { \g_@@_y_final_dim - \g_@@_y_initial_dim }
965   }

```

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`.

```

966   {
967     \dim_gset:Nn \g_@@_y_final_dim
968     {
969       \g_@@_y_initial_dim +
970       ( \g_@@_x_final_dim - \g_@@_x_initial_dim ) *
971       \dim_ratio:nn \l_@@_delta_y_one_dim \l_@@_delta_x_one_dim
972     }
973   }
974 }

```

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

```

975   \@@_draw_tikz_line:
976 }
977 }

```

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

```

978 \cs_new_protected:Nn \@@_draw_Iddots:nn
979 {
980   \@@_find_extremities_of_line:nnnn { #1 } { #2 } 1 { -1 }
981   \@@_if_yet_drawn:F
982   { \@@_retrieve_coords:xx
983     {
984       \bool_if:NTF \l_@@_initial_open_bool
985       { - medium.north~east }
986       { .south~west }
987     }
988     {
989       \bool_if:NTF \l_@@_final_open_bool
990       { - medium.south~west }
991       { .north~east }
992     }
993     \bool_if:NT \l_@@_parallelize_diags_bool
994     {
995       \int_incr:N \l_@@_iddots_int
996       \int_compare:nNnTF \l_@@_iddots_int = \c_one_int
997       {
998         \dim_set:Nn \l_@@_delta_x_two_dim
999         { \g_@@_x_final_dim - \g_@@_x_initial_dim }
1000        \dim_set:Nn \l_@@_delta_y_two_dim
1001        { \g_@@_y_final_dim - \g_@@_y_initial_dim }
1002      }
1003      {
1004        \dim_gset:Nn \g_@@_y_final_dim
1005        {
1006          \g_@@_y_initial_dim +
1007          ( \g_@@_x_final_dim - \g_@@_x_initial_dim ) *
1008          \dim_ratio:nn \l_@@_delta_y_two_dim \l_@@_delta_x_two_dim
1009        }
1010      }
1011    }
1012    \@@_draw_tikz_line:
1013  }
1014 }

```

## 15.7 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`.

```

1015 \cs_new_protected:Nn \@@_draw_tikz_line:
1016 {

```

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.

```

1017   \dim_zero_new:N \l_@@_l_dim
1018   \dim_set:Nn \l_@@_l_dim
1019   {
1020     \fp_to_dim:n
1021     {
1022       sqrt
1023       ( ( \dim_use:N \g_@@_x_final_dim
1024         - \dim_use:N \g_@@_x_initial_dim
1025         ) ^ 2
1026         +
1027         ( \dim_use:N \g_@@_y_final_dim
1028         - \dim_use:N \g_@@_y_initial_dim

```

```

1029         ) ^ 2
1030     )
1031 }
1032 }

```

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

```

1033 \dim_compare:nNnF \l_@@_l_dim = \c_zero_dim

```

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

```

1034 {
1035   \bool_if:NTF \l_@@_initial_open_bool
1036   {
1037     \bool_if:NTF \l_@@_final_open_bool
1038     {
1039       \int_set:Nn \l_tmpa_int
1040       { \dim_ratio:nn \l_@@_l_dim { 0.45 em } }
1041     }
1042     {
1043       \int_set:Nn \l_tmpa_int
1044       { \dim_ratio:nn { \l_@@_l_dim - 0.3 em } { 0.45 em } }
1045     }
1046   }
1047   {
1048     \bool_if:NTF \l_@@_final_open_bool
1049     {
1050       \int_set:Nn \l_tmpa_int
1051       { \dim_ratio:nn { \l_@@_l_dim - 0.3 em } { 0.45 em } }
1052     }
1053     {
1054       \int_set:Nn \l_tmpa_int
1055       { \dim_ratio:nn { \l_@@_l_dim - 0.6 em } { 0.45 em } }
1056     }
1057   }

```

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

```

1058   \dim_set:Nn \l_tmpa_dim
1059   {
1060     ( \g_@@_x_final_dim - \g_@@_x_initial_dim ) *
1061     \dim_ratio:nn { 0.45 em } \l_@@_l_dim
1062   }
1063   \dim_set:Nn \l_tmpb_dim
1064   {
1065     ( \g_@@_y_final_dim - \g_@@_y_initial_dim ) *
1066     \dim_ratio:nn { 0.45 em } \l_@@_l_dim
1067   }

```

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`.

```

1068   \int_set:Nn \l_tmpb_int
1069   {
1070     \bool_if:NTF \l_@@_initial_open_bool
1071     { \bool_if:NTF \l_@@_final_open_bool 1 0 }
1072     { \bool_if:NTF \l_@@_final_open_bool 2 1 }
1073   }

```

In the loop over the dots (`\int_step_inline:nnnn`), 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.

```

1074   \dim_gadd:Nn \g_@@_x_initial_dim
1075   {
1076     ( \g_@@_x_final_dim - \g_@@_x_initial_dim ) *

```

```

1077         \dim_ratio:nn
1078         { \l_@@_l_dim - 0.45 em * \l_tmpa_int } { \l_@@_l_dim * 2 } *
1079         \l_tmpb_int
1080     }

```

(In a multiplication of a dimension and an integer, the integer must always be put in second position.)

```

1081     \dim_gadd:Nn \g_@@_y_initial_dim
1082     {
1083         ( \g_@@_y_final_dim - \g_@@_y_initial_dim ) *
1084         \dim_ratio:nn
1085         { \l_@@_l_dim - 0.45 em * \l_tmpa_int }
1086         { \l_@@_l_dim * 2 } *
1087         \l_tmpb_int
1088     }
1089     \begin { tikzpicture } [ overlay ]
1090     \int_step_inline:nnnn 0 1 \l_tmpa_int
1091     {
1092         \pgfpathcircle
1093         { \pgfpoint { \g_@@_x_initial_dim } { \g_@@_y_initial_dim } }
1094         { 0.53 pt }
1095         \pgfusepath { fill }
1096         \dim_gadd:Nn \g_@@_x_initial_dim \l_tmpa_dim
1097         \dim_gadd:Nn \g_@@_y_initial_dim \l_tmpb_dim
1098     }
1099     \end { tikzpicture }
1100 }
1101 }

```

## 15.8 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).

```

1102 \cs_set_eq:NN \@@_ldots \ldots
1103 \cs_set_eq:NN \@@_cdots \cdots
1104 \cs_set_eq:NN \@@_vdots \vdots
1105 \cs_set_eq:NN \@@_ddots \ddots
1106 \cs_set_eq:NN \@@_iddots \iddots

```

The command `\@@_add_to_empty_cells:` adds the current cell to `\g_@@_empty_cells_seq` which is the list of the empty cells (the cells explicitly declared “empty”: there may be, of course, other empty cells in the matrix).

```

1107 \cs_new_protected:Nn \@@_add_to_empty_cells:
1108 {
1109     \seq_gput_right:Nx \g_@@_empty_cells_seq
1110     { \int_use:N \g_@@_row_int - \int_use:N \g_@@_column_int }
1111 }

```

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}`).

```

1112 \NewDocumentCommand \@@_Ldots { s }
1113 {
1114     \bool_if:nF { #1 } { \@@_instruction_of_type:n { Ldots } }
1115     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_ldots }
1116     \@@_add_to_empty_cells:
1117 }

```

```

1118 \NewDocumentCommand \@@_Cdots { s }
1119 {
1120   \bool_if:nF { #1 } { \@@_instruction_of_type:n { Cdots } }
1121   \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_cdots }
1122   \@@_add_to_empty_cells:
1123 }

1124 \NewDocumentCommand \@@_Vdots { s }
1125 {
1126   \bool_if:nF { #1 } { \@@_instruction_of_type:n { Vdots } }
1127   \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_vdots }
1128   \@@_add_to_empty_cells:
1129 }

1130 \NewDocumentCommand \@@_Ddots { s }
1131 {
1132   \bool_if:nF { #1 } { \@@_instruction_of_type:n { Ddots } }
1133   \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_ddots }
1134   \@@_add_to_empty_cells:
1135 }

1136 \NewDocumentCommand \@@_Iddots { s }
1137 {
1138   \bool_if:nF { #1 } { \@@_instruction_of_type:n { Iddots } }
1139   \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_iddots }
1140   \@@_add_to_empty_cells:
1141 }

```

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

```

1142 \cs_new_protected:Nn \@@_Hspace:
1143 {
1144   \@@_add_to_empty_cells:
1145   \hspace
1146 }

```

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

```

1147 \cs_set_eq:NN \@@_old_multicolumn \multicolumn
1148 \cs_new:Npn \@@_multicolumn:nnn #1 #2 #3
1149 {
1150   \@@_old_multicolumn { #1 } { #2 } { #3 }
1151   \int_compare:nNnT #1 > 1
1152   {
1153     \seq_gput_left:Nx \g_@@_multicolumn_cells_seq
1154     { \int_eval:n \g_@@_row_int - \int_use:N \g_@@_column_int }
1155     \seq_gput_left:Nn \g_@@_multicolumn_sizes_seq { #1 }
1156   }
1157   \int_gadd:Nn \g_@@_column_int { #1 - 1 }
1158 }

```

The command `\@@_Hdotsfor` will be linked to `\Hdotsfor` in `{NiceArray}`. This command uses an optional argument like `\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`.

```

1159 \bool_if:NTF \c_@@_draft_bool
1160 {
1161   \NewDocumentCommand \@@_Hdotsfor { 0 { } m }
1162   { \prg_replicate:nn { #2 - 1 } { & } }

```



```

1163 }
1164 {
1165   \NewDocumentCommand \@@_Hdotsfor { 0 { } m }
1166   {
1167     \tl_gput_right:Nx \g_@@_lines_to_draw_tl
1168     {
1169       \@@_draw_Hdotsfor:nnn
1170       { \int_use:N \g_@@_row_int }
1171       { \int_use:N \g_@@_column_int }
1172       { #2 }
1173     }
1174     \prg_replicate:nn { #2 - 1 } { & }
1175   }
1176 }

```

```

1177 \cs_new_protected:Nn \@@_draw_Hdotsfor:nnn
1178 {
1179   \bool_set_false:N \l_@@_initial_open_bool
1180   \bool_set_false:N \l_@@_final_open_bool

```

For the row, it's easy.

```

1181   \int_set:Nn \l_@@_initial_i_int { #1 }
1182   \int_set:Nn \l_@@_final_i_int { #1 }

```

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

```

1183   \int_compare:nNnTF #2 = 1
1184   {
1185     \int_set:Nn \l_@@_initial_j_int 1
1186     \bool_set_true:N \l_@@_initial_open_bool
1187   }
1188   {
1189     \int_set:Nn \l_tmpa_int { #2 - 1 }
1190     \@@_if_not_empty_cell:nnTF \l_@@_initial_i_int \l_tmpa_int
1191     { \int_set:Nn \l_@@_initial_j_int { #2 - 1 } }
1192     {
1193       \int_set:Nn \l_@@_initial_j_int {#2}
1194       \bool_set_true:N \l_@@_initial_open_bool
1195     }
1196   }
1197   \int_compare:nNnTF { #2 + #3 - 1 } = \g_@@_column_int
1198   {
1199     \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
1200     \bool_set_true:N \l_@@_final_open_bool
1201   }
1202   {
1203     \int_set:Nn \l_tmpa_int { #2 + #3 }
1204     \@@_if_not_empty_cell:nnTF \l_@@_final_i_int \l_tmpa_int
1205     { \int_set:Nn \l_@@_final_j_int { #2 + #3 } }
1206     {
1207       \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
1208       \bool_set_true:N \l_@@_final_open_bool
1209     }
1210   }
1211   \bool_if:nT { \l_@@_initial_open_bool || \l_@@_final_open_bool }
1212   \@@_create_extra_nodes:
1213   \@@_actually_draw_Ldots:
1214 }

```

## 15.9 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 specification of two cells in the array (in the format *i-j*) and draws a dotted line between these cells.

```

1215 \cs_new_protected:Nn \@@_line:nn
1216 {
1217   \dim_zero_new:N \g_@@_x_initial_dim
1218   \dim_zero_new:N \g_@@_y_initial_dim
1219   \dim_zero_new:N \g_@@_x_final_dim
1220   \dim_zero_new:N \g_@@_y_final_dim
1221   \bool_set_false:N \l_@@_initial_open_bool
1222   \bool_set_false:N \l_@@_final_open_bool
1223   \begin { tikzpicture }
1224     \path~( #1 ) ~--~( #2 ) ~node[at~start]~( i ) ~{}~node[at~end]~( f ) ~{} ;
1225     \tikz@parse@node \pgfutil@firstofone ( i )
1226     \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1227     \dim_gset:Nn \g_@@_y_initial_dim \pgf@y
1228     \tikz@parse@node \pgfutil@firstofone ( f )
1229     \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1230     \dim_gset:Nn \g_@@_y_final_dim \pgf@y
1231   \end { tikzpicture }
1232   \@@_draw_tikz_line:
1233 }

```

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).

## 15.10 The commands to draw dotted lines to separate columns and rows

The command `\hdottedline` draws an horizontal dotted line to separate two rows. Similarly, the letter “.” in the preamble draws a vertical dotted line (the letter can be changed with the option `letter-for-dotted-lines`). Both mechanisms write instructions in the `code-after`. The actual instructions in the `code-after` use the commands `\@@_hdottedline:n` and `\@@_vdottedline:n`.

We want the horizontal lines at the same position<sup>22</sup> as the line created by `\hline` (or `\hdashline` of `arydshln`). To this end, we construct a “false row” and, in this row, we create a Tikz node (`\coordinate`) that will be used to have the *y*-value of the line.

```

1234 \cs_generate_variant:Nn \dim_set:Nn { N v }

```

Some extension, like the extension `doc`, do a redefinition of the command `\dotfill` of LaTeX. That's why we define a command `\@@_dotfill:` as we wish.

```

1235 \bool_if:NTF \c_@@_draft_bool
1236 { \cs_set_eq:NN \@@_dotfill: \prg_do_nothing: }
1237 {
1238   \cs_set:Npn \@@_dotfill:
1239   {
1240     \cleaders \hbox_to_wd:nn { .44 em } { \hss .\hss } \hfill
1241     \skip_horizontal:n \c_zero_dim
1242   }
1243 }

```

This command must *not* be protected because it starts with `\noalign`.

```

1244 \cs_new:Npn \@@_hdottedline:
1245 {
1246   \noalign
1247   {
1248     \bool_gset_true:N \g_@@_extra_nodes_bool
1249     \cs_if_exist:cTF { @@_width_ \int_use:N \g_@@_env_int }
1250     { \dim_set:Nv \l_tmpa_dim { @@_width_ \int_use:N \g_@@_env_int } }
1251     { \dim_set:Nn \l_tmpa_dim { 5 mm } }
1252     \hbox_overlap_right:n
1253     { \hbox_to_wd:nn
1254       {

```

<sup>22</sup>In fact, almost the same position because of the width of the line: the width of a dotted line is not the same as the width of a line created by `\hline`.

```

1255         \l_tmpa_dim + 2 \arraycolsep
1256         - \l_@@_left_margin_dim - \g_@@_right_margin_dim
1257     }
1258     \@@_dotfill:
1259 }
1260 }
1261 }

```

```

1262 \cs_new_protected:Nn \@@_vdottedline:n
1263 {

```

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

```

1264     \int_compare:nNnTF #1 = \c_zero_int
1265     { \@@_error:n { Use~of~.:~in~first~position } }
1266     {
1267         \@@_create_extra_nodes:
1268         \bool_if:NF \c_@@_draft_bool
1269         {
1270             \dim_zero_new:N \g_@@_x_initial_dim
1271             \dim_zero_new:N \g_@@_y_initial_dim
1272             \dim_zero_new:N \g_@@_x_final_dim
1273             \dim_zero_new:N \g_@@_y_final_dim
1274             \bool_set_true:N \l_@@_initial_open_bool
1275             \bool_set_true:N \l_@@_final_open_bool

```

In order to have the coordinates of the line to draw, we use the “large nodes”.

```

1276     \begin { tikzpicture } [ remember~picture ]
1277     \tikz@parse@node\pgfutil@firstofone
1278     ( 1 - #1 - large .north-east )
1279     \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1280     \dim_gset:Nn \g_@@_y_initial_dim \pgf@y
1281     \tikz@parse@node\pgfutil@firstofone
1282     ( \int_use:N \g_@@_row_int - #1 - large .south-east )
1283     \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1284     \dim_gset:Nn \g_@@_y_final_dim \pgf@y
1285     \end { tikzpicture }

```

However, if the w-nodes are created in the previous column (that is if the previous column was constructed explicitly or implicitly<sup>23</sup> with a letter w), we use the w-nodes to change the x-value of the nodes in order to have the dotted lines perfectly aligned when we use the environment `{NiceMatrixBlock}` with the option `auto-columns-width`.

```

1286     \cs_if_exist:cT
1287     { pgf@sh@ns@nm -\int_use:N \g_@@_env_int - 1 - #1 - w }
1288     {
1289         \begin { tikzpicture } [ remember~picture ]
1290         \tikz@parse@node\pgfutil@firstofone
1291         ( 1 - #1 - w .north-east )
1292         \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1293         \tikz@parse@node\pgfutil@firstofone
1294         ( \int_use:N \g_@@_row_int - #1 - w .south-east )
1295         \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1296         \end { tikzpicture }
1297         \dim_gadd:Nn \g_@@_x_initial_dim \arraycolsep
1298         \dim_gadd:Nn \g_@@_x_final_dim \arraycolsep
1299     }
1300     \@@_draw_tikz_line:
1301 }
1302 }
1303 }

```

<sup>23</sup>A column is constructed implicitly with the letter w if the option `columns-width` is used or if the environment `{NiceMatrixBlock}` is used with the option `auto-columns-width`.

```

1304 \@@_msg_new:nn { Use-of-~in-first-position }
1305 {
1306   You-can't-use-the-column-specifier~"\l_@@_letter_for_dotted_lines_str"~in-the-
1307   first-position-of-the-preamble-of-the-environment~{\@currentenv}. \\
1308   If-you-go-on,~this-dotted-line-will-be-ignored.
1309 }

```

## 15.11 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.

```

1310 \bool_new:N \l_@@_block_auto_columns_width_bool

```

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

```

1311 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
1312 {
1313   auto-columns-width .code:n =
1314   {
1315     \bool_set_true:N \l_@@_block_auto_columns_width_bool
1316     \dim_gzero_new:N \g_@@_max_cell_width_dim
1317     \bool_set_true:N \l_@@_auto_columns_width_bool
1318   }
1319 }

1320 \NewDocumentEnvironment { NiceMatrixBlock } { ! 0 { } }
1321 {
1322   \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
1323   \int_zero_new:N \l_@@_first_env_block_int
1324   \int_set:Nn \l_@@_first_env_block_int { \g_@@_env_int + 1 }
1325 }

```

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).

```

1326 {
1327   \bool_if:NT \l_@@_block_auto_columns_width_bool
1328   {
1329     \iow_now:Nn \@mainaux \ExplSyntaxOn
1330     \int_step_inline:nnnn \l_@@_first_env_block_int 1 \g_@@_env_int
1331     {
1332       \iow_now:Nx \@mainaux
1333       {
1334         \cs_gset:cpn { @@_max_cell_width_ ##1 }
1335         { \dim_use:N \g_@@_max_cell_width_dim }
1336       }
1337     }
1338     \iow_now:Nn \@mainaux \ExplSyntaxOff
1339   }
1340 }

```

## 15.12 The environment {pNiceArrayC} and its variants

The code in this section can be removed without affecting the previous code.

First, we define a set of options for the environment {pNiceArrayC} and its variants. This set of keys is named NiceMatrix/NiceArrayC even though there is no environment called {NiceArrayC}.

```

1341 \keys_define:nn { NiceMatrix / NiceArrayC }
1342 {
1343   code-for-last-col .tl_set:N = \l_@@_code_for_last_col_tl ,

```

```

1344     code-for-last-col .value_required:n = true ,
1345     unknown .code:n = \@@_error:n { Unknown~option~for~NiceArrayC }
1346 }
1347 \@@_msg_new:nnn { Unknown~option~for~NiceArrayC }
1348 {
1349     The~option~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~environment~
1350     \{\@currenvir\}. \\
1351     If~you~go~on,~it~will~be~ignored. \\
1352     For~a~list~of~the~available~options,~type~H~<return>.
1353 }
1354 {
1355     The~available~options~are~(in~alphabetic~order):~
1356     code~after,~
1357     code~for~last~col,~
1358     columns~width,~
1359     create~extra~nodes,~
1360     extra~left~margin,~
1361     extra~right~margin,~
1362     hlines,~
1363     left~margin,~
1364     name,~
1365     nullify~dots,~
1366     parallelize~diags~
1367     renew~dots~
1368     and~right~margin.
1369 }

```

In the environment `{pNiceArrayC}` (and its variants), the last column is composed with instructions `\hbox_overlap_right:n` (this instruction may be seen as the `expl3` equivalent of the classical command `\rlap`). After the composition of the array, an horizontal skip is inserted to compensate for these overlapping boxes.

The command `\@@_NiceArrayC:n` will be used in `{NiceArrayCwithDelims}` but also in the environment `{NiceArrayRCwithDelims}`.

```

1370 \cs_new_protected:Nn \@@_NiceArrayC:n
1371 {
1372     \bool_set_true:N \l_@@_exterior_column_bool
1373     \bool_gset_false:N \g_@@_exterior_column_found_bool
1374     \begin { NiceArray }

```

The beginning of the preamble is the argument of the environment `{pNiceArrayC}`.

```

1375     { #1

```

However, we add a last column with its own specification. For a cell in this last column, the first operation is to store the content of the cell in the box `\l_tmpa_box`. This is allowed in `expl3` with the construction `\hbox_set:Nw \l_tmpa_box ... \hbox_set_end:`.

```

1376         >
1377         {
1378             \bool_gset_true:N \g_@@_exterior_column_found_bool
1379             \int_gincr:N \g_@@_column_int
1380             \int_gset:Nn \g_@@_column_total_int
1381             { \int_max:nn \g_@@_column_total_int \g_@@_column_int }
1382             \hbox_set:Nw \l_tmpa_box \c_math_toggle_token
1383             \l_@@_code_for_last_col_tl
1384         }
1385     1

```

We actualize the value of `\g_@@_width_last_col_dim` which, at the end of the array, will contain the maximal width of the cells of the last column (thus, it will be equal to the width of the last column).

```

1386     < { \c_math_toggle_token
1387         \hbox_set_end:
1388         \dim_gset:Nn \g_@@_width_last_col_dim
1389     {

```

```

1390         \dim_max:nn
1391         \g_@@_width_last_col_dim
1392         { \box_wd:N \l_tmpa_box }
1393     }
1394     \skip_horizontal:n { - 2 \arraycolsep }
The content of the cell is inserted in an overlapping position.
1395     \hbox_overlap_right:n
1396     {
1397         \skip_horizontal:n
1398         {
1399             2 \arraycolsep +
1400             \l_@@_right_margin_dim +
1401             \l_@@_extra_right_margin_dim
1402         }
1403         \tikz
1404         [
1405             remember~picture ,
1406             inner~sep = \c_zero_dim ,
1407             minimum~width = \c_zero_dim ,
1408             baseline
1409         ]
1410         \node
1411         [
1412             anchor = base ,
1413             name =
1414             nm -
1415             \int_use:N \g_@@_env_int -
1416             \int_use:N \g_@@_row_int -
1417             \int_use:N \g_@@_column_int ,
1418             alias =
1419             \str_if_empty:NF \l_@@_name_str
1420             {
1421                 \l_@@_name_str -
1422                 \int_use:N \g_@@_row_int -
1423                 \int_use:N \g_@@_column_int
1424             }
1425         ]
1426         { \box_use:N \l_tmpa_box } ;
1427     }
1428 }
1429 }
1430 }

```

The environments of the type of `{pNiceArrayC}` will be constructed over `{NiceArrayCwithDelims}`. The first two arguments of this environment are the left and the right delimiter.

```

1431 \NewDocumentEnvironment { NiceArrayCwithDelims } { m m O { } m ! O { } }
1432 {
1433     \@@_test_if_math_mode:
1434     \dim_gzero_new:N \g_@@_width_last_col_dim
1435     \keys_set:nn { NiceMatrix / NiceArrayC } { #3 , #5 }
1436     \bool_set_false:N \l_@@_exterior_arraycolsep_bool
1437     \str_set:Nn \l_@@_pos_env_str c
1438     \left #1
1439     \@@_NiceArrayC:n { #4 }
1440 }
1441 {
1442     \end { NiceArray }
1443     \right #2
1444     \skip_horizontal:n \g_@@_width_last_col_dim
1445 }

```

In the following environments, we don't use the form with `\begin{...}` and `\end{...}` because we use `\@currenvir` in the error message for an unknown option.

```

1446 \NewDocumentEnvironment { pNiceArrayC } { }
1447   { \NiceArrayCwithDelims ( ) }
1448   { \endNiceArrayCwithDelims }

1449 \NewDocumentEnvironment { vNiceArrayC } { }
1450   { \NiceArrayCwithDelims | | }
1451   { \endNiceArrayCwithDelims }

1452 \NewDocumentEnvironment { VNiceArrayC } { }
1453   { \NiceArrayCwithDelims \l \r }
1454   { \endNiceArrayCwithDelims }

1455 \NewDocumentEnvironment { bNiceArrayC } { }
1456   { \NiceArrayCwithDelims [ ] }
1457   { \endNiceArrayCwithDelims }

1458 \NewDocumentEnvironment { BNiceArrayC } { }
1459   { \NiceArrayCwithDelims \{ \} }
1460   { \endNiceArrayCwithDelims }

```

### 15.13 The environment `{pNiceArrayRC}`

The code in this section can be removed without affecting the previous code.

```

1461 \keys_define:nn { NiceMatrix / NiceArrayRC }
1462   {
1463     code-for-first-row .tl_set:N = \l_@@_code_for_first_row_tl ,
1464     code-for-first-row .value_required:n = true ,
1465     code-for-last-col .tl_set:N = \l_@@_code_for_last_col_tl ,
1466     code-for-last-col .value_required:n = true ,
1467     unknown .code:n = \@_error:n { Unknown-option-for-NiceArrayRC }
1468   }

1469 \@@_msg_new:nnn { Unknown-option-for-NiceArrayRC }
1470   {
1471     The-option-'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~environment~
1472     \{ \@currenvir \}. \\
1473     If~you~go~on,~it~will~be~ignored. \\
1474     For~a~list~of~the~available~options,~type~H~<return>.
1475   }
1476   {
1477     The~available~options~are~(in~alphabetic~order):~
1478     code-after,~
1479     code-for-last-col,~
1480     code-for-first-row,~
1481     columns-width,~
1482     create-extra-nodes,~
1483     extra-left-margin,~
1484     extra-right-margin,~
1485     hlines,~
1486     left-margin,~
1487     name,~
1488     nullify-dots,~
1489     parallelize-diags,~
1490     renew-dots~
1491     and~right-margin.
1492   }

```

The first and the second argument of the environment `{NiceArrayRCwithDelims}` are the delimiters which will be used in the array. Usually, the final user will not use directly this environment `{NiceArrayRCwithDelims}` because he will use one of the variants `{pNiceArrayRC}`, `{vNiceArrayRC}`, etc.

We don't want that a vertical rule drawn by this specifier extends in the first row of the array (since this first row is for the labels and is "outside" the matrix).

The natural way to do that would be to redefine the specifier "|" with `\newcolumnntype`:

```
\newcolumnntype { | }
{ ! { \int_compare:nNnF \g_@@_row_int = \c_zero_int \vline } }
```

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

That's why we will do a redefinition of the macro `\@arrayrule` of `array` and this redefinition will add `\@@_vline`: instead of `\vline` to the preamble.

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 test must be effective in each row and not once when the preamble is constructed).

```
1493 \cs_new_protected:Npn \@@_vline:
1494 {
1495   \int_compare:nNnTF \g_@@_column_int = \c_zero_int
1496     { \int_compare:nNnF \g_@@_row_int < \c_zero_int \vline }
1497     { \int_compare:nNnF \g_@@_row_int < \c_one_int \vline }
1498 }

1499 \NewDocumentEnvironment { NiceArrayRCwithDelims } { m m O { } m ! O { } }
1500 {
1501   \@@_test_if_math_mode:
1502   \cs_set_protected:Npn \@arrayrule { \addtopreamble \@@_vline: }
1503   \int_zero:N \l_@@_nb_first_row_int
1504   \dim_gzero_new:N \g_@@_width_last_col_dim
1505   \keys_set:nn { NiceMatrix / NiceArrayRC } { #3 , #5 }
1506   \bool_set_false:N \l_@@_exterior_arraycolsep_bool
1507   \str_set:Nn \l_@@_pos_env_str c
1508   \box_clear_new:N \l_@@_the_array_box
1509   \hbox_set:Nw \l_@@_the_array_box
1510   \c_math_toggle_token
1511   \@@_NiceArrayC:n { #4 }
1512 }
1513 {
1514   \end { NiceArray }
1515   \c_math_toggle_token
1516   \hbox_set_end:
1517   \dim_set:Nn \l_tmpa_dim
1518     {
1519       (
1520         \dim_max:nn
1521           { 12 pt }
1522           { \g_@@_max_ht_row_one_dim + \g_@@_max_dp_row_zero_dim }
1523       )
1524       + \g_@@_max_ht_row_zero_dim - \g_@@_max_ht_row_one_dim
1525     }
1526   \hbox_set:Nn \l_tmpa_box
1527     {
1528       \c_math_toggle_token
1529       \left #1
1530       \vcenter
1531         {
1532           \skip_vertical:n { - \l_tmpa_dim }
1533           \box_use_drop:N \l_@@_the_array_box
1534         }
1535       \right #2
1536       \c_math_toggle_token
1537       \skip_horizontal:n \g_@@_width_last_col_dim
1538     }
1539   \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }
```



```

1540 \box_use_drop:N \l_tmpa_box
1541 }

```

In the following environments, we don't use the form with `\begin{...}` and `\end{...}` because we use `\@currentenv` in the error message for an unknown option.

```

1542 \NewDocumentEnvironment { pNiceArrayRC } { }
1543 { \NiceArrayRCwithDelims ( ) }
1544 { \endNiceArrayRCwithDelims }

1545 \NewDocumentEnvironment { bNiceArrayRC } { }
1546 { \NiceArrayRCwithDelims [ ] }
1547 { \endNiceArrayRCwithDelims }

1548 \NewDocumentEnvironment { vNiceArrayRC } { }
1549 { \NiceArrayRCwithDelims | | }
1550 { \endNiceArrayRCwithDelims }

1551 \NewDocumentEnvironment { VNiceArrayRC } { }
1552 { \NiceArrayRCwithDelims \| \| }
1553 { \endNiceArrayRCwithDelims }

1554 \NewDocumentEnvironment { BNiceArrayRC } { }
1555 { \NiceArrayRCwithDelims \{ \} }
1556 { \endNiceArrayRCwithDelims }

```

## 15.14 The extra nodes

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

```

1557 \cs_generate_variant:Nn \dim_min:nn { v n }
1558 \cs_generate_variant:Nn \dim_max:nn { v n }

```

For each row  $i$ , we compute two dimensions `l_@@_row\_texts1{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`.

```

1559 \cs_new_protected:Nn \@@_create_extra_nodes:
1560 {
1561   \begin { tikzpicture } [ remember-picture , overlay ]
1562     \int_step_variable:nnnNn \l_@@_nb_first_row_int 1 \g_@@_row_int \@@_i:
1563     {
1564       \dim_zero_new:c { l_@@_row\_@@_i: _min_dim }
1565       \dim_set_eq:cN { l_@@_row\_@@_i: _min_dim } \c_max_dim
1566       \dim_zero_new:c { l_@@_row\_@@_i: _max_dim }
1567       \dim_set:cn { l_@@_row\_@@_i: _max_dim } { - \c_max_dim }
1568     }
1569     \int_step_variable:nNn \g_@@_column_total_int \@@_j:
1570     {
1571       \dim_zero_new:c { l_@@_column\_@@_j: _min_dim }
1572       \dim_set_eq:cN { l_@@_column\_@@_j: _min_dim } \c_max_dim
1573       \dim_zero_new:c { l_@@_column\_@@_j: _max_dim }
1574       \dim_set:cn { l_@@_column\_@@_j: _max_dim } { - \c_max_dim }
1575     }

```

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

```

1576 \int_step_variable:nnNn \l_@@_nb_first_row_int \g_@@_row_int \@@_i:
1577 {
1578   \int_step_variable:nNn \g_@@_column_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.

```

1579         { \cs_if_exist:cT
1580           { 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`.

```

1581         {
1582           \tikz@parse@node \pgfutil@firstofone
1583             ( nm - \int_use:N \g_@@_env_int
1584               - \@@_i: - \@@_j: .south-west )
1585           \dim_set:cn { l_@@_row_ \@@_i: _min_dim }
1586             { \dim_min:vn { l_@@_row _ \@@_i: _min_dim } \pgf@y }
1587           \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
1588             {
1589               \dim_set:cn { l_@@_column _ \@@_j: _min_dim }
1590               { \dim_min:vn { l_@@_column _ \@@_j: _min_dim } \pgf@x }
1591             }

```

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`.

```

1592           \tikz@parse@node \pgfutil@firstofone
1593             ( nm - \int_use:N \g_@@_env_int - \@@_i: - \@@_j: .north-east )
1594           \dim_set:cn { l_@@_row _ \@@_i: _max_dim }
1595             { \dim_max:vn { l_@@_row _ \@@_i: _max_dim } \pgf@y }
1596           \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
1597             {
1598               \dim_set:cn { l_@@_column _ \@@_j: _max_dim }
1599               { \dim_max:vn { l_@@_column _ \@@_j: _max_dim } \pgf@x }
1600             }
1601         }
1602     }
1603 }

```

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`).

```

1604     \tikzset { name-suffix = -medium }
1605     \@@_create_nodes:

```

For “large nodes”, the eventual “first row” and “last column” (in environments like `{pNiceArrayRC}`) don't interfere. That's why the loop over the rows will start at 1 and the loop over the columns will stop at `\g_@@_column_int` (and not `\g_@@_column_total_int`).<sup>24</sup>

```

1606     \int_set:Nn \l_@@_nb_first_row_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`.

```

1607     \int_step_variable:nNn { \g_@@_row_int - 1 } \@@_i:
1608     {
1609       \dim_set:cn { l_@@_row _ \@@_i: _min _ dim }
1610       {
1611         (
1612           \dim_use:c { l_@@_row _ \@@_i: _min _ dim } +
1613           \dim_use:c { l_@@_row _ \int_eval:n { \@@_i: + 1 } _ max _ dim }
1614         )
1615         / 2
1616       }
1617       \dim_set_eq:cc { l_@@_row _ \int_eval:n { \@@_i: + 1 } _ max _ dim }
1618         { l_@@_row_ \@@_i: _min_dim }
1619     }
1620     \int_step_variable:nNn { \g_@@_column_int - 1 } \@@_j:
1621     {

```

<sup>24</sup>We recall that `\g_@@_column_total_int` is equal to `\g_@@_column_int` except if there is an exterior column. In this case, `\g_@@_column_total_int` is equal to `\g_@@_column_int + 1`.

```

1622     \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim }
1623     {
1624     (
1625     \dim_use:c
1626     { l_@@_column _ \@@_j: _ max _ dim } +
1627     \dim_use:c
1628     { l_@@_column _ \int_eval:n { \@@_j: + 1 } _ min _ dim }
1629     )
1630     / 2
1631     }
1632     \dim_set:eq:cc { l_@@_column _ \int_eval:n { \@@_j: + 1 } _ min _ dim }
1633     { l_@@_column _ \@@_j: _ max _ dim }
1634     }
1635     \dim_sub:cn
1636     { l_@@_column _ 1 _ min _ dim }
1637     \g_@@_left_margin_dim
1638     \dim_add:cn
1639     { l_@@_column _ \int_use:N \g_@@_column_int _ max _ dim }
1640     \g_@@_right_margin_dim

```

Now, we can actually create the “large nodes”.

```

1641     \tikzset { name~suffix = -large }
1642     \@@_create_nodes:
1643     \end{tikzpicture}

```

When used once, the command `\@@_create_extra_nodes:` must become no-op (in the current TeX group). That’s why we put a nullification of the command.

```

1644     \cs_set:Npn \@@_create_extra_nodes: { }

```

We can now compute the width of the array (used by `\hdottedline`).

```

1645     \begin { tikzpicture } [ remember~picture , overlay ]
1646     \tikz@parse@node \pgfutil@firstofone
1647     ( nm - \int_use:N \g_@@_env_int - 1 - 1 - large .north-west )
1648     \dim_gset:Nn \g_tmpa_dim \pgf@x
1649     \tikz@parse@node \pgfutil@firstofone
1650     ( nm - \int_use:N \g_@@_env_int - 1 -
1651     \int_use:N \g_@@_column_int - large .north-east )
1652     \dim_gset:Nn \g_tmpb_dim \pgf@x
1653     \end { tikzpicture }
1654     \iow_now:Nn \@mainaux \ExplSyntaxOn
1655     \iow_now:Nx \@mainaux
1656     {
1657     \cs_gset:cpn { @@_width _ \int_use:N \g_@@_env_int }
1658     { \dim_eval:n { \g_tmpb_dim - \g_tmpa_dim } }
1659     }
1660     \iow_now:Nn \@mainaux \ExplSyntaxOff
1661 }

```

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.

```

1662 \cs_new_protected:Nn \@@_create_nodes:
1663 {
1664     \int_step_variable:nnNn \l_@@_nb_first_row_int \g_@@_row_int \@@_i:
1665     {
1666         \int_step_variable:nnNn \g_@@_column_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.

```

1667 {
1668     \coordinate ( @@~south~west )
1669         at ( \dim_use:c { l_@@_column_ \@@_j: _min_dim } ,
1670             \dim_use:c { l_@@_row_ \@@_i: _min_dim } ) ;
1671     \coordinate ( @@~north~east )
1672         at ( \dim_use:c { l_@@_column_ \@@_j: _max_dim } ,
1673             \dim_use:c { l_@@_row_ \@@_i: _max_dim } ) ;

```

We can eventually draw the rectangular node for the cell ( $\backslash@@_i\text{-}\backslash@@_j$ ). 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`.

```

1674     \node
1675     [
1676         node~contents = { } ,
1677         fit = ( @@~south~west ) ( @@~north~east ) ,
1678         inner~sep = \c_zero_dim ,
1679         name = nm - \int_use:N \g_@@_env_int - \@@_i: - \@@_j: ,
1680         alias =
1681             \str_if_empty:NF \g_@@_name_str
1682             { \g_@@_name_str - \@@_i: - \@@_j: }
1683     ]
1684     ;
1685 }
1686 }

```

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  $n$ .

```

1687     \@@_seq_mapthread_function:NNN
1688     \g_@@_multicolumn_cells_seq
1689     \g_@@_multicolumn_sizes_seq
1690     \@@_node_for_multicolumn:nn
1691 }

1692 \cs_new_protected:Npn \@@_extract_coords: #1 - #2 \q_stop
1693 {
1694     \cs_set:Npn \@@_i: { #1 }
1695     \cs_set:Npn \@@_j: { #2 }
1696 }

```

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  $i\text{-}j$  and the second is the value of  $n$  (the length of the “multi-cell”).

```

1697 \cs_new_protected:Nn \@@_node_for_multicolumn:nn
1698 {
1699     \@@_extract_coords: #1 \q_stop
1700     \coordinate ( @@~south~west ) at
1701     (
1702         \dim_use:c { l_@@_column_ \@@_j: _min _ dim } ,
1703         \dim_use:c { l_@@_row_ \@@_i: _min _ dim }
1704     ) ;
1705     \coordinate ( @@~north~east ) at
1706     (
1707         \dim_use:c { l_@@_column_ \int_eval:n { \@@_j: + #2 - 1 } _ max _ dim } ,
1708         \dim_use:c { l_@@_row_ \@@_i: _ max _ dim }
1709     ) ;
1710     \node
1711     [
1712         node~contents = { } ,
1713         fit = ( @@~south~west ) ( @@~north~east ) ,
1714         inner~sep = \c_zero_dim ,
1715         name = nm - \int_use:N \g_@@_env_int - \@@_i: - \@@_j: ,
1716         alias =
1717             \str_if_empty:NF \g_@@_name_str { \g_@@_name_str - \@@_i: - \@@_j: }
1718     ]

```

```

1719     ;
1720 }

```

## 15.15 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` execute the code `\cs_set_eq:NN \env@matrix \NiceMatrix`.

Of course, the command `\NiceMatrix` must be defined before such an instruction is executed.

```

1721 \ProcessKeysOptions { NiceMatrix }

```

## 15.16 Code for `\seq_mapthread_function:NNN`

In `\@@_create_nodes:` (used twice in `\@@_create_extra_nodes:` to create the “medium nodes” and “large nodes”), we want to use `\seq_mapthread_function:NNN` which is in `l3candidates`). For security, we define a function `\@@_seq_mapthread_function:NNN`. We will delete the following code when `\seq_mapthread_function:NNN` will be in `l3seq`.

```

1722 \cs_new:Npn \@@_seq_mapthread_function:NNN #1 #2 #3
1723 {
1724   \group_begin:

```

In the group, we can use `\seq_pop:NN` safely.

```

1725   \int_step_inline:nn { \seq_count:N #1 }
1726   {
1727     \seq_pop:NN #1 \l_tmpa_tl
1728     \seq_pop:NN #2 \l_tmpb_tl
1729     \exp_args:NVV #3 \l_tmpa_tl \l_tmpb_tl
1730   }
1731   \group_end:
1732 }

```

```

1733 \cs_set_protected:Npn \@@_renew_matrix:
1734 {
1735   \RenewDocumentEnvironment { pmatrix } { } { }
1736   { \pNiceMatrix }
1737   { \endpNiceMatrix }
1738   \RenewDocumentEnvironment { vmatrix } { } { }
1739   { \vNiceMatrix }
1740   { \endvNiceMatrix }
1741   \RenewDocumentEnvironment { Vmatrix } { } { }
1742   { \VNiceMatrix }
1743   { \endVNiceMatrix }
1744   \RenewDocumentEnvironment { bmatrix } { } { }
1745   { \bNiceMatrix }
1746   { \endbNiceMatrix }
1747   \RenewDocumentEnvironment { Bmatrix } { } { }
1748   { \BNiceMatrix }
1749   { \endBNiceMatrix }
1750 }

```

# 16 History

## 16.1 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.

## 16.2 Changes between versions 1.1 and 1.2

New environment `{NiceArray}` with column types `L`, `C` and `R`.

## 16.3 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”. New names are in lowercase and hyphens (but backward compatibility is kept).

## 16.4 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.

## 16.5 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.

## 16.6 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”.

## 16.7 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>25</sup>, Tikz externalization is now deactivated in the environments of the extension `nicematrix`.<sup>26</sup>

## 16.8 Changes between version 2.1 and 2.1.2

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

## 16.9 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 column `C`), the cells in the column `C` 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`).

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

---

<sup>25</sup>cf. [tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package](https://tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package)

<sup>26</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`.

## 16.10 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 <https://www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end>

## 16.11 Changes between version 2.1.4 and 2.1.5

Compatibility with the classes `revtex4-1` and `revtex4-2`.

Option `allow-duplicate-names`.

## 16.12 Changes between version 2.1.5 and 2.2

Possibility to draw horizontal dotted lines to separate rows with the command `\hdottedline` (similar to the classical command `\hline` and the command `\hdashline` of `arydshln`).

Possibility to draw vertical dotted lines to separate columns with the specifier `“:”` in the preamble (similar to the classical specifier `“|”` and the specifier `“:”` of `arydshln`).

## 16.13 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`.

## 16.14 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.

# Index

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

	Symbols	
<code>@@</code> commands:		<code>\g_@@_code_after_tl</code> . . . . . 160, 546, 743, 744
<code>\@@_Cdots</code> . . . . .	552, 563, 1118	<code>\l_@@_code_for_first_row_tl</code> . 201, 324, 1463
<code>\@@_Cell:</code> . . . . .	92, 315, 400, 524, 525, 526	<code>\l_@@_code_for_last_col_tl</code> . . . . .
<code>\@@_Ddots</code> . . . . .	554, 565, 1130	. . . . . 117, 199, 1343, 1383, 1465
<code>\@@_Hdotsfor</code> . . . . .	558, 568, 1161, 1165	<code>\g_@@_column_int</code> 317, 318, 320, 356, 362,
<code>\@@_Hspace:</code> . . . . .	557, 1142	434, 440, 502, 543, 545, 547, 576, 686, 708,
<code>\@@_Iddots</code> . . . . .	555, 566, 1136	710, 767, 1110, 1154, 1157, 1171, 1197,
<code>\@@_Ldots</code> . . . . .	551, 562, 567, 1112	1379, 1381, 1417, 1423, 1495, 1620, 1639, 1651
<code>\@@_NiceArrayC:n</code> . . . . .	1370, 1439, 1511	<code>\g_@@_column_total_int</code> . . . 319, 320, 577,
<code>\@@_Vdots</code> . . . . .	553, 564, 1124	708, 766, 794, 1380, 1381, 1569, 1578, 1666
<code>\@@_actually_draw_Ldots:</code> . . .	852, 854, 1213	<code>\l_@@_columns_width_dim</code> . . . . . 60,
<code>\@@_adapt_S_column:</code> . . . . .	67, 83, 457	152, 208, 477, 479, 486, 522, 529, 530, 531
<code>\@@_add_to_empty_cells:</code> . . . . .		<code>\@@_create_extra_nodes:</code> . . . . .
. . . 1107, 1116, 1122, 1128, 1134, 1140, 1144		. . . . . 721, 809, 1212, 1267, 1559, 1644
<code>\@@_after_array:</code> . . . . .	464, 690	<code>\@@_create_nodes:</code> . . . . . 1605, 1642, 1662
<code>\@@_after_array_i:</code> . . . . .	693, 703	<code>\@@_ddots</code> . . . . . 1105, 1133
<code>\l_@@_auto_columns_width_bool</code> . . . . .		<code>\l_@@_ddots_int</code> . . . . . 714, 958, 959
. . . . . 116, 151, 473, 1317		<code>\l_@@_delta_x_one_dim</code> . . . . . 716, 961, 971
<code>\l_@@_block_auto_columns_width_bool</code> . .		<code>\l_@@_delta_x_two_dim</code> . . . . . 718, 998, 1008
. . . . . 467, 634, 1310, 1315, 1327		<code>\l_@@_delta_y_one_dim</code> . . . . . 717, 963, 971
<code>\@@_cdots</code> . . . . .	1103, 1121	<code>\l_@@_delta_y_two_dim</code> . . . . . 719, 1000, 1008
		<code>\@@_dotfill:</code> . . . . . 1236, 1238, 1258

\c_@@_draft_bool .....	10, 11, 32, 677, 1159, 1235, 1268	\l_@@_initial_open_bool ..	728, 753, 797, 808, 858, 867, 882, 891, 904, 917, 947, 984, 1035, 1070, 1179, 1186, 1194, 1211, 1221, 1274
@@_draw_Cdots:nn .....	875	@@_instruction_of_type:n .....	678, 680, 1114, 1120, 1126, 1132, 1138
@@_draw_Ddots:nn .....	940	@@_j: .....	1569, 1571, 1572, 1573, 1574, 1578, 1580, 1584, 1587, 1589, 1590, 1593, 1596, 1598, 1599, 1620, 1622, 1626, 1628, 1632, 1633, 1666, 1669, 1672, 1679, 1682, 1695, 1702, 1707, 1715, 1717
@@_draw_Hdotsfor:nnn .....	1169, 1177	\l_@@_l_dim .....	1017, 1018, 1033, 1040, 1044, 1051, 1055, 1061, 1066, 1078, 1085, 1086
@@_draw_Iddots:nn .....	978	\g_@@_last_vdotted_col_int	543, 545, 578, 579
@@_draw_Ldots:nn .....	849	@@_ldots .....	1102, 1115
@@_draw_Vdots:nn .....	898	\g_@@_left_margin_dim .....	124, 492, 1637
@@_draw_tikz_line: .....	873, 895, 937, 975, 1012, 1015, 1232, 1300	\l_@@_left_margin_dim	122, 140, 492, 583, 1256
\g_@@_empty_cells_seq .....	571, 659, 1109	\l_@@_letter_for_dotted_lines_str .....	212, 537, 1306
@@_end_Cell: ....	94, 326, 404, 524, 525, 526	@@_line:nn .....	742, 1215
\g_@@_env_int .....	59, 354, 432, 466, 476, 480, 639, 656, 664, 739, 835, 841, 1249, 1250, 1287, 1324, 1330, 1415, 1580, 1583, 1593, 1647, 1650, 1657, 1679, 1715	\g_@@_lines_to_draw_tl .....	465, 682, 722, 730, 732, 1167
@@_error:n .....	17, 192, 196, 207, 215, 246, 288, 452, 460, 694, 1265, 1345, 1467	\g_@@_max_cell_width_dim .....	330, 331, 468, 640, 647, 1316, 1335
@@_error:nn .....	18, 156	\g_@@_max_dp_row_zero_dim	334, 335, 469, 1522
\l_@@_exterior_arraycolsep_bool .....	111, 203, 373, 581, 597, 1436, 1506	\g_@@_max_ht_row_one_dim .....	341, 342, 471, 1522, 1524
\l_@@_exterior_column_bool ..	64, 709, 1372	\g_@@_max_ht_row_zero_dim	336, 337, 470, 1524
\g_@@_exterior_column_found_bool .....	65, 709, 1373, 1378	@@_msg_new:nn .....	19, 30, 99, 104, 238, 382, 388, 696, 1304
\l_@@_extra_left_margin_dim ..	126, 144, 584	@@_msg_new:nnn .....	20, 216, 247, 269, 290, 1347, 1469
\g_@@_extra_nodes_bool .....	121, 491, 542, 721, 1248	@@_msg_redirect_name:nn .....	21, 210
\l_@@_extra_nodes_bool .....	120, 138, 491	@@_multicolumn:nnn .....	559, 1148
\g_@@_extra_right_margin_dim .....	128, 379, 494, 600	\g_@@_multicolumn_cells_seq .....	572, 1153, 1587, 1596, 1688
\l_@@_extra_right_margin_dim .....	127, 145, 494, 1401	\g_@@_multicolumn_sizes_seq	573, 1155, 1689
@@_extract_coords: .....	1692, 1699	\g_@@_name_str .....	118, 495, 642, 646, 1681, 1682, 1717
\l_@@_final_i_int .....	726, 751, 758, 763, 764, 772, 777, 817, 842, 1182, 1204	\l_@@_name_str .....	119, 158, 358, 360, 436, 438, 482, 484, 487, 495, 1419, 1421
\l_@@_final_j_int ...	727, 752, 759, 765, 766, 767, 773, 777, 818, 843, 1199, 1205, 1207	\g_@@_names_seq .....	61, 155, 157, 281
\l_@@_final_open_bool .....	729, 754, 771, 808, 863, 869, 887, 893, 909, 922, 952, 989, 1037, 1048, 1071, 1072, 1180, 1200, 1208, 1211, 1222, 1275	\l_@@_nb_first_row_int .....	62, 63, 575, 763, 788, 1503, 1562, 1576, 1606, 1664
@@_find_extremities_of_line:nnnn ..	747, 851, 877, 900, 942, 980	@@_node_for_multicolumn:nn ....	1690, 1697
\l_@@_first_env_block_int	1323, 1324, 1330	\l_@@_nullify_dots_bool .....	115, 136, 1115, 1121, 1127, 1133, 1139
@@_hdottedline: .....	556, 1244	@@_old_multicolumn .....	1147, 1150
\l_@@_hlines_bool .....	114, 131, 503	\l_@@_parallelize_diags_bool .....	112, 113, 132, 712, 956, 993
@@_i: .....	1562, 1564, 1565, 1566, 1567, 1576, 1580, 1584, 1585, 1586, 1587, 1593, 1594, 1595, 1596, 1607, 1609, 1612, 1613, 1617, 1618, 1664, 1670, 1673, 1679, 1682, 1694, 1703, 1708, 1715, 1717	\l_@@_pos_env_str .....	109, 110, 285, 286, 287, 372, 594, 1437, 1507
@@_iddots .....	1106, 1139	@@_renew_NC@rewrite@S: .....	85, 570
\l_@@_iddots_int .....	715, 995, 996	\l_@@_renew_dots_bool .....	134, 198, 560
@@_if_not_empty_cell:nn .....	653	@@_renew_matrix: ....	190, 193, 197, 1733
@@_if_not_empty_cell:nnTF .....	777, 803, 1190, 1204	@@_renewcolumntype:nn .....	394, 535, 536
@@_if_yet_drawn: .....	811	@@_retrieve_coords:nn .....	827, 848, 856, 880, 902, 915, 945, 982
@@_if_yet_drawn:TF ..	852, 878, 901, 943, 981	\c_@@_revtex_bool .....	23, 25, 28, 585
\l_@@_in_NiceArray_bool .....	66, 459, 461	\g_@@_right_margin_dim .....	125, 379, 493, 599, 1256, 1640
\l_@@_initial_i_int .....	724, 749, 784, 788, 790, 798, 804, 815, 836, 1181, 1190	\l_@@_right_margin_dim ..	123, 142, 493, 1400
\l_@@_initial_j_int .....	725, 750, 785, 792, 794, 799, 804, 816, 837, 1185, 1191, 1193	\g_@@_row_int .....	318, 323, 332, 339, 355, 361, 433, 439, 505, 574, 575, 685, 692,



764, 790, 1110, 1154, 1170, 1282, 1294,  
1416, 1422, 1496, 1497, 1562, 1576, 1607, 1664  
`\@@_seq_mapthread_function:NNN` . 1687, 1722  
`\c_@@_siunitx_loaded_bool` . . . . . 69, 570  
`\l_@@_stop_loop_bool` . . . . .  
. . . . . 755, 756, 774, 778, 781, 782, 800, 805  
`\c_@@_table_collect_begin_tl` . . . . . 77, 79, 92  
`\c_@@_table_print_tl` . . . . . 80, 81, 94  
`\@@_test_if_math_mode:` . . . . .  
. . . . . 449, 458, 604, 610, 616, 622, 628, 1433, 1501  
`\l_@@_the_array_box` . . . . . 1508, 1509, 1533  
`\@@_vdots` . . . . . 1104, 1127  
`\@@_vdottedline:n` . . . . . 547, 1262  
`\@@_vline:` . . . . . 1493, 1502  
`\g_@@_width_last_col_dim` . . . . .  
. . . . . 1388, 1391, 1434, 1444, 1504, 1537  
`\@@_write_max_cell_width:` . . . . . 475, 632  
`\g_@@_x_final_dim` . . . . . 831, 844, 914,  
927, 933, 935, 962, 970, 999, 1007, 1023,  
1060, 1076, 1219, 1229, 1272, 1283, 1295, 1298  
`\g_@@_x_initial_dim` . . . . .  
. . . . . 829, 838, 914, 927, 930, 933, 935, 962,  
970, 999, 1007, 1024, 1060, 1074, 1076,  
1093, 1096, 1217, 1226, 1270, 1279, 1292, 1297  
`\g_@@_y_final_dim` . . . . . 832, 845,  
868, 870, 872, 892, 894, 964, 967, 1001,  
1004, 1027, 1065, 1083, 1220, 1230, 1273, 1284  
`\g_@@_y_initial_dim` . . . . .  
. . . . . 830, 839, 868, 870, 871,  
892, 894, 964, 969, 1001, 1006, 1028, 1065,  
1081, 1083, 1093, 1097, 1218, 1227, 1271, 1280  
`\g_@@_yet_drawn_seq` . . . . . 711, 820, 823  
`\` . . . . . 101, 106, 219, 220, 250, 251, 274, 275,  
293, 294, 700, 1307, 1350, 1351, 1472, 1473  
`\{` . . . . . 250, 293, 384,  
390, 617, 698, 1307, 1350, 1459, 1472, 1555  
`\}` . . . . . 250, 293, 384,  
390, 619, 698, 1307, 1350, 1459, 1472, 1555  
`\_` . . . . . 391

## A

`\array` . . . . . 593  
`\arraycolsep` . . . . . 141, 143,  
379, 582, 598, 1255, 1297, 1298, 1394, 1399  
`\arrayrulewidth` . . . . . 507, 508  
`\AtBeginDocument` . . . . . 34

## B

`\begin` . . . . . 605, 611, 617, 623, 629, 662,  
833, 1089, 1223, 1276, 1289, 1374, 1561, 1645  
`\bgroup` . . . . . 365  
`\BNiceMatrix` . . . . . 1748  
`\bNiceMatrix` . . . . . 1745  
bool commands:  
`\bool_do_until:Nn` . . . . . 756, 782  
`\bool_gset_eq:NN` . . . . . 491  
`\bool_gset_false:N` . . . . . 1373  
`\bool_gset_true:N` . . . . . 542, 1248, 1378  
`\bool_if:NTF` . . . . . 32,  
69, 459, 467, 473, 503, 560, 570, 581, 585,  
597, 634, 677, 712, 721, 858, 863, 867,  
869, 882, 887, 891, 893, 904, 909, 917,  
922, 928, 932, 947, 952, 956, 984, 989, 993,

1035, 1037, 1048, 1070, 1071, 1072, 1115,  
1121, 1127, 1133, 1139, 1159, 1235, 1268, 1327  
`\bool_if:nTF` . . . . . 709, 760,  
786, 808, 1114, 1120, 1126, 1132, 1138, 1211  
`\bool_new:N` . . . . . 10, 23, 41, 64,  
65, 66, 111, 112, 114, 115, 116, 120, 121, 1310  
`\bool_set:Nn` . . . . . 913, 926  
`\bool_set_false:N` . . . . . 373, 728, 729, 753, 754,  
755, 781, 1179, 1180, 1221, 1222, 1436, 1506  
`\bool_set_true:N` . . . . .  
. . . . . 11, 25, 28, 43, 113, 151, 198,  
461, 771, 774, 778, 797, 800, 805, 1186,  
1194, 1200, 1208, 1274, 1275, 1315, 1317, 1372  
`\l_tmpa_bool` . . . . . 913, 932  
`\l_tmpb_bool` . . . . . 926, 928  
box commands:  
`\box_clear_new:N` . . . . . 1508  
`\box_dp:N` . . . . . 335, 409  
`\box_ht:N` . . . . . 337, 342, 1539  
`\box_move_down:nn` . . . . . 410  
`\box_move_up:nn` . . . . . 424  
`\box_set_ht:Nn` . . . . . 1539  
`\box_use:N` . . . . . 366, 408, 424, 1426  
`\box_use_drop:N` . . . . . 1533, 1540  
`\box_wd:N` . . . . . 331, 414, 1392  
`\l_tmpa_box` . . . . . 321, 331, 335, 337, 342, 366,  
399, 408, 1382, 1392, 1426, 1526, 1539, 1540  
`\l_tmpb_box` . . . . . 407, 409, 414, 424

## C

`\Cdots` . . . . . 552  
`\cdots` . . . . . 563, 1103  
`\cleaders` . . . . . 1240  
clist commands:  
`\clist_map_inline:nn` . . . . . 36  
`\coordinate` . . . . . 418, 423, 1668, 1671, 1700, 1705  
cs commands:  
`\cs_generate_variant:Nn` . . . . .  
. . . . . 381, 848, 1234, 1557, 1558  
`\cs_gset:Npn` . . . . . 639, 646, 1334, 1657  
`\cs_gset_eq:NN` . . . . . 83  
`\cs_if_exist:NTF` . . . . . 462, 706, 1249, 1286, 1579  
`\cs_if_free:NTF` . . . . . 476, 484, 655  
`\cs_new:Npn` . . . . . 1148, 1244, 1722  
`\cs_new_protected:Nn` . . . . . 17,  
18, 315, 326, 394, 632, 690, 703, 747, 827,  
849, 854, 875, 898, 940, 978, 1015, 1107,  
1142, 1177, 1215, 1262, 1370, 1559, 1662, 1697  
`\cs_new_protected:Npn` . . . . .  
. . . . . 19, 20, 21, 85, 449, 680, 1493, 1692  
`\cs_set:Npn` . . . . .  
. . . . . 496, 514, 590, 1238, 1644, 1694, 1695  
`\cs_set_eq:NN` . . . . . 73, 533, 534, 536, 551, 552,  
553, 554, 555, 556, 557, 558, 559, 562, 563,  
564, 565, 566, 567, 568, 580, 587, 588, 589,  
742, 1102, 1103, 1104, 1105, 1106, 1147, 1236  
`\cs_set_protected:Npn` . . . . . 67, 678, 1502, 1733

## D

`\Ddots` . . . . . 554  
`\ddots` . . . . . 565, 1105  
`\DeclareOption` . . . . . 11, 12  
dim commands:  
`\dim_abs:n` . . . . . 671

<code>\dim_add:Nn</code> .....	1638	<code>\exp_args:Nx</code> .....	537
<code>\dim_compare:nNnTF</code> .....	522, 670, 1033	<code>\exp_not:N</code> .....	684
<code>\dim_compare_p:nNn</code> .....	914, 927	<code>\ExplSyntaxOff</code> .....	650, 1338, 1660
<code>\dim_eval:n</code> .....	1658	<code>\ExplSyntaxOn</code> .....	636, 1329, 1654
<code>\dim_gadd:Nn</code> .....		<b>F</b>	
871, 872, 1074, 1081, 1096, 1097, 1297, 1298		<code>\fi</code> .....	453
<code>\dim_gset:Nn</code> .....	330, 334,	fp commands:	
336, 341, 666, 668, 838, 839, 844, 845, 930,		<code>\fp_to_dim:n</code> .....	1020
967, 1004, 1226, 1227, 1229, 1230, 1279,		<b>G</b>	
1280, 1283, 1284, 1292, 1295, 1388, 1648, 1652		group commands:	
<code>\dim_gset_eq:NN</code> .....		<code>\group_begin:</code> .....	71, 705, 1724
492, 493, 494, 868, 870, 892, 894, 935		<code>\group_end:</code> .....	76, 745, 1731
<code>\dim_gzero_new:N</code> .....	468, 469,	<code>\group_insert_after:N</code> .....	464, 475
470, 471, 829, 830, 831, 832, 1316, 1434, 1504		<b>H</b>	
<code>\dim_max:nn</code> .....	331, 335,	<code>\halign</code> .....	518, 520
337, 342, 932, 1390, 1520, 1558, 1595, 1599		hbox commands:	
<code>\dim_min:nn</code> .....	932, 1557, 1586, 1590	<code>\hbox:n</code> .....	52, 54, 56, 420
<code>\dim_new:N</code> 60, 122, 123, 124, 125, 126, 127, 128		<code>\hbox_overlap_right:n</code> .....	1252, 1395
<code>\dim_ratio:nn</code> .....	971, 1008,	<code>\hbox_set:Nn</code> .....	407, 1526
1040, 1044, 1051, 1055, 1061, 1066, 1077, 1084		<code>\hbox_set:Nw</code> .....	321, 399, 1382, 1509
<code>\dim_set:Nn</code> .....	152, 208,	<code>\hbox_set_end:</code> .....	329, 405, 1387, 1516
381, 409, 479, 486, 961, 963, 998, 1000,		<code>\hbox_to_wd:nn</code> .....	414, 1240, 1253
1018, 1058, 1063, 1234, 1250, 1251, 1517,		<code>\Hdotsfor</code> .....	558
1567, 1574, 1585, 1589, 1594, 1598, 1609, 1622		<code>\hdotsfor</code> .....	568
<code>\dim_set_eq:NN</code> .....	1565, 1572, 1617, 1632	<code>\hdottedline</code> .....	556
<code>\dim_sub:Nn</code> .....	1635	<code>\hfil</code> .....	416, 536
<code>\dim_use:N</code> .....		<code>\hfill</code> .....	1240
529, 530, 531, 640, 647, 1023, 1024,		<code>\hrule</code> .....	507
1027, 1028, 1335, 1612, 1613, 1625, 1627,		<code>\Hspace</code> .....	557
1669, 1670, 1672, 1673, 1702, 1703, 1707, 1708		<code>\hspace</code> .....	1145
<code>\dim_zero:N</code> .....	477	<code>\hss</code> .....	536, 1240
<code>\dim_zero_new:N</code> .....	716, 717,	<b>I</b>	
718, 719, 1017, 1217, 1218, 1219, 1220,		<code>\ialign</code> .....	496, 514
1270, 1271, 1272, 1273, 1564, 1566, 1571, 1573		<code>\Iddots</code> .....	555
<code>\c_max_dim</code> .....	1565, 1567, 1572, 1574	<code>\iddots</code> .....	47, 566, 1106
<code>\g_tmpa_dim</code> .....	666, 671, 1648, 1658	<code>\ifmmode</code> .....	451
<code>\l_tmpa_dim</code> .....	409, 410, 424,	int commands:	
1058, 1096, 1250, 1251, 1255, 1517, 1532, 1539		<code>\int_add:Nn</code> .....	758, 759, 798, 799
<code>\g_tmpb_dim</code> .....	668, 671, 1652, 1658	<code>\int_compare:nNnTF</code> .....	
<code>\l_tmpb_dim</code> .....	1063, 1097	318, 323, 332, 339, 505, 543, 692, 959,	
<code>\c_zero_dim</code> .....	347, 348,	996, 1151, 1183, 1197, 1264, 1495, 1496, 1497	
442, 522, 1033, 1241, 1406, 1407, 1678, 1714		<code>\int_compare_p:nNn</code> .....	762,
<code>\dots</code> .....	567	764, 765, 766, 767, 768, 788, 790, 792, 794	
<b>E</b>		<code>\int_eval:n</code> 1154, 1613, 1617, 1628, 1632, 1707	
<code>\egroup</code> .....	367	<code>\int_gadd:Nn</code> .....	1157
<code>\else</code> .....	451	<code>\int_gdecr:N</code> .....	710
<code>\end</code> .....	607, 613, 619, 625, 631, 669, 846,	<code>\int_gincr:N</code> .....	317, 318, 466, 1379
1099, 1231, 1285, 1296, 1442, 1514, 1643, 1653		<code>\int_gset:Nn</code> .....	319, 575, 579, 1380
<code>\endarray</code> .....	377, 596	<code>\int_gset_eq:NN</code> .....	545, 708
<code>\endBNiceMatrix</code> .....	1749	<code>\int_gzero:N</code> .....	502
<code>\endbNiceMatrix</code> .....	1746	<code>\int_gzero_new:N</code> .....	574, 576, 577, 578
<code>\endNiceArrayCwithDelims</code> .....		<code>\int_incr:N</code> .....	958, 995
1448, 1451, 1454, 1457, 1460		<code>\int_max:nn</code> .....	320, 1381
<code>\endNiceArrayRCwithDelims</code> .....		<code>\int_new:N</code> .....	59, 62
1544, 1547, 1550, 1553, 1556		<code>\int_set:Nn</code> .....	
<code>\endpNiceMatrix</code> .....	1737	63, 749, 750, 751, 752, 1039, 1043,	
<code>\endVNiceMatrix</code> .....	1743	1050, 1054, 1068, 1181, 1182, 1185, 1189,	
<code>\endvNiceMatrix</code> .....	1740	1191, 1193, 1199, 1203, 1205, 1207, 1324, 1606	
<code>\everycr</code> .....	498, 516	<code>\int_step_inline:nn</code> .....	1725
exp commands:		<code>\int_step_inline:nnnn</code> .....	1090, 1330
<code>\exp_after:wN</code> .....	89		
<code>\exp_args:NNo</code> .....	75		
<code>\exp_args:NVV</code> .....	1729		

<code>\int_step_variable:nNn</code> .....	1569, 1578, 1607, 1620, 1666
<code>\int_step_variable:nnNn</code> .....	1576, 1664
<code>\int_step_variable:nnnNn</code> .....	1562
<code>\int_sub:Nn</code> .....	772, 773, 784, 785
<code>\int_use:N</code> .. 354, 355, 356, 361, 362, 432, 433, 434, 439, 440, 476, 480, 547, 639, 656, 659, 664, 685, 686, 739, 815, 816, 817, 818, 835, 836, 837, 841, 842, 843, 1110, 1154, 1170, 1171, 1249, 1250, 1282, 1287, 1294, 1415, 1416, 1417, 1422, 1423, 1580, 1583, 1593, 1639, 1647, 1650, 1651, 1657, 1679, 1715	
<code>\int_zero:N</code> .....	1503
<code>\int_zero_new:N</code> .....	714, 715, 724, 725, 726, 727, 1323
<code>\c_one_int</code> .....	339, 765, 851, 877, 900, 942, 959, 996, 1497
<code>\l_tmpa_int</code> .....	1039, 1043, 1050, 1054, 1078, 1085, 1090, 1189, 1190, 1203, 1204
<code>\l_tmpb_int</code> .....	1068, 1079, 1087
<code>\c_zero_int</code> .....	323, 332, 692, 768, 851, 877, 900, 1264, 1495, 1496
iow commands:	
<code>\iow_now:Nn</code> .....	636, 637, 644, 650, 1329, 1332, 1338, 1654, 1655, 1660
<b>K</b>	
<code>\kern</code> .....	56
keys commands:	
<code>\keys_define:nn</code> .....	129, 147, 164, 188, 245, 283, 1311, 1341, 1461
<code>\l_keys_key_tl</code> ....	218, 249, 292, 1349, 1471
<code>\keys_set:nn</code> ..	244, 371, 472, 1322, 1435, 1505
<code>\l_keys_value_tl</code> .....	271
<b>L</b>	
<code>\Ldots</code> .....	551
<code>\ldots</code> .....	562, 1102
<code>\left</code> .....	605, 611, 617, 623, 629, 1438, 1529
<code>\line</code> .....	742
<code>\lVert</code> .....	629
<code>\lvert</code> .....	623
<b>M</b>	
<code>\makebox</code> .....	408
math commands:	
<code>\c_math_toggle_token</code> .....	322, 328, 1382, 1386, 1510, 1515, 1528, 1536
<code>\mathinner</code> .....	49
<code>\mkern</code> .....	51, 53, 55, 56
msg commands:	
<code>\msg_error:nn</code> .....	17, 18
<code>\msg_new:nnn</code> .....	19
<code>\msg_new:nnnn</code> .....	20
<code>\msg_redirect_name:nnn</code> .....	22
<code>\msg_warning:nn</code> .....	33
<code>\multicolumn</code> .....	559, 1147
<code>\myfiledate</code> .....	7
<code>\myfileversion</code> .....	8
<b>N</b>	
<code>\newcolumntype</code> .....	396, 524, 525, 526, 529, 530, 531, 537
<code>\NewDocumentCommand</code> .....	243, 1112, 1118, 1124, 1130, 1136, 1161, 1165
<code>\NewDocumentEnvironment</code> .....	369, 455, 602, 608, 614, 620, 626, 1320, 1431, 1446, 1449, 1452, 1455, 1458, 1499, 1542, 1545, 1548, 1551, 1554
<code>\NiceArray</code> .....	374
<code>\NiceArrayCwithDelims</code> .....	1447, 1450, 1453, 1456, 1459
<code>\NiceArrayRCwithDelims</code> .....	1543, 1546, 1549, 1552, 1555
<code>\NiceMatrixOptions</code> .....	219, 243
<code>\noalign</code> .....	500, 1246
<code>\node</code> .....	351, 429, 1410, 1674, 1710
<b>P</b>	
<code>\path</code> .....	1224
<code>\pgfpathcircle</code> .....	1092
<code>\pgfpoint</code> .....	1093
<code>\pgfpointanchor</code> .....	665, 667
<code>\pgfusepath</code> .....	1095
<code>\phantom</code> .....	1115, 1121, 1127, 1133, 1139
<code>\pNiceMatrix</code> .....	1736
prg commands:	
<code>\prg_do_nothing:</code> .....	73, 83, 1236
<code>\prg_replicate:nn</code> .....	1162, 1174
<code>\prg_return_false:</code> .....	657, 660, 672, 824
<code>\prg_return_true:</code> .....	673, 821
<code>\prg_set_conditional:Npnn</code> .....	653, 811
<code>\ProcessKeysOptions</code> .....	1721
<code>\ProcessOptions</code> .....	13
<code>\ProvideDocumentCommand</code> .....	47
<code>\ProvidesExplPackage</code> .....	5
<b>Q</b>	
quark commands:	
<code>\q_stop</code> .....	1692, 1699
<b>R</b>	
<code>\raise</code> .....	52, 54, 56
<code>\relax</code> .....	13, 533, 534
<code>\renewcommand</code> .....	87
<code>\RenewDocumentEnvironment</code> .....	1735, 1738, 1741, 1744, 1747
<code>\RequirePackage</code> .....	1, 3, 4, 14, 15, 16
<code>\right</code> .....	607, 613, 619, 625, 631, 1443, 1535
<code>\rVert</code> .....	631
<code>\rvert</code> .....	625
<b>S</b>	
seq commands:	
<code>\seq_count:N</code> .....	1725
<code>\seq_gclear_new:N</code> .....	571, 572, 573, 711
<code>\seq_gput_left:Nn</code> ....	157, 823, 1153, 1155
<code>\seq_gput_right:Nn</code> .....	1109
<code>\seq_if_in:NnTF</code> ...	155, 659, 820, 1587, 1596
<code>\seq_new:N</code> .....	61
<code>\seq_pop:NN</code> .....	1727, 1728
<code>\seq_use:Nnnn</code> .....	281
skip commands:	
<code>\skip_horizontal:n</code> 378, 541, 582, 583, 584, 598, 599, 600, 1241, 1394, 1397, 1444, 1537	
<code>\skip_vertical:n</code> .....	508, 1532
<code>\c_zero_skip</code> .....	513, 517



<b>7</b>	<b>The environment <code>{pNiceArrayC}</code> and its variants</b>	<b>8</b>
<b>8</b>	<b>The environment <code>{pNiceArrayRC}</code> and its variants</b>	<b>9</b>
<b>9</b>	<b>The dotted lines to separate rows or columns</b>	<b>9</b>
<b>10</b>	<b>The width of the columns</b>	<b>10</b>
<b>11</b>	<b>The option <code>hlines</code></b>	<b>11</b>
<b>12</b>	<b>Utilisation of the column type <code>S</code> of <code>siunitx</code></b>	<b>11</b>
<b>13</b>	<b>Technical remarks</b>	<b>12</b>
	13.1 Diagonal lines . . . . .	12
	13.2 The “empty” cells . . . . .	12
	13.3 The option <code>exterior-arraycolsep</code> . . . . .	13
	13.4 The class option <code>draft</code> . . . . .	13
	13.5 A technical problem with the argument of <code>\</code> . . . . .	13
<b>14</b>	<b>Examples</b>	<b>14</b>
	14.1 Dotted lines . . . . .	14
	14.2 Width of the columns . . . . .	16
	14.3 How to highlight cells of the matrix . . . . .	16
	14.4 Block matrices . . . . .	19
<b>15</b>	<b>Implementation</b>	<b>19</b>
	15.1 Declaration of the package and extensions loaded . . . . .	19
	15.2 Technical definitions . . . . .	20
	15.3 The column <code>S</code> of <code>siunitx</code> . . . . .	22
	15.4 The options . . . . .	23
	15.5 The environments <code>{NiceArray}</code> and <code>{NiceMatrix}</code> . . . . .	28
	15.6 After the construction of the array . . . . .	38
	15.7 The actual instructions for drawing the dotted line with <code>Tikz</code> . . . . .	45
	15.8 User commands available in the new environments . . . . .	47
	15.9 The command <code>\line</code> accessible in code-after . . . . .	49
	15.10The commands to draw dotted lines to separate columns and rows . . . . .	50
	15.11The environment <code>{NiceMatrixBlock}</code> . . . . .	52
	15.12The environment <code>{pNiceArrayC}</code> and its variants . . . . .	52
	15.13The environment <code>{pNiceArrayRC}</code> . . . . .	55
	15.14The extra nodes . . . . .	57
	15.15We process the options . . . . .	61
	15.16Code for <code>\seq_mapthread_function:NNN</code> . . . . .	61
<b>16</b>	<b>History</b>	<b>61</b>
	16.1 Changes between versions 1.0 and 1.1 . . . . .	61
	16.2 Changes between versions 1.1 and 1.2 . . . . .	62
	16.3 Changes between version 1.2 and 1.3 . . . . .	62
	16.4 Changes between version 1.3 and 1.4 . . . . .	62
	16.5 Changes between version 1.4 and 2.0 . . . . .	62
	16.6 Changes between version 2.0 and 2.1 . . . . .	62
	16.7 Changes between version 2.1 and 2.1.1 . . . . .	62
	16.8 Changes between version 2.1 and 2.1.2 . . . . .	62
	16.9 Changes between version 2.1.2 and 2.1.3 . . . . .	62
	16.10Changes between version 2.1.3 and 2.1.4 . . . . .	63
	16.11Changes between version 2.1.4 and 2.1.5 . . . . .	63
	16.12Changes between version 2.1.5 and 2.2 . . . . .	63
	16.13Changes between version 2.2 and 2.2.1 . . . . .	63
	16.14Changes between version 2.2.1 and 2.3 . . . . .	63

