

The package `nicematrix`*

F. Pantigny
fpantigny@wanadoo.fr

September 19, 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`. The final user only has to load the extension with `\usepackage{nicematrix}`.

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

- continuous dotted lines¹;
- exterior rows and columns for labels;
- a control of the width of the columns.

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

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 3.2 of `nicematrix`, at the date of 2019/09/19.
¹If the class option `draft` is used, these dotted lines will not be drawn for a faster compilation.

2 The environments of this extension

The extension `nicematrix` defines the following new environments.

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

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

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

3 The continuous dotted lines

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

Each of them must be used alone in the cell of the array and it draws a dotted line between the first non-empty cells⁴ on both sides of the current cell. Of course, for `\Ldots` and `\Cdots`, it's an horizontal line; for `\Vdots`, it's a vertical line and for `\Ddots` and `\Iddots` diagonal ones.

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

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

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

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

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

⁴The precise definition of a “non-empty cell” is given below (cf. p. 13).

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

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

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

In the first column of this example, there are two instructions `\Vdots` but only one dotted line is drawn (there is no overlapping graphic objects in the resulting PDF⁵).

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

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

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

There are also other means to change the size of the matrix. Someone might want to use the optional argument of the command `\` for the vertical dimension and a command `\hspace*` in a cell for the horizontal dimension.⁶

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

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

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

3.1 The option `nullify-dots`

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

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

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

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

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

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

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

⁶It's also possible to fix the width of a column by using the environment `{NiceArray}` (or one of its variants) with a column of type `w` or `W`: see p. 9

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

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

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

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

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

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

3.2 The command `\Hdotsfor`

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

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

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

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

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

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

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

3.3 How to generate the continuous dotted lines transparently

The package `nicematrix` provides an option called `transparent` for using existing code transparently in the environments `{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`³ 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}
```

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

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.⁷

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

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`.⁸

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

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

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

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

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

We explain below how to fill the nodes created by `nicematrix` (cf. p. 18).

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}
```

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

⁷There is no “large nodes” created in the exterior rows and columns (for these rows and columns, cf. p. 7).

⁸The options `left-margin` and `right-margin` take dimensions as values but, if no value is given, the default value is used, which is `\arraycolsep` (by default: 5 pt). There is also an option `margin` to fix both `left-margin` and `right-margin` to the same value.

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 R⁹ 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.¹⁰

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):

```


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


```

In fact, there is also variants for the environment `{NiceArray}`: `{pNiceArray}`, `{bNiceArray}`, `{BNiceArray}`, `{vNiceArray}` and `{VNiceArray}`.

In the following example, we use an environment `{pNiceArray}` (we don't use `{pNiceMatrix}` because we want to use the types L and R — in `{pNiceMatrix}`, all the columns are of type C).

```


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


```

With the environment `{NiceArray}` and its the variants, it's possible to compose exterior rows and columns with the options `first-row`, `last-row`, `first-col` and `last-col`.

There is no specification of column to provide for the potential “first column” (it will automatically be a R column) and for the potential “last column” (it will automatically be a L column).

```


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


```

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

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

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

However, there is a particularity for the option `last-row`: when LaTeX composes an array (with the TeX command `\halign`) it composes it row by row and there is no direct way to know if we are at the last row before the composition of that row. That's why `nicematrix` writes in the `aux` file the number of rows of the array in order to use it at the next run. Nevertheless, it's possible to give directly the number of rows as the value of the key `last-row`. In that way, `nicematrix` will know the correct value by the first compilation.

It's possible to control the appearance of these rows and columns with options `code-for-first-row`, `code-for-last-row`, `code-for-first-col` and `code-for-last-col`. These options specify tokens that will be inserted before each cell of the corresponding row or column.

```
\NiceMatrixOptions{code-for-first-row = \color{red},
                  code-for-first-col = \color{blue},
                  code-for-last-row = \color{green},
                  code-for-last-col = \color{magenta}}
$\begin{pNiceArray}{CC|CC}[first-row,last-row=5,first-col,last-col]
  & C_1 & & C_2 & & C_3 & & C_4 & & \\\
L_1 & a_{11} & & a_{12} & & a_{13} & & a_{14} & & L_1 \\\
L_2 & a_{21} & & a_{22} & & a_{23} & & a_{24} & & L_2 \\\
\hline
L_3 & a_{31} & & a_{32} & & a_{33} & & a_{34} & & L_3 \\\
L_4 & a_{41} & & a_{42} & & a_{43} & & a_{44} & & L_4 \\\
  & C_1 & & C_2 & & C_3 & & C_4 & & \\
\end{pNiceArray}$
```

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

Remarks

- As shown in the previous example, an horizontal rule (drawn by `\hline`) doesn't extend in the exterior columns and a vertical rule (specified by a `|` in the preamble of the array) doesn't extend in the exterior rows.¹¹
- The "first row" of an environment `{pNiceArray}` has the number 0, and not 1. Idem for the "first column". 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`).
- Logically, the potential option `columns-width` (described p. 9) doesn't apply to the "first column" and "last column".
- For technical reasons, it's not possible to use the option of the command `\|` after the "first row" or before the "last row" (the placement of the delimiters would be wrong).

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

```
$\begin{NiceArrayWithDelims}
  {\downarrow}{\downarrow}{CCC}[last-col]
1 & 2 & 3 & L_1 \\\
4 & 5 & 6 & L_2 \\\
7 & 8 & 9 & L_3 \\
\end{NiceArrayWithDelims}$
```

¹¹The latter is not true when the extension `arydshn` is loaded besides `nicematrix`. In fact, `nicematrix` and `arydshn` are not totally compatible because `arydshn` redefines many internals of `array`. On another note, if one really wants a vertical rule running in the first and in the last row, he should use `!\vline` instead of `|` in the preamble of the array.

7 The dotted lines to separate rows or columns

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

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

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

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

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

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

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

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

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

It's possible to change in `nicematrix` the letter used to specify a vertical dotted line with the option `letter-for-dotted-lines` available in `\NiceMatrixOptions`. For example, in this document, we have loaded the extension `arydshln` which uses the letter “:” to specify a vertical dashed line. Thus, by using `letter-for-dotted-lines`, we can use the vertical lines of both `arydshln` and `nicematrix`.

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

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

8 The width of the columns

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

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

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

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
 $\end{pNiceMatrix}$ 

```

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

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

It's possible to give the 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`.¹²

```

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

```

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

9 Block matrices

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

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

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

```

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

```

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

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

```

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

```

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

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

10 The option small

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

```

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

```

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

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

In fact, the option `small` corresponds to the following tuning:

- the cells of the array are composed with `\scriptstyle` ;
- `\arraystretch` is set to 0.47 ;
- `\arraycolsep` is set to 1.45 pt ;
- the characteristics of the dotted lines are also modified.

11 The option hlines

You can add horizontal rules between rows in the environments of `nicematrix` with the usual command `\hline`. But, by convenience, the extension `nicematrix` also provides the option `hlines`. With this option, all the horizontal rules will be drawn (excepted, of course, the rule before the potential “first row” and the rule after the potential “last row”).

```


$$\begin{NiceArray}{|*{4}{C|}}[hlines,first-row,first-col]
  e & a & b & c \\
e & e & a & b & c \\
a & a & e & c & b \\
b & b & c & e & a \\
c & c & b & a & e
\end{NiceArray}$$


```

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

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`. The `d` columns of the package `dcolumn` are not supported by `nicematrix`.

```


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


```

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

13 Technical remarks

13.1 Intersections of dotted lines

Since the version 3.1 of `nicematrix`, the dotted lines created by `\Cdots`, `\Ldots`, `\Vdots`, etc. can’t intersect.¹³

That means that a dotted line created by one these commands automatically stops when it arrives on a dotted line already drawn. Therefore, the order in which dotted lines are drawn is important. Here’s that order (by design) : `\Hdotsfor`, `\Vdots`, `\Ddots`, `\Iddots`, `\Cdots` and `\Ldots`.

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

```


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


```

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

¹³Of the contrary, dotted lines created by `\hdottedline`, the letter “:” in the preamble of the array and the command `\line` in the `code-after` can have intersections with other dotted lines.

13.2 Diagonal lines

By default, all the diagonal lines¹⁴ of a same array are “parallelized”. That means that the first diagonal line is drawn and, then, the other lines are drawn parallel to the first one (by rotation around the left-most extremity of the line). That’s why the position of the instructions `\Ddots` in the array can have a marked effect on the final result.

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

Example with parallelization (default):

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

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

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

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

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

The same example without parallelization:

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

13.3 The “empty” cells

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

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

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

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

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

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

¹⁴We speak of the lines created by `\Ddots` and not the lines created by a command `\line` in `code-after`.

13.4 The option `exterior-arraycolsep`

The environment `{array}` inserts an horizontal space equal to `\arraycolsep` before and after each column. In particular, there is a space equal to `\arraycolsep` before and after the array. This feature of the environment `{array}` was probably not a good idea.¹⁵

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`.

13.5 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 `:`).¹⁶

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

13.6 A technical problem with the argument of `\`

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

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

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

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

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

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

13.7 Obsolete environments

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

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

- `{NiceArrayCwithDelims}` ;
- `{pNiceArrayC}`, `{bNiceArrayC}`, `{BNiceArrayC}`, `{vNiceArrayC}`, `{VNiceArrayC}` ;

¹⁵In 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@{}}`.

¹⁶The 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.

- `{NiceArrayRCwithDelims}`;
- `{pNiceArrayRC}`, `{bNiceArrayRC}`, `{BNiceArrayRC}`, `{vNiceArrayRC}`, `{vNiceArrayRC}`.

They might be deleted in a future version of nicematrix.

14 Examples

14.1 Dotted lines

A tridiagonal matrix:

```


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


```

A permutation matrix:

```


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


```

An example with `\Iddots`:

```


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


```

An example with `\multicolumn`:

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

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

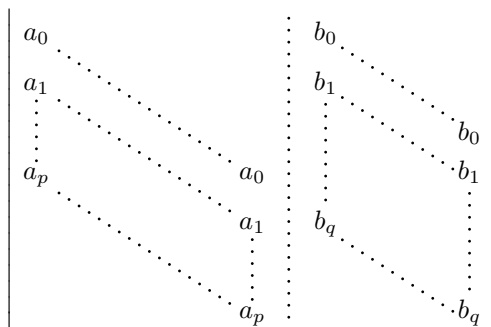
An example with `\Hdotsfor`:

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

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

An example for the resultant of two polynomials:

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



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

```

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

```

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

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

```

$$\left(\begin{array}{cccc|c}
1 & 1 & 1 & 1 & \dots & 1 \\
2 & 4 & 8 & 16 & \dots & 9 \\
3 & 9 & 27 & 81 & \dots & 36 \\
4 & 16 & 64 & 256 & \dots & 100 \\
\end{array} \right) \left| \begin{array}{cccc|c}
1 & 1 & 1 & 1 & \dots & 1 \\
0 & 1 & 3 & 7 & \dots & \frac{7}{2} \\
0 & 0 & 3 & 18 & \dots & 6 \\
0 & 0 & -2 & -14 & \dots & -\frac{9}{2} \\
\end{array} \right) \begin{array}{l} \\ \\ L_3 \leftarrow -3L_2 + L_3 \\ L_4 \leftarrow L_2 - L_4 \end{array}$$

$$\left(\begin{array}{cccc|c}
1 & 1 & 1 & 1 & \dots & 1 \\
0 & 2 & 6 & 14 & \dots & 7 \\
0 & 6 & 24 & 78 & \dots & 33 \\
0 & 12 & 60 & 252 & \dots & 96 \\
\end{array} \right) \begin{array}{l} L_2 \leftarrow -2L_1 + L_2 \\ L_3 \leftarrow -3L_1 + L_3 \\ L_4 \leftarrow -4L_1 + L_4 \end{array} \left| \begin{array}{cccc|c}
1 & 1 & 1 & 1 & \dots & 1 \\
0 & 1 & 3 & 7 & \dots & \frac{7}{2} \\
0 & 0 & 1 & 6 & \dots & 2 \\
0 & 0 & -2 & -14 & \dots & -\frac{9}{2} \\
\end{array} \right) \begin{array}{l} \\ \\ L_3 \leftarrow \frac{1}{3}L_3 \\ \end{array}$$

$$\left(\begin{array}{cccc|c}
1 & 1 & 1 & 1 & \dots & 1 \\
0 & 1 & 3 & 7 & \dots & \frac{7}{2} \\
0 & 3 & 12 & 39 & \dots & \frac{33}{2} \\
0 & 1 & 5 & 21 & \dots & 8 \\
\end{array} \right) \begin{array}{l} L_2 \leftarrow \frac{1}{2}L_2 \\ L_3 \leftarrow \frac{1}{2}L_3 \\ L_4 \leftarrow \frac{1}{12}L_4 \end{array} \left| \begin{array}{cccc|c}
1 & 1 & 1 & 1 & \dots & 1 \\
0 & 1 & 3 & 7 & \dots & \frac{7}{2} \\
0 & 0 & 1 & 6 & \dots & 2 \\
0 & 0 & 0 & -2 & \dots & -\frac{1}{2} \\
\end{array} \right) \begin{array}{l} \\ \\ \\ L_4 \leftarrow 2L_3 + L_4 \end{array}$$

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`.

```

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

```

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

The package `nicematrix` is constructed upon the environment `{array}` and, therefore, it's possible to use the package `colortbl` in the environments of `nicematrix`. However, it's not always easy to do a fine tuning of `colortbl`. That's why we propose another method to highlight a row of the matrix. We create a rectangular Tikz node which encompasses the nodes of the second row with the Tikz library `fit`. This Tikz node is filled after the construction of the matrix. In order to see the text *under* this node, we have to use transparency with the `blend mode` equal to `multiply`. 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}}

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

```

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

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

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

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

```
\tikzset{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.

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

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

14.4 Direct utilisation of the Tikz nodes

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

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

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

\NiceMatrixOptions{nullify-dots}
```

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

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

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

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

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

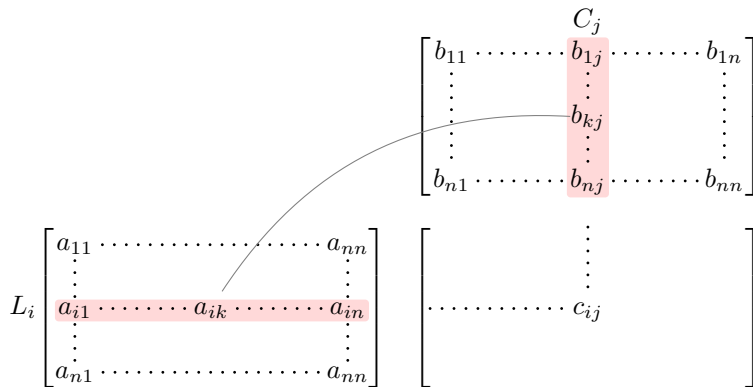
```
\begin{bNiceArray}{CC>{\strut}CCC}[name=A,first-col]
  & a_{11} & \Cdots & & a_{nn} \\
  & \Vdots & & & \Vdots \\
L_i & a_{i1} & \Cdots & a_{ik} & \Cdots & a_{in} \\
  & \Vdots & & & \Vdots \\
  & a_{n1} & \Cdots & & a_{nn} \\
\end{bNiceArray}
```

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

```
\begin{bNiceArray}{CC>{\strut}CCC}
  & & & \\
  & & & \\
\Cdots & & c_{ij} \\
\\
\\
\end{bNiceArray}
\end{array}$

\end{NiceMatrixBlock}
```

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



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.¹⁷

```
1 <@@=nm>
2 \RequirePackage{tikz}
3 \usetikzlibrary{fit}
4 \RequirePackage{expl3}[2019/02/15]
```

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

```
5 \RequirePackage{l3keys2e}
6 \ProvidesExplPackage
7   {nicematrix}
8   {\myfiledate}
9   {\myfileversion}
10  {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.

```
11 \bool_new:N \c__nm_draft_bool
12 \DeclareOption { draft } { \bool_set_true:N \c__nm_draft_bool }
13 \DeclareOption* { }
14 \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`.

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

18 \cs_new_protected:Npn \__nm_error:n { \msg_error:nn { nicematrix } }
19 \cs_new_protected:Npn \__nm_error:nn { \msg_error:nn { nicematrix } }
20 \cs_new_protected:Npn \__nm_error:nnn { \msg_error:nnn { nicematrix } }
21 \cs_new_protected:Npn \__nm_fatal:n { \msg_fatal:nn { nicematrix } }
22 \cs_new_protected:Npn \__nm_fatal:nn { \msg_fatal:nn { nicematrix } }
23 \cs_new_protected:Npn \__nm_msg_new:nn { \msg_new:nnn { nicematrix } }
24 \cs_new_protected:Npn \__nm_msg_new:nnn { \msg_new:nnnn { nicematrix } }

25 \cs_new_protected:Npn \__nm_msg_redirect_name:nn
26   { \msg_redirect_name:nnn { nicematrix } }
```

15.2 Technical definitions

We test whether the current class is `revtex4-1` or `revtex4-2` because these classes redefines `\array` (of `array`) in a way incompatible with our programming.

```
27 \bool_new:N \c__nm_revtex_bool
28 \@ifclassloaded { revtex4-1 }
29   { \bool_set_true:N \c__nm_revtex_bool }
30   { }
31 \@ifclassloaded { revtex4-2 }
32   { \bool_set_true:N \c__nm_revtex_bool }
33   { }

34 \bool_if:NT \c__nm_draft_bool
35   { \msg_warning:nn { nicematrix } { Draft-mode } }
```

¹⁷cf. tex.stackexchange.com/questions/57424/using-of-usetikzlibrary-in-an-expl3-package-fails

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.

```

36 \ProvideDocumentCommand \iddots { }
37 {
38   \mathinner
39   {
40     \mkern 1 mu
41     \raise \p@ \hbox:n { . }
42     \mkern 2 mu
43     \raise 4 \p@ \hbox:n { . }
44     \mkern 2 mu
45     \raise 7 \p@ \vbox { \kern 7 pt \hbox:n { . } } \mkern 1 mu
46   }
47 }

```

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.

```

48 \int_new:N \g__nm_env_int

```

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

```

49 \dim_new:N \l__nm_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`.

```

50 \seq_new:N \g__nm_names_seq

```

We want to know if we are in an environment of `nicematrix` because we will raise an error if the user tries to use nested environments.

```

51 \bool_new:N \l__nm_in_env_bool

```

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

```

52 \bool_new:N \l__nm_NiceArray_bool
53 \bool_new:N \g__nm_NiceArray_bool

```

```

54 \cs_new_protected:Npn \__nm_test_if_math_mode:
55 {
56   \if_mode_math: \else:
57     \__nm_fatal:n { Outside~math~mode }
58   \fi:
59 }

```

Consider the following code:

```

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

```

First, the dotted line created by the `\Vdots` will be drawn. The implicit cell in position 2-3 will be considered as “dotted”. Then, we will have to draw the dotted line specified by the `\Cdots`; the final extremity of that line will be exactly in position 2-3 and, for that new second line, it should be considered as a *closed* extremity (since it is dotted). However, we don’t have the (normal) Tikz node of that node (since it’s an implicit cell): we can’t draw such a line. That’s why that dotted line will be said *impossible* and an error will be raised.¹⁸

```
60 \bool_new:N \l__nm_impossible_line_bool
```

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

```
61 \bool_new:N \c__nm_colortbl_loaded_bool
62 \AtBeginDocument
63 {
64   \@ifpackageloaded { colortbl }
65   {
66     \bool_set_true:N \c__nm_colortbl_loaded_bool
67     \cs_set_protected:Npn \__nm_vline_i: { { \CT@arc@ \vline } }
68   }
69   { }
70 }
```

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

```
71 \dim_new:N \l__nm_inter_dots_dim
72 \dim_set:Nn \l__nm_inter_dots_dim { 0.45 em }
```

The length `\l_@@_radius_dim` is the radius of the dots for the dotted lines. The default value is 0.34 pt but it will be changed if the option `small` is used.

```
73 \dim_new:N \l__nm_radius_dim
74 \dim_set:Nn \l__nm_radius_dim { 0.53 pt }
```

The name of the current environment (will be used only in the error messages).

```
75 \str_new:N \g__nm_type_env_str
```

15.2.1 Variables for the exterior rows and columns

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

- **First row**

The integer `\l_@@_first_row_int` is the number of the first row of the array. The default value is 1, but, if the option `first-row` is used, the value will be 0. As usual, the global version is for the passage in the `\group_insert_after:N`.

```
76 \int_new:N \l__nm_first_row_int
77 \int_set:Nn \l__nm_first_row_int 1
78 \int_new:N \g__nm_first_row_int
```

- **First column**

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

```
79 \int_new:N \l__nm_first_col_int
80 \int_set:Nn \l__nm_first_col_int 1
81 \int_new:N \g__nm_first_col_int
```

¹⁸Of course, the user should solve the problem by adding the lacking ampersands.

- **Last row**

The counter `\l_@@_last_row_int` is the number of the eventual “last row”, as specified by the key `last-row`. A value of `-2` means that there is no “last row”. A value of `-1` means that there is a “last row” but we don’t know the number of that row (the key `last-row` has been used without value and the actual value has not still been read in the `aux` file).

```
82 \int_new:N \l__nm_last_row_int
83 \int_set:Nn \l__nm_last_row_int { -2 }
84 \int_new:N \g__nm_last_row_int
```

If, in an environment like `{pNiceArray}`, the option `last-row` is used without value, we will globally raise the following flag. It will be used to know if we have, after the construction of the array, to write in the `aux` file the number of the “last row”.¹⁹

```
85 \bool_new:N \l__nm_last_row_without_value_bool
86 \bool_new:N \g__nm_last_row_without_value_bool
```

- **Last column**

For the eventual “last column”, we have a boolean and not an integer.

```
87 \bool_new:N \l__nm_last_col_bool
```

However, we have also another boolean. Consider the following code:

```
\begin{pNiceArray}{CC}[last-col]
  1 & 2 \\
  3 & 4
\end{pNiceArray}
```

In such a code, the “last column” specified by the key `last-col` is not used. We want to be able to detect such a situation and we create a boolean for that job.

```
88 \bool_new:N \g__nm_last_col_found_bool
```

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

15.2.2 The column `S` of `siunitx`

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

```
89 \bool_new:N \c__nm_siunitx_loaded_bool
90 \AtBeginDocument
91 {
92   \ifpackageloaded { siunitx }
93     { \bool_set_true:N \c__nm_siunitx_loaded_bool }
94     { }
95 }
```

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:

¹⁹We can’t use `\l_@@_last_row_int` for this usage because, if `nicematrix` has read its value from the `aux` file, the value of the counter won’t be `-1` any longer.

```

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

```

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

```

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

```

However, we don't want to use explicitly any private command of `siunitx`. That's why we will extract the name of the two `_siunitx...` commands by their position in the code of `\NC@rewrite@S`. Since the command `\NC@rewrite@S` appends some tokens to the `toks` list `\@temptokena`, we use the LaTeX command `\NC@rewrite@S` in a group (`\group_begin:-\group_end:`) and we extract the two command names which are in the `toks` `\@temptokena`. However, this extraction can be done only when `siunitx` is loaded (and it may be loaded after `nicematrix`) and, in fact, after the beginning of the document — because some instructions of `siunitx` are executed in a `\AtBeginDocument`). That's why this extraction will be done only at the first utilisation of an environment of `nicematrix` with the command `\@@_adapt_S_column:`.

```

96 \cs_set_protected:Npn \_nm_adapt_S_column:
97 {

```

In the preamble of the LaTeX document, the boolean `\c_@@_siunitx_loaded_bool` won't be known. That's why we test the existence of `\c_@@_siunitx_loaded_bool` and not its value.²⁰

```

98   \bool_if:NT \c_@@_siunitx_loaded_bool
99   {
100     \group_begin:
101     \@temptokena = { }

```

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

```

102     \cs_set_eq:MN \NC@find \prg_do_nothing:
103     \NC@rewrite@S { }

```

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

```

104     \tl_gset:NV \g_tmpa_tl \@temptokena
105     \group_end:
106     \tl_new:N \c_@@_nm_table_collect_begin_tl
107     \tl_set:Nx \l_tmpa_tl { \tl_item:Nn \g_tmpa_tl 2 }
108     \tl_gset:Nx \c_@@_nm_table_collect_begin_tl { \tl_item:Nn \l_tmpa_tl 1 }
109     \tl_new:N \c_@@_nm_table_print_tl
110     \tl_gset:Nx \c_@@_nm_table_print_tl { \tl_item:Nn \g_tmpa_tl { -1 } }

```

²⁰Indeed, `nicematrix` may be used in the preamble of the LaTeX document. For example, in this document, we compose a matrix in the box `\ExampleOne` before loading `arydshln` (because `arydshln` is not totally compatible with `nicematrix`).

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

If the adaptation has been done, the command `\@@_adapt_S_column:` becomes no-op (globally).

```

111     \cs_gset_eq:NN \_nm_adapt_S_column: \prg_do_nothing:
112   }
113 }

```

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).

```

114 \cs_new_protected:Npn \_nm_renew_NC@rewrite@S:
115   {
116     \renewcommand*{\NC@rewrite@S}[1] []
117     {
118       \@temptokena \exp_after:wN
119       {
120         \tex_the:D \@temptokena
121         > { \_nm_Cell: \c_@@_table_collect_begin_tl S {##1} }
122         c
123         < { \c_@@_table_print_tl \_nm_end_Cell: }
124       }
125       \NC@find
126     }
127 }

```

15.3 The options

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}`, `{pNiceArray}` and their variants, the value will programmatically be fixed to `c`. For the environment `{NiceArray}`, however, the three values `t`, `c` and `b` are possible.

```

128 \str_new:N \l_@@_nm_pos_env_str
129 \str_set:Nn \l_@@_nm_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}` (as it is done in `{array}` of `array`).

```

130 \bool_new:N \l_@@_nm_exterior_arraycolsep_bool

```

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

```

131 \bool_new:N \l_@@_nm_parallelize_diags_bool
132 \bool_set_true:N \l_@@_nm_parallelize_diags_bool

```

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

```

133 \bool_new:N \l_@@_nm_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.).

```

134 \bool_new:N \l_@@_nm_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 potential exterior columns).

```

135 \bool_new:N \l_@@_nm_auto_columns_width_bool

```

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.

```
136 \str_new:N \l__nm_name_str
137 \str_new:N \g__nm_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.

```
138 \bool_new:N \l__nm_extra_nodes_bool
139 \bool_new:N \g__nm_extra_nodes_bool
```

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

```
140 \dim_new:N \l__nm_left_margin_dim
141 \dim_new:N \l__nm_right_margin_dim
142 \dim_new:N \g__nm_left_margin_dim
143 \dim_new:N \g__nm_right_margin_dim
144 \dim_new:N \g__nm_width_last_col_dim
145 \dim_new:N \g__nm_width_first_col_dim
```

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

```
146 \dim_new:N \l__nm_extra_left_margin_dim
147 \dim_new:N \l__nm_extra_right_margin_dim
148 \dim_new:N \g__nm_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 sets of keys.

```
149 \keys_define:nn { NiceMatrix / Global }
150   {
151     small .bool_set:N = \l__nm_small_bool ,
152     hlines .bool_set:N = \l__nm_hlines_bool ,
153     parallelize-diags .bool_set:N = \l__nm_parallelize_diags_bool ,
```

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

```
154   renew-dots .bool_set:N = \l__nm_renew_dots_bool ,
155   nullify-dots .bool_set:N = \l__nm_nullify_dots_bool ,
```

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.²¹

```
156   create-extra-nodes .bool_set:N = \l__nm_extra_nodes_bool ,
157   left-margin .dim_set:N = \l__nm_left_margin_dim ,
158   left-margin .default:n = \arraycolsep ,
159   right-margin .dim_set:N = \l__nm_right_margin_dim ,
160   right-margin .default:n = \arraycolsep ,
161   margin .meta:n = { left-margin = #1 , right-margin = #1 } ,
162   margin .default:n = \arraycolsep ,
163   extra-left-margin .dim_set:N = \l__nm_extra_left_margin_dim ,
164   extra-right-margin .dim_set:N = \l__nm_extra_right_margin_dim ,
165   extra-margin .meta:n =
166     { extra-left-margin = #1 , extra-right-margin = #1 } ,
167 }
```

²¹In fact, we should not because, if there is a `w` node, the `w` node is used instead of the “medium” node.

The following set of keys concerns the options to *customize* the exterior rows and columns, that is to say the options `code-for-first-row`, etc.

```

168 \keys_define:nn { NiceMatrix / Code }
169   {
170     code-for-first-col .tl_set:N = \l__nm_code_for_first_col_tl ,
171     code-for-first-col .value_required:n = true ,
172     code-for-last-col .tl_set:N = \l__nm_code_for_last_col_tl ,
173     code-for-last-col .value_required:n = true ,
174     code-for-first-row .tl_set:N = \l__nm_code_for_first_row_tl ,
175     code-for-first-row .value_required:n = true ,
176     code-for-last-row .tl_set:N = \l__nm_code_for_last_row_tl ,
177     code-for-last-row .value_required:n = true ,
178   }

```

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

```

179 \keys_define:nn { NiceMatrix / Env }
180   {
181     columns-width .code:n =
182       \str_if_eq:nnTF { #1 } { auto }
183       { \bool_set_true:N \l__nm_auto_columns_width_bool }
184       { \dim_set:Nn \l__nm_columns_width_dim { #1 } } ,
185     columns-width .value_required:n = true ,
186     name .code:n =
187       \str_set:Nn \l_tmpa_str { #1 }
188       \seq_if_in:NVTF \g__nm_names_seq \l_tmpa_str
189       { \__nm_error:nn { Duplicate-name } { #1 } }
190       { \seq_gput_left:NV \g__nm_names_seq \l_tmpa_str }
191       \str_set_eq:NN \l__nm_name_str \l_tmpa_str ,
192     name .value_required:n = true ,
193     code-after .tl_gset:N = \g__nm_code_after_tl ,
194     code-after .initial:n = \c_empty_tl ,
195     code-after .value_required:n = true ,
196   }

```

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

```

197 \keys_define:nn { NiceMatrix }
198   {
199     NiceMatrixOptions .inherit:n =
200       {
201         NiceMatrix / Global ,
202         NiceMatrix / Code
203       } ,
204     NiceMatrix .inherit:n =
205       {
206         NiceMatrix / Global ,
207         NiceMatrix / Env
208       } ,
209     NiceArray .inherit:n =
210       {
211         NiceMatrix / Global ,
212         NiceMatrix / Env ,
213         NiceMatrix / Code
214       } ,
215     pNiceArray .inherit:n =
216       {
217         NiceMatrix / Global ,
218         NiceMatrix / Env ,
219         NiceMatrix / Code
220       }
221   }

```

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

```
222 \keys_define:nn { NiceMatrix / NiceMatrixOptions }
223   {
```

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

```
224   renew-matrix .code:n = \__nm_renew_matrix: ,
225   renew-matrix .value_forbidden:n = true ,
226   RenewMatrix .code:n = \__nm_error:n { Option~RenewMatrix~suppressed }
227     \__nm_renew_matrix: ,
228   transparent .meta:n = { renew-dots , renew-matrix } ,
229   transparent .value_forbidden:n = true,
230   Transparent .code:n = \__nm_error:n { Option~Transparent~suppressed }
231     \__nm_renew_matrix:
232     \bool_set_true:N \l__nm_renew_dots_bool ,
```

The option `exterior-arraycolsep` will have effect only in `{NiceArray}` for those who want to have for `{NiceArray}` the same behaviour as `{array}`.

```
233   exterior-arraycolsep .bool_set:N = \l__nm_exterior_arraycolsep_bool ,
```

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

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

```
234   columns-width .code:n =
235     \str_if_eq:nnTF { #1 } { auto }
236     { \__nm_error:n { Option~auto~for~columns~width } }
237     { \dim_set:Nn \l__nm_columns_width_dim { #1 } } ,
```

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

```
238   allow-duplicate-names .code:n =
239     \__nm_msg_redirect_name:nn { Duplicate~name } { none } ,
240   allow-duplicate-names .value_forbidden:n = true ,
```

By default, the specifier used in the preamble of the array (for example in `{pNiceArray}`) to draw a vertical dotted line between two columns is the colon “:”. However, it’s possible to change this letter with `letter-for-dotted-lines` and, by the way, the letter “:” will remain free for other packages (for example `arydshln`).

```
241   letter-for-dotted-lines .code:n =
242     {
243       \int_compare:nTF { \tl_count:n { #1 } = \c_one_int }
244         { \str_set:Nx \l__nm_letter_for_dotted_lines_str { #1 } }
245         { \__nm_error:n { Bad~value~for~letter~for~dotted~lines } }
246     } ,
247   letter-for-dotted-lines .value_required:n = true ,
248   letter-for-dotted-lines .initial:n = \c_colon_str ,

249   unknown .code:n = \__nm_error:n { Unknown~key~for~NiceMatrixOptions } }
```

`\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.

```
250 \NewDocumentCommand \NiceMatrixOptions { m }
251   { \keys_set:nn { NiceMatrix / NiceMatrixOptions } { #1 } }
```

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

```
252 \keys_define:nn { NiceMatrix / NiceMatrix }
253   { unknown .code:n = \__nm_error:n { Unknown~option~for~NiceMatrix } }
```

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

```
254 \keys_define:nn { NiceMatrix / NiceArray }
255   {
```

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

```
256   c .code:n = \str_set:Nn \l__nm_pos_env_str c ,
257   t .code:n = \str_set:Nn \l__nm_pos_env_str t ,
258   b .code:n = \str_set:Nn \l__nm_pos_env_str b ,
259   first-col .code:n = \int_zero:N \l__nm_first_col_int ,
260   last-col .bool_set:N = \l__nm_last_col_bool ,
261   first-row .code:n = \int_zero:N \l__nm_first_row_int ,
262   last-row .int_set:N = \l__nm_last_row_int ,
263   last-row .default:n = -1 ,
264   unknown .code:n = \__nm_error:n { Unknown~option-for-NiceArray }
265 }

```

```
266 \keys_define:nn { NiceMatrix / pNiceArray }
267   {
268     first-col .code:n = \int_zero:N \l__nm_first_col_int ,
269     last-col .bool_set:N = \l__nm_last_col_bool ,
270     first-row .code:n = \int_zero:N \l__nm_first_row_int ,
271     last-row .int_set:N = \l__nm_last_row_int ,
272     last-row .default:n = -1 ,
273     unknown .code:n = \__nm_error:n { Unknown~option-for-pNiceArray }
274 }

```

15.4 Code common to {NiceArrayWithDelims} 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}).

```
275 \cs_new_protected:Nn \__nm_Cell:
276   {
```

We increment \g_@@_col_int, which is the counter of the columns.

```
277   \int_gincr:N \g__nm_col_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.

```
278   \int_compare:nNnT \g__nm_col_int = \c_one_int
279     {
280       \int_compare:nNnT \l__nm_first_col_int = \c_one_int
281         \__nm_begin_of_row:
282     }

```

```
283   \int_gset:Nn \g__nm_col_total_int
284     { \int_max:nn \g__nm_col_total_int \g__nm_col_int }

```

The content of the cell is composed in the box \l_tmpa_box because we want to compute some dimensions of the box. The \hbox_set_end: corresponding to this \hbox_set:Nw will be in the \@@_end_Cell: (and the \c_math_toggle_token also).

```
285   \hbox_set:Nw \l_tmpa_box
286   \c_math_toggle_token
287   \bool_if:NT \l__nm_small_bool \scriptstyle
288   \int_compare:nNnTF \g__nm_row_int = \c_zero_int
289     \l__nm_code_for_first_row_tl
290     {
291       \int_compare:nNnT \g__nm_row_int = \l__nm_last_row_int
292         \l__nm_code_for_last_row_tl
293     }
294 }

```

The following macro `\@@_begin_of_row` is usually used in the cell number 1 of the array. However, when the key `first-col` is used, `\@@_begin_of_row` is executed in the cell number 0 of the array.

```

295 \cs_new_protected:Nn \__nm_begin_of_row:
296 {
297   \int_gincr:N \g__nm_row_int
298   \dim_gset_eq:NN \g__nm_dp_ante_last_row_dim \g__nm_dp_last_row_dim
299   \dim_gzero:N \g__nm_dp_last_row_dim
300   \dim_gzero:N \g__nm_ht_last_row_dim
301 }

```

The following code is used in each cell of the array. It actualises quantities that, at the end of the array, will give informations about the vertical dimension of the two first rows and the two last rows.

```

302 \cs_new_protected:Npn \__nm_actualization_for_first_and_last_row:
303 {
304   \int_compare:nNnT \g__nm_row_int = \c_zero_int
305   {
306     \dim_gset:Nn \g__nm_dp_row_zero_dim
307     { \dim_max:nn \g__nm_dp_row_zero_dim { \box_dp:N \l_tmpa_box } }
308     \dim_gset:Nn \g__nm_ht_row_zero_dim
309     { \dim_max:nn \g__nm_ht_row_zero_dim { \box_ht:N \l_tmpa_box } }
310   }
311   \int_compare:nNnT \g__nm_row_int = \c_one_int
312   {
313     \dim_gset:Nn \g__nm_ht_row_one_dim
314     { \dim_max:nn \g__nm_ht_row_one_dim { \box_ht:N \l_tmpa_box } }
315   }
316   \dim_gset:Nn \g__nm_ht_last_row_dim
317   { \dim_max:nn \g__nm_ht_last_row_dim { \box_ht:N \l_tmpa_box } }
318   \dim_gset:Nn \g__nm_dp_last_row_dim
319   { \dim_max:nn \g__nm_dp_last_row_dim { \box_dp:N \l_tmpa_box } }
320 }
321 \cs_new_protected:Nn \__nm_end_Cell:
322 {
323   \c_math_toggle_token
324   \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 “first column” and the “last column”).

```

325 \dim_gset:Nn \g__nm_max_cell_width_dim
326 { \dim_max:nn \g__nm_max_cell_width_dim { \box_wd:N \l_tmpa_box } }

```

The following computations are for the “first row” and the “last row”.

```

327 \__nm_actualization_for_first_and_last_row:

```

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

```

328 \tikz
329 [
330   remember~picture ,
331   inner~sep = \c_zero_dim ,
332   minimum~width = \c_zero_dim ,
333   baseline
334 ]
335 \node
336 [
337   anchor = base ,
338   name = nm - \int_use:N \g__nm_env_int -
339           \int_use:N \g__nm_row_int -
340           \int_use:N \g__nm_col_int ,
341   alias =
342     \str_if_empty:NF \l__nm_name_str
343     {
344       \l__nm_name_str -

```



```

345         \int_use:N \g__nm_row_int -
346         \int_use:N \g__nm_col_int
347     }
348 ]
349 \bgroup
350 \box_use:N \l_tmpa_box
351 \egroup ;
352 }
353 \cs_generate_variant:Nn \dim_set:Nn { N x }

```

In the environment `{NiceArrayWithDelims}`, 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 `\@@_renewcolumnntype:nn`. The first argument is the type of the column (`w` or `W`) and the second argument is a code inserted at a special place and which is the only difference between the two definitions.

```

354 \cs_new_protected:Nn \_nm_renewcolumnntype:nn
355 {
356     \newcolumnntype #1 [ 2 ]
357     {
358         > {
359             \hbox_set:Nw \l_tmpa_box
360             \_nm_Cell:
361         }
362         c
363         < {
364             \_nm_end_Cell:
365             \hbox_set_end:
366             #2
367             \hbox_set:Nn \l_tmpb_box
368             { \makebox [ ##2 ] [ ##1 ] { \box_use:N \l_tmpa_box } }
369             \dim_set:Nn \l_tmpa_dim { \box_dp:N \l_tmpb_box }
370             \box_move_down:nn \l_tmpa_dim
371             {
372                 \vbox:n
373                 {
374                     \hbox_to_wd:nn { \box_wd:N \l_tmpb_box }
375                     {
376                         \hfil
377                         \tikz [ remember-picture , overlay ]
378                         \coordinate ( __nm-north-east ) ;
379                     }
380                     \hbox:n
381                     {
382                         \tikz [ remember-picture , overlay ]
383                         \coordinate ( __nm-south-west ) ;
384                         \box_move_up:nn \l_tmpa_dim { \box_use:N \l_tmpb_box }
385                     }
386                 }
387             }

```

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 an erroneous result with some engines (XeTeX, LuaTeX) although the result is good with `pdfflatex` (why?).

```

388         \tikz [ remember-picture , overlay ]
389         \node
390         [
391             node~contents = { } ,
392             name = nm - \int_use:N \g__nm_env_int -
393                 \int_use:N \g__nm_row_int -
394                 \int_use:N \g__nm_col_int - w,
395             alias =

```

```

396         \str_if_empty:NF \l__nm_name_str
397         {
398             \l__nm_name_str -
399             \int_use:N \g__nm_row_int -
400             \int_use:N \g__nm_col_int - w
401         } ,
402         inner~sep = \c_zero_dim ,
403         fit = ( __nm~south~west ) ( __nm~north~east )
404     ]
405 ;
406 }
407 }
408 }

```

The argument of the following command `\@@_instruction_of_type:n` defined below is the type of the instruction (Cdots, Vdots, Ddots, etc.). This command writes in the correspondent `\g_@@_type_lines_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 & \Cdots & & 
\end{pNiceMatrix}

```

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

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

```

\@@_draw_Cdots:nn {2}{2}
\@@_draw_Cdots:nn {3}{2}

```

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

```

409 \bool_if:NTF \c__nm_draft_bool
410 { \cs_set_protected:Npn \__nm_instruction_of_type:n #1 { } }
411 {
412     \cs_new_protected:Npn \__nm_instruction_of_type:n #1
413     {

```

It's important to use a `\tl_gput_right:cx` and not a `\tl_gput_left:cx` because we want the `\Ddots` lines to be drawn in the order of appearance in the array (for parallelisation).

```

414     \tl_gput_right:cx
415     { g__nm_ #1 _ lines _ tl }
416     {
417         \use:c { __nm _ draw _ #1 : nn }
418         { \int_use:N \g__nm_row_int }
419         { \int_use:N \g__nm_col_int }
420     }
421 }
422 }

```

We want to use `\array` of `array`. However, if the class used is `revtex4-1` or `revtex4-2`, we have to do some tuning and use the command `\@array@array` instead of `\array` because these classes do a redefinition of `\array` incompatible with our use of `\array`.

```

423 \cs_new_protected:Npn \__nm_array:
424 {
425     \bool_if:NTF \c__nm_revtex_bool
426     {
427         \cs_set_eq:NN \@acol1 \@arrayacol
428         \cs_set_eq:NN \@acolr \@arrayacol
429         \cs_set_eq:NN \@acol \@arrayacol
430         \cs_set:Npn \@halignto { }
431         \@array@array
432     }
433     \array

```

`\l_@@_pos_env_str` may have the value `t`, `c` or `b`.

```
434   [ \l__nm_pos_env_str ]
435 }
```

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

```
436 \cs_new:Npn \__nm_everycr:
437   { \noalign { \__nm_everycr_i: } }
438 \cs_new_protected:Npn \__nm_everycr_i:
439   {
440     \int_gzero:N \g__nm_col_int
441     \bool_if:NT \l__nm_hlines_bool
442     {
```

The counter `\g_@@_row_int` has the value `-1` only if there is a “first row” and that we are before that “first row”, i.e. just before the beginning of the array.

```
443     \int_compare:nNnT \g__nm_row_int > { -1 }
444     {
445       \int_compare:nNnF \g__nm_row_int = \g__nm_last_row_int
446       {
447         \hrule \@height \arrayrulewidth
448         \skip_vertical:n { - \arrayrulewidth }
449       }
450     }
451   }
452 }
```

The following code `\@@_pre_array:` is used in `{NiceArray}` and in `{NiceMatrix}`. It contains code that will be executed *before* the construction of the array.

```
453 \cs_new_protected:Npn \__nm_pre_array:
454   {
```

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

```
455   \bool_if:NT \l__nm_small_bool
456   {
457     \cs_set:Npn \arraystretch { 0.47 }
458     \dim_set:Nn \arraycolsep { 1.45 pt }
459   }
```

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). Remark that, even if the environment has a name (see just below) we have to write in the `aux` file the information with the number of environment because of `auto-columns-width` of `{NiceMatrixBlock}`.

```
460   \bool_if:NT \l__nm_auto_columns_width_bool
461   {
462     \group_insert_after:N \__nm_write_max_cell_width:
463     \cs_if_free:cTF { __nm_max_cell_width_ \int_use:N \g__nm_env_int }
464     { \dim_zero:N \l__nm_columns_width_dim }
465     {
466       \dim_set:Nx \l__nm_columns_width_dim
467       { \use:c { __nm_max_cell_width _ \int_use:N \g__nm_env_int } }
468     }
469   }
```

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)).

```
469     \str_if_empty:NF \l__nm_name_str
```

```

470     {
471       \cs_if_free:cF { __nm_max_cell_width_ \l__nm_name_str }
472       {
473         \dim_set:Nx \l__nm_columns_width_dim
474         { \use:c { __nm_max_cell_width_ \l__nm_name_str } }
475       }
476     }
477   }

```

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.

```

478   \bool_gset_eq:NN \g__nm_extra_nodes_bool \l__nm_extra_nodes_bool
479   \dim_gset_eq:NN \g__nm_left_margin_dim \l__nm_left_margin_dim
480   \dim_gset_eq:NN \g__nm_right_margin_dim \l__nm_right_margin_dim
481   \dim_gset_eq:NN \g__nm_extra_right_margin_dim \l__nm_extra_right_margin_dim
482   \int_gset_eq:NN \g__nm_last_row_int \l__nm_last_row_int
483   \tl_gset_eq:NN \g__nm_name_str \l__nm_name_str
484   \int_gset_eq:NN \g__nm_first_row_int \l__nm_first_row_int
485   \int_gset_eq:NN \g__nm_first_col_int \l__nm_first_col_int
486   \bool_gset_eq:NN \g__nm_NiceArray_bool \l__nm_NiceArray_bool
487   \bool_gset_eq:NN \g__nm_last_row_without_value_bool
488   \l__nm_last_row_without_value_bool
489   \bool_gset_eq:NN \g__nm_small_bool \l__nm_small_bool

```

The environment `{array}` uses internally the command `\ialign`. We change the definition of `\ialign` for several reasons. In particular, `\ialign` sets `\everycr` to `{ }`. However, we have to change the value of `\everycr`.

```

490   \cs_set:Npn \ialign
491   {
492     \bool_if:NTF \c__nm_colortbl_loaded_bool
493     {
494       \CT@everycr
495       {
496         \noalign { \cs_gset_eq:NN \CT@row@color \prg_do_nothing: }
497         \__nm_everycr:
498       }
499     }
500     { \everycr { \__nm_everycr: } }
501     \tabskip = \c_zero_skip

```

The box `\@arstrutbox` is a box constructed in the beginning of the environment `{array}`. The construction of that box takes into account the current values of `\arraystretch`²² and `\extrarowheight` (of `array`). That box is inserted (via `\@arstrut`) in the beginning of each row of the array. That's why we use the dimensions of that box to initialize the variables which will be the dimensions of the potential first and last row of the environment. This initialization must be done after the creation of `\@arstrutbox` and that's why we do it in the `\ialign`.

```

502   \dim_gzero_new:N \g__nm_dp_row_zero_dim
503   \dim_gset:Nn \g__nm_dp_row_zero_dim { \box_dp:N \@arstrutbox }
504   \dim_gzero_new:N \g__nm_ht_row_zero_dim
505   \dim_gset:Nn \g__nm_ht_row_zero_dim { \box_ht:N \@arstrutbox }
506   \dim_gzero_new:N \g__nm_ht_row_one_dim
507   \dim_gset:Nn \g__nm_ht_row_one_dim { \box_ht:N \@arstrutbox }
508   \dim_gzero_new:N \g__nm_dp_ante_last_row_dim
509   \dim_gset:Nn \g__nm_dp_ante_last_row_dim { \box_dp:N \@arstrutbox }
510   \dim_gzero_new:N \g__nm_ht_last_row_dim
511   \dim_gset:Nn \g__nm_ht_last_row_dim { \box_ht:N \@arstrutbox }
512   \dim_gzero_new:N \g__nm_dp_last_row_dim
513   \dim_gset:Nn \g__nm_dp_last_row_dim { \box_dp:N \@arstrutbox }

```

²²The option `small` of `nicematrix` changes (among other) the value of `\arraystretch`. This is done, of course, before the call of `{array}`.

After its first utilisation, the definition of `\ialign` will revert automatically to its default definition. With this programming, we will have, in the cells of the array, a clean version of `\ialign`.²³

```

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__nm_columns_width_dim = \c_zero_dim
523     {
524         \newcolumntype L { > \__nm_Cell: l < \__nm_end_Cell: }
525         \newcolumntype C { > \__nm_Cell: c < \__nm_end_Cell: }
526         \newcolumntype R { > \__nm_Cell: r < \__nm_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__nm_columns_width_dim } }
530         \newcolumntype C { w c { \dim_use:N \l__nm_columns_width_dim } }
531         \newcolumntype R { w r { \dim_use:N \l__nm_columns_width_dim } }
532     }

533     \cs_set_eq:NN \Ldots \__nm_Ldots
534     \cs_set_eq:NN \Cdots \__nm_Cdots
535     \cs_set_eq:NN \Vdots \__nm_Vdots
536     \cs_set_eq:NN \Ddots \__nm_Ddots
537     \cs_set_eq:NN \Iddots \__nm_Iddots
538     \cs_set_eq:NN \hdottedline \__nm_hdottedline:
539     \cs_set_eq:NN \Hspace \__nm_Hspace:
540     \cs_set_eq:NN \Hdotsfor \__nm_Hdotsfor:
541     \cs_set_eq:NN \multicolumn \__nm_multicolumn:nnn
542     \cs_set_eq:NN \Block \__nm_Block:
543     \bool_if:NT \l__nm_renew_dots_bool
544     {
545         \cs_set_eq:NN \ldots \__nm_Ldots
546         \cs_set_eq:NN \cdots \__nm_Cdots
547         \cs_set_eq:NN \vdots \__nm_Vdots
548         \cs_set_eq:NN \ddots \__nm_Ddots
549         \cs_set_eq:NN \iddots \__nm_Iddots
550         \cs_set_eq:NN \dots \__nm_Ldots
551         \cs_set_eq:NN \hdotsfor \__nm_Hdotsfor:
552     }

```

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).

```

553     \seq_gclear_new:N \g__nm_multicolumn_cells_seq
554     \seq_gclear_new:N \g__nm_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).

```

555     \int_gzero_new:N \g__nm_row_int
556     \int_gset:Nn \g__nm_row_int { \l__nm_first_row_int - 1 }

```

²³The user will probably not employ directly `\ialign` in the array... but more likely environments that utilize `\ialign` internally (e.g.: `{substack}`).

At the end of the environment `{array}`, `\g_@@_row_int` will be the total number of rows and `\g_@@_row_total_int` will be the number of rows excepted the last row (if `\l_@@_last_row_bool` has been raised with the option `last-row`).

```
557 \int_gzero_new:N \g__nm_row_total_int
```

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

```
558 \int_gzero_new:N \g__nm_col_int
559 \int_gzero_new:N \g__nm_col_total_int
560 \cs_set_eq:NN \@ifnextchar \new@ifnextchar
```

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

```
561 \cs_set_eq:NN \NC@find@w \relax
562 \cs_set_eq:NN \NC@find@W \relax
563 \__nm_renewcolumnntype:nn w { }
564 \__nm_renewcolumnntype:nn W { \cs_set_eq:NN \hss \hfil }
```

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

```
565 \tl_set_rescan:Nno
566 \l__nm_letter_for_dotted_lines_str { } \l__nm_letter_for_dotted_lines_str
567 \exp_args:NV \newcolumnntype \l__nm_letter_for_dotted_lines_str
568 {
569 !
570 {
571 \skip_horizontal:n { 0.53 pt }
572 \bool_gset_true:N \g__nm_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`.

```
573 \int_compare:nNnT \g__nm_col_int > \g__nm_last_vdotted_col_int
574 {
575 \int_gset_eq:NN \g__nm_last_vdotted_col_int \g__nm_col_int
576 \tl_gput_right:Nx \g__nm_code_after_tl
```

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

```
577 { \__nm_vdottedline:n { \int_use:N \g__nm_col_int } }
578 }
579 }
580 }
581 \int_gzero_new:N \g__nm_last_vdotted_col_int
582 \bool_if:NT \c__nm_siunitx_loaded_bool \__nm_renew_NC@rewrite@S:
```

```

583 \int_gset:Nn \g__nm_last_vdotted_col_int { -1 }
584 \bool_gset_false:N \g__nm_last_col_found_bool

```

During the construction of the array, the instructions `\Cdots`, `\Ldots`, etc. will be written in token lists `\g_@@_Cdots_lines_tl`, etc. which will be executed after the construction of the array.

```

585 \tl_gclear_new:N \g__nm_Cdots_lines_tl
586 \tl_gclear_new:N \g__nm_Ldots_lines_tl
587 \tl_gclear_new:N \g__nm_Vdots_lines_tl
588 \tl_gclear_new:N \g__nm_Ddots_lines_tl
589 \tl_gclear_new:N \g__nm_Iddots_lines_tl
590 \tl_gclear_new:N \g__nm_Hdotsfor_lines_tl
591 }

```

15.5 The environment `{NiceArrayWithDelims}`

```

592 \NewDocumentEnvironment { NiceArrayWithDelims } { m m O { } m ! O { } }
593 {
594   \str_if_empty:NT \g__nm_type_env_str
595   { \str_gset:Nn \g__nm_type_env_str { NiceArrayWithDelims } }
596   \__nm_adapt_S_column:
597   \__nm_test_if_math_mode:
598   \bool_if:NT \l__nm_in_env_bool { \__nm_fatal:n { Yet~in~env } }
599   \bool_set_true:N \l__nm_in_env_bool

```

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

```

600   \cs_if_exist:NT \tikz@library@external@loaded
601   { \tikzset { external / export = false } }

```

In `{NiceArrayWithDelims}`, it would have been possible to avoid the `\group_insert_after:N` and to put `\@@_after_array` in the second part of the environment `{NiceArrayWithDelims}`. However, it's not possible to do that in `{NiceMatrix}` (because of the option `renew-matrix`) and that's why we use this technique in `{NiceArrayWithDelims}` and in `{NiceMatrix}`.

```

602   \group_insert_after:N \__nm_after_array:

```

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

```

603   \int_gincr:N \g__nm_env_int
604   \bool_if:NF \l__nm_block_auto_columns_width_bool
605   { \dim_gzero_new:N \g__nm_max_cell_width_dim }

```

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

```

606   \cs_set_protected:Npn \@arrayrule { \@addtopreamble \__nm_vline: }

```

The set of keys is not exactly the same for `{NiceArray}` and for the variants of `{NiceArray}` (`{pNiceArray}`, `{bNiceArray}`, etc.) because, for `{NiceArray}`, we have the options `t`, `c` and `b`.

```

607   \bool_if:NTF \l__nm_NiceArray_bool
608   { \keys_set:nn { NiceMatrix / NiceArray } }
609   { \keys_set:nn { NiceMatrix / pNiceArray } }
610   { #3 , #5 }

```

A value of `-1` for the counter `\l_@@_last_row_int` means that the user has used the option `last-row` without value, that is to say without specifying the number of that last row. In this case, we try to read that value from the aux file (if it has been written on a previous run).

```

611   \int_compare:nNnT \l__nm_last_row_int = { -1 }
612   {
613     \bool_set_true:N \l__nm_last_row_without_value_bool

```

A value based on the name is more reliable than a value based on the number of the environment.

```

614   \str_if_empty:NTF \g__nm_name_str
615   {
616     \cs_if_exist:cT { __nm_last_row_ \int_use:N \g__nm_env_int }
617     {
618       \int_set:Nn \l__nm_last_row_int
619       { \use:c { __nm_last_row_ \int_use:N \g__nm_env_int } }

```

```

620     }
621   }
622   {
623     \cs_if_exist:cT { __nm_last_row_ \g__nm_name_str }
624     {
625       \int_set:Nn \l__nm_last_row_int
626         { \use:c { __nm_last_row_ \g__nm_name_str } }
627     }
628   }
629 }

```

The code in `\@@_pre_array:` is common to `{NiceArrayWithDelims}` and `{NiceMatrix}`.

```

630   \__nm_pre_array:

```

We compute the width of the two delimiters.

```

631   \dim_zero_new:N \g__nm_left_delim_dim
632   \dim_zero_new:N \g__nm_right_delim_dim
633   \bool_if:NTF \l__nm_NiceArray_bool
634   {
635     \dim_gset:Nn \g__nm_left_delim_dim { 2 \arraycolsep }
636     \dim_gset:Nn \g__nm_right_delim_dim { 2 \arraycolsep }
637   }
638   {
639     \group_begin:
640     \dim_set_eq:NN \nulldelimiterspace \c_zero_dim
641     \hbox_set:Nn \l_tmpa_box
642     {
643       \c_math_toggle_token
644       \left #1 \vcenter to 3 cm { } \right.
645       \c_math_toggle_token
646     }
647     \dim_gset:Nn \g__nm_left_delim_dim { \box_wd:N \l_tmpa_box }
648     \hbox_set:Nn \l_tmpa_box
649     {
650       \dim_set_eq:NN \nulldelimiterspace \c_zero_dim
651       \c_math_toggle_token
652       \left. \vcenter to 3 cm { } \right #2
653       \c_math_toggle_token
654     }
655     \dim_gset:Nn \g__nm_right_delim_dim { \box_wd:N \l_tmpa_box }
656     \group_end:
657   }
658 }

```

The array will be composed in a box (named `\l_@@_the_array_box`) because we have to do manipulations concerning the potential exterior rows (such construction in a box is not possible for `{NiceMatrix}`).

```

659   \box_clear_new:N \l__nm_the_array_box

```

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

```

660   \tl_set:Nn \l_tmpa_tl { #4 }
661   \int_compare:nNnTF \l__nm_first_col_int = \c_zero_int
662   { \tl_put_left:NV \l_tmpa_tl \c__nm_preamble_first_col_tl }
663   {
664     \bool_if:NT \l__nm_NiceArray_bool
665     {
666       \bool_if:NF \l__nm_exterior_arraycolsep_bool
667       { \tl_put_left:Nn \l_tmpa_tl { @ { } } }
668     }
669   }
670   \bool_if:NTF \l__nm_last_col_bool
671   { \tl_put_right:NV \l_tmpa_tl \c__nm_preamble_last_col_tl }
672   {
673     \bool_if:NT \l__nm_NiceArray_bool

```



```

674     {
675         \bool_if:NF \l__nm_exterior_arraycolsep_bool
676         { \tl_put_right:Nn \l_tmpa_tl { @ { } } }
677     }
678 }

```

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

```

679 \hbox_set:Nw \l__nm_the_array_box
680 \skip_horizontal:n \l__nm_left_margin_dim
681 \skip_horizontal:n \l__nm_extra_left_margin_dim
682 \c_math_toggle_token

```

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

```

683 \exp_args:NV \__nm_array: \l_tmpa_tl
684 }

```

We begin the second part of the environment `{NiceArrayWithDelims}`.

```

685 {
686 \endarray
687 \c_math_toggle_token
688 \skip_horizontal:n \g__nm_right_margin_dim
689 \skip_horizontal:n \g__nm_extra_right_margin_dim
690 \hbox_set_end:

691 \int_compare:nNnT \g__nm_last_row_int > { -2 }
692 {
693     \bool_if:NF \g__nm_last_row_without_value_bool
694     {
695         \int_compare:nNnF \g__nm_last_row_int = \g__nm_row_int
696         {
697             \__nm_error:n { Wrong~last~row }
698             \int_gset_eq:NN \g__nm_last_row_int \g__nm_row_int
699         }
700     }
701 }

```

Now, we compute `\l_tmpa_dim` which is the vertical dimension of the “first row” above the array (when the key `first-row` is used).

```

702 \int_compare:nNnTF \l__nm_first_row_int = \c_zero_int
703 {
704     \dim_set:Nn \l_tmpa_dim
705     {
706         \g__nm_ht_row_one_dim + \g__nm_dp_row_zero_dim
707         + \lineskip
708         + \g__nm_ht_row_zero_dim - \g__nm_ht_row_one_dim
709     }
710 }
711 { \dim_zero:N \l_tmpa_dim }

```

We compute `\l_tmpb_dim` which is the vertical dimension of the “last row” below the array (when the key `last-row` is used). A value of `-2` for `\l_@@_last_row_int` means that there is no “last row”²⁴.

```

712 \int_compare:nNnTF \l__nm_last_row_int > { -2 }
713 {
714     \dim_set:Nn \l_tmpb_dim
715     {
716         \g__nm_ht_last_row_dim + \g__nm_dp_ante_last_row_dim
717         + \lineskip
718         + \g__nm_dp_last_row_dim - \g__nm_dp_ante_last_row_dim

```

²⁴A value of `-1` for `\l_@@_last_row_int` means that there is a “last row” but the number of that row is unknown (the user have not set the value with the option `last row`).

```

719     }
720   }
721   { \dim_zero:N \l_tmpb_dim }

```

Now, we begin the real construction in the output flow of TeX. First, we take into account a potential “first column” (we remind that this “first column” has been constructed in an overlapping position and that we have computed its width in `\g_@@_width_first_col_dim`: see p. 43).

```

722   \int_compare:nNnT \g__nm_first_col_int = \c_zero_int
723   {
724     \skip_horizontal:n \arraycolsep
725     \skip_horizontal:n \g__nm_width_first_col_dim
726   }

```

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

```

727   \bool_if:NTF \g__nm_NiceArray_bool
728   {
729     \int_compare:nNnT \g__nm_first_row_int = \c_zero_int
730     {
731       \str_if_eq:VnTF \l__nm_pos_env_str { t }
732       {
733         \box_move_up:nn
734         { \l_tmpa_dim - \g__nm_ht_row_zero_dim + \g__nm_ht_row_one_dim }
735       }
736     }
737     {
738       \bool_if:NT \g__nm_last_row_int
739       {
740         \str_if_eq:VnT \l__nm_pos_env_str { b }
741         {
742           \box_move_down:nn
743           {
744             \l_tmpb_dim
745             - \g__nm_dp_last_row_dim + \g__nm_dp_ante_last_row_dim
746           }
747         }
748       }
749     }
750     { \box_use_drop:N \l__nm_the_array_box }
751   }

```

Now, in the case of an environment `{pNiceArray}`, `{bNiceArray}`, etc.

```

752   {
753     \hbox_set:Nn \l_tmpa_box
754     {
755       \c_math_toggle_token
756       \left #1
757       \vcenter
758       {

```

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

```

759         \skip_vertical:n { - \l_tmpa_dim }
760         \hbox:n
761         {
762           \skip_horizontal:n { - \arraycolsep }
763           \box_use_drop:N \l__nm_the_array_box
764           \skip_horizontal:n { - \arraycolsep }
765         }

```

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

```

766         \skip_vertical:n { - \l_tmpb_dim }
767       }
768       \right #2
769       \c_math_toggle_token

```

```

770     }
771     \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }
772     \box_set_dp:Nn \l_tmpa_box { \box_dp:N \l_tmpa_box + \l_tmpb_dim }
773     \box_use_drop:N \l_tmpa_box
774 }

```

We take into account a potential “last column” (this “last column” has been constructed in an overlapping position and we have computed its width in `\g__nm_width_last_col_dim`: see p. 44).

```

775     \bool_if:NT \g__nm_last_col_found_bool
776     {
777         \skip_horizontal:n \g__nm_width_last_col_dim
778         \skip_horizontal:n \arraycolsep
779     }
780 }

```

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

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

```

781 \tl_const:Nn \c_nm_preamble_first_col_tl
782 {
783     >
784     {
785         \__nm_begin_of_row:

```

The contents of the cell is constructed in the box `\l_tmpa_box` because we have to compute some dimensions of this box.

```

786         \hbox_set:Nw \l_tmpa_box
787         \c_math_toggle_token
788         \bool_if:NT \l_nm_small_bool \scriptstyle
789         \l_nm_code_for_first_col_tl
790     }
791     l
792     <
793     {
794         \c_math_toggle_token
795         \hbox_set_end:
796         \__nm_actualization_for_first_and_last_row:

```

We actualise the width of the “first column” because we will use this width after the construction of the array.

```

797     \dim_gset:Nn \g__nm_width_first_col_dim
798     {
799         \dim_max:nn
800         \g__nm_width_first_col_dim
801         { \box_wd:N \l_tmpa_box }
802     }

```

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

```

803     \hbox_overlap_left:n
804     {
805         \tikz
806         [
807             remember~picture ,
808             inner~sep = \c_zero_dim ,
809             minimum~width = \c_zero_dim ,
810             baseline
811         ]
812         \node
813         [
814             anchor = base ,
815             name =
816                 nm -
817                 \int_use:N \g__nm_env_int -
818                 \int_use:N \g__nm_row_int -
819                 0 ,
820             alias =

```

```

821         \str_if_empty:NF \l__nm_name_str
822         {
823             \l__nm_name_str -
824             \int_use:N \g__nm_row_int -
825             0
826         }
827     ]
828     { \box_use:N \l_tmpa_box } ;
829     \skip_horizontal:n
830     {
831         \g__nm_left_delim_dim +
832         \l__nm_left_margin_dim +
833         \l__nm_extra_left_margin_dim
834     }
835 }
836 \skip_horizontal:n { - 2 \arraycolsep }
837 }
838 }

```

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

```

839 \tl_const:Nn \c__nm_preamble_last_col_tl
840 {
841     >
842     {

```

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

```

843         \bool_gset_true:N \g__nm_last_col_found_bool
844         \int_gincr:N \g__nm_col_int
845         \int_gset:Nn \g__nm_col_total_int
846         { \int_max:nn \g__nm_col_total_int \g__nm_col_int }

```

The contents of the cell is constructed in the box `\l_tmpa_box` because we have to compute some dimensions of this box.

```

847         \hbox_set:Nw \l_tmpa_box
848         \c_math_toggle_token
849         \bool_if:NT \l__nm_small_bool \scriptstyle
850         \l__nm_code_for_last_col_tl
851     }
852     l
853     <
854     {
855         \c_math_toggle_token
856         \hbox_set_end:
857         \__nm_actualization_for_first_and_last_row:

```

We actualise the width of the “last column” because we will use this width after the construction of the array.

```

858         \dim_gset:Nn \g__nm_width_last_col_dim
859         {
860             \dim_max:nn
861             \g__nm_width_last_col_dim
862             { \box_wd:N \l_tmpa_box }
863         }
864         \skip_horizontal:n { - 2 \arraycolsep }

```

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

```

865         \hbox_overlap_right:n
866         {
867             \skip_horizontal:n
868             {
869                 \g__nm_right_delim_dim +
870                 \l__nm_right_margin_dim +
871                 \l__nm_extra_right_margin_dim
872             }
873         \tikz
874         [
875             remember-picture ,

```

```

876         inner~sep = \c_zero_dim ,
877         minimum~width = \c_zero_dim ,
878         baseline
879     ]
880     \node
881     [
882         anchor = base ,
883         name =
884         nm -
885         \int_use:N \g__nm_env_int -
886         \int_use:N \g__nm_row_int -
887         \int_use:N \g__nm_col_int ,
888         alias =
889         \str_if_empty:NF \l__nm_name_str
890         {
891             \l__nm_name_str -
892             \int_use:N \g__nm_row_int -
893             \int_use:N \g__nm_col_int
894         }
895     ]
896     { \box_use:N \l_tmpa_box } ;
897 }
898 }
899 }

```

The environment `{NiceArray}` is constructed upon the environment `{NiceArrayWithDelims}` but, in fact, there is a flag `\l_@@_NiceArray_bool`. In `{NiceArrayWithDelims}`, some special code will be executed if this flag is raised.

```

900 \NewDocumentEnvironment { NiceArray } { }
901 {
902     \bool_set_true:N \l__nm_NiceArray_bool
903     \str_if_empty:NT \g__nm_type_env_str
904     { \str_gset:Nn \g__nm_type_env_str { NiceArray } }

```

We put `.` and `.` for the delimiters but, in fact, that doesn't matter because these arguments won't be used in `{NiceArrayWithDelims}` (because the flag `\l_@@_NiceArray_bool` is raised).

```

905     \NiceArrayWithDelims . .
906 }
907 { \endNiceArrayWithDelims }

```

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

```

908 \NewDocumentEnvironment { pNiceArray } { }
909 {
910     \str_gset:Nn \g__nm_type_env_str { pNiceArray }
911     \__nm_test_if_math_mode:
912     \NiceArrayWithDelims ( )
913 }
914 { \endNiceArrayWithDelims }

915 \NewDocumentEnvironment { bNiceArray } { }
916 {
917     \str_gset:Nn \g__nm_type_env_str { bNiceArray }
918     \__nm_test_if_math_mode:
919     \NiceArrayWithDelims [ ]
920 }
921 { \endNiceArrayWithDelims }

922 \NewDocumentEnvironment { BNiceArray } { }
923 {
924     \str_gset:Nn \g__nm_type_env_str { BNiceArray }
925     \__nm_test_if_math_mode:
926     \NiceArrayWithDelims \{ \}

```

```

927 }
928 { \endNiceArrayWithDelims }

929 \NewDocumentEnvironment { vNiceArray } { }
930 {
931   \str_gset:Nn \g__nm_type_env_str { vNiceArray }
932   \__nm_test_if_math_mode:
933   \NiceArrayWithDelims | |
934 }
935 { \endNiceArrayWithDelims }

936 \NewDocumentEnvironment { VNiceArray } { }
937 {
938   \str_gset:Nn \g__nm_type_env_str { VNiceArray }
939   \__nm_test_if_math_mode:
940   \NiceArrayWithDelims \ | \ |
941 }
942 { \endNiceArrayWithDelims }

```

15.6 The environment `{NiceMatrix}` and its variants

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 have to use `\array` and not `\begin{array}`. This command `\array` is in `\@@_array:` (we have written this command because of the redefinition of `\array` done in the classes `revtex4-1` and `revtex4-2`).

In order to execute code after the array, we use a command `\group_insert_after:N`.

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

```

943 \NewDocumentEnvironment { NiceMatrix } { ! 0 { } }
944 {
945   \str_if_empty:NT \g__nm_type_env_str
946   { \str_gset:Nn \g__nm_type_env_str { NiceMatrix } }
947   \__nm_test_if_math_mode:
948   \bool_if:NT \l__nm_in_env_bool { \__nm_fatal:n { Yet~in~env } }
949   \bool_set_true:N \l__nm_in_env_bool

```

We have to deactivate the potential externalisation of Tikz because `nicematrix` uses Tikz with `remember picture`.

```

950   \cs_if_exist:NT \tikz@library@external@loaded
951   { \tikzset { external / export = false } }

```

The instruction for actual drawing of the dotted lines must be in a `\group_insert_after:N` because the second part of the environment must be the same as in `{array}` (for the option `renew-matrix`).

```

952   \group_insert_after:N \__nm_after_array:
953   \int_gincr:N \g__nm_env_int
954   \bool_if:NF \l__nm_block_auto_columns_width_bool
955   { \dim_gzero_new:N \g__nm_max_cell_width_dim }
956   \keys_set:nm { NiceMatrix / NiceMatrix } { #1 }

```

The macro `\@@_pre_array:` is defined above (see p. 35). It is also used in `{NiceArrayWithDelims}`.

```

957   \__nm_pre_array:
958   \skip_horizontal:n { - \arraycolsep }
959   \skip_horizontal:n \l__nm_left_margin_dim
960   \skip_horizontal:n \l__nm_extra_left_margin_dim
961   \str_set:Nn \l__nm_pos_env_str c
962   \bool_set_false:N \l__nm_exterior_arraycolsep_bool
963   \__nm_array: { * \c@MaxMatrixCols C }
964 }
965 {

```

```

966 \endarray
967 \skip_horizontal:n { - \arraycolsep }

```

The two following lines are, of course, not in the second part of `{array}`, but that doesn't matter because, when `renew-matrix` is used, `\g_@@_right_margin_dim` and `\g_@@_extra_right_margin_dim` will be equal to 0 pt.

```

968 \skip_horizontal:n \g__nm_right_margin_dim
969 \skip_horizontal:n \g__nm_extra_right_margin_dim
970 }

```

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

```

971 \NewDocumentEnvironment { pNiceMatrix } { }
972 {
973   \str_gset:Nn \g__nm_type_env_str { pNiceMatrix }
974   \__nm_test_if_math_mode:
975   \left( \begin{NiceMatrix}
976   }
977   { \end{NiceMatrix} \right) }
978 \NewDocumentEnvironment { bNiceMatrix } { }
979 {
980   \str_gset:Nn \g__nm_type_env_str { bNiceMatrix }
981   \__nm_test_if_math_mode:
982   \left[ \begin{NiceMatrix}
983   }
984   { \end{NiceMatrix} \right] }
985 \NewDocumentEnvironment { BNiceMatrix } { }
986 {
987   \str_gset:Nn \g__nm_type_env_str { BNiceMatrix }
988   \__nm_test_if_math_mode:
989   \left\{ \begin{NiceMatrix}
990   }
991   { \end{NiceMatrix} \right\} }
992 \NewDocumentEnvironment { vNiceMatrix } { }
993 {
994   \str_gset:Nn \g__nm_type_env_str { vNiceMatrix }
995   \__nm_test_if_math_mode:
996   \left\lvert \begin{NiceMatrix}
997   }
998   { \end{NiceMatrix} \right\rvert }
999 \NewDocumentEnvironment { VNiceMatrix } { }
1000 {
1001   \str_gset:Nn \g__nm_type_env_str { VNiceMatrix }
1002   \__nm_test_if_math_mode:
1003   \left\lVert \begin{NiceMatrix}
1004   }
1005   { \end{NiceMatrix} \right\rVert }

```

15.7 Automatic width of the cells

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`.

```

1006 \cs_new_protected:Nn \__nm_write_max_cell_width:
1007 {

```

```

1008 \bool_if:NF \l__nm_block_auto_columns_width_bool
1009 {
1010   \iow_now:Nn \@mainaux \ExplSyntaxOn
1011   \iow_now:Nx \@mainaux
1012   {
1013     \cs_gset:cpn { __nm_max_cell_width_ \int_use:N \g__nm_env_int }
1014     { \dim_use:N \g__nm_max_cell_width_dim }
1015   }

```

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

```

1016   \str_if_empty:NF \g__nm_name_str
1017   {
1018     \iow_now:Nx \@mainaux
1019     {
1020       \cs_gset:cpn { __nm_max_cell_width_ \g__nm_name_str }
1021       { \dim_use:N \g__nm_max_cell_width_dim }
1022     }
1023   }
1024   \iow_now:Nn \@mainaux \ExplSyntaxOff
1025 }
1026 }

```

15.8 How to know whether a cell is “empty”

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

```

1027 \prg_set_conditional:Npnn \__nm_if_not_empty_cell:nn #1 #2 { T , TF }
1028 {

```

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

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

```

1029   \bool_set_false:N \l_tmpa_bool
1030   \cs_if_exist:cTF
1031   { __nm _dotted _ \int_use:N #1 - \int_use:N #2 }
1032   \prg_return_true:
1033   {

```

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

```

1034   \cs_if_free:cTF
1035   {
1036     pgf@sh@ns@nm -
1037     \int_use:N \g__nm_env_int -
1038     \int_use:N #1 -
1039     \int_use:N #2
1040   }
1041   { \prg_return_false: }
1042   {

```

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

- we don’t use a `expl3` sequence for efficiency ;

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

```

1043     \bool_set_false:N \l_tmpa_bool
1044     \cs_if_exist:cT
1045     { __nm _ empty _ \int_use:N #1 - \int_use:N #2 }
1046     {
1047         \int_compare:nNnT
1048         { \use:c { __nm _ empty _ \int_use:N #1 - \int_use:N #2 } }
1049         =
1050         \g__nm_env_int
1051         { \bool_set_true:N \l_tmpa_bool }
1052     }
1053     \bool_if:NTF \l_tmpa_bool
1054     \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.

```

1055     {
1056         \begin { pgfpicture }

```

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

```

1057         \tl_set:Nx \l_tmpa_tl
1058         {
1059             nm -
1060             \int_use:N \g__nm_env_int -
1061             \int_use:N #1 -
1062             \int_use:N #2
1063         }
1064         \pgfpointanchor \l_tmpa_tl { east }
1065         \dim_gset:Nn \g_tmpa_dim \pgf@x
1066         \pgfpointanchor \l_tmpa_tl { west }
1067         \dim_gset:Nn \g_tmpb_dim \pgf@x
1068         \end { pgfpicture }
1069         \dim_compare:nNnTF
1070         { \dim_abs:n { \g_tmpb_dim - \g_tmpa_dim } } < { 0.5 pt }
1071         \prg_return_false:
1072         \prg_return_true:
1073     }
1074 }
1075 }
1076 }

```

15.9 After the construction of the array

The macro `\@@_after_array:` is called (via `\group_insert_after:N`) in `{NiceArrayWithDelims}` and `{NiceMatrix}`.

```

1077 \cs_new_protected:Nn \__nm_after_array:
1078 {
1079     \int_compare:nNnTF \g__nm_row_int > \c_zero_int
1080     \__nm_after_array_i:
1081     { \__nm_error:n { Zero-row } }
1082 }

```

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

```

1083 \cs_new_protected:Nn \__nm_after_array_i:
1084 {
1085     \group_begin:
1086     \cs_if_exist:NT \tikz@library@external@loaded
1087     { \tikzset { external / export = false } }

```

Now, the definition of `\g_@@_col_int` and `\g_@@_col_total_int` change: `\g_@@_col_int` will be the number of columns without the “last column”; `\g_@@_col_total_int` will be the number of columns with this “last column”.²⁵

```
1088 \int_gset_eq:NN \g__nm_col_int \g__nm_col_total_int
1089 \bool_if:nT \g__nm_last_col_found_bool { \int_gdecr:N \g__nm_col_int }
```

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

```
1090 \int_gset_eq:NN \g__nm_row_total_int \g__nm_row_int
1091 \int_compare:nNnT \g__nm_last_row_int > { -1 }
1092 { \int_gsub:Nn \g__nm_row_int \c_one_int }
```

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

```
1093 \bool_if:NT \g__nm_last_row_without_value_bool
1094 {
1095   \iow_now:Nn \@mainaux \ExplSyntaxOn
1096   \iow_now:Nx \@mainaux
1097   {
1098     \cs_gset:cpn { __nm_last_row_ \int_use:N \g__nm_env_int }
1099     { \int_use:N \g__nm_row_total_int }
1100   }
}
```

If the environment has a name, we also write a value based on the name because it’s more reliable than a value based on the number of the environment.

```
1101 \str_if_empty:NF \g__nm_name_str
1102 {
1103   \iow_now:Nx \@mainaux
1104   {
1105     \cs_gset:cpn { __nm_last_row_ \g__nm_name_str }
1106     { \int_use:N \g__nm_row_total_int }
1107   }
1108 }
1109 \iow_now:Nn \@mainaux \ExplSyntaxOff
1110 }
```

By default, the diagonal lines will be parallelized²⁶. 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.

```
1111 \bool_if:NT \l__nm_parallelize_diags_bool
1112 {
1113   \int_zero_new:N \l__nm_ddots_int
1114   \int_zero_new:N \l__nm_iddotts_int
```

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

```
1115 \dim_zero_new:N \l__nm_delta_x_one_dim
1116 \dim_zero_new:N \l__nm_delta_y_one_dim
1117 \dim_zero_new:N \l__nm_delta_x_two_dim
1118 \dim_zero_new:N \l__nm_delta_y_two_dim
1119 }
```

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).

```
1120 \bool_if:NT \g__nm_extra_nodes_bool \_nm_create_extra_nodes:
1121 \int_zero_new:N \l__nm_initial_i_int
1122 \int_zero_new:N \l__nm_initial_j_int
1123 \int_zero_new:N \l__nm_final_i_int
1124 \int_zero_new:N \l__nm_final_j_int
1125 \bool_set_false:N \l__nm_initial_open_bool
1126 \bool_set_false:N \l__nm_final_open_bool
```

²⁵We remind that the potential “first column” has the number 0.

²⁶It’s possible to use the option `parallelize-diags` to disable this parallelization.

- `\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.

```
1153 \cs_new_protected:Nn \__nm_find_extremities_of_line:nmmn
1154 {
```

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

```
1155   \cs_set:cpn { __nm _ dotted _ #1 - #2 } { }
```

Initialization of variables.

```
1156   \int_set:Nn \l__nm_initial_i_int { #1 }
1157   \int_set:Nn \l__nm_initial_j_int { #2 }
1158   \int_set:Nn \l__nm_final_i_int { #1 }
1159   \int_set:Nn \l__nm_final_j_int { #2 }
```

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

```
1160   \bool_set_false:N \l__nm_stop_loop_bool
1161   \bool_do_until:Nn \l__nm_stop_loop_bool
1162   {
1163     \int_add:Nn \l__nm_final_i_int { #3 }
1164     \int_add:Nn \l__nm_final_j_int { #4 }
```

We test if we are still in the matrix.

```
1165     \bool_set_false:N \l__nm_final_open_bool
1166     \int_compare:nNnTF \l__nm_final_i_int > \g__nm_row_int
1167     {
1168       \int_compare:nNnT { #3 } = 1
1169       { \bool_set_true:N \l__nm_final_open_bool }
1170     }
1171     {
1172       \int_compare:nNnTF \l__nm_final_j_int < 1
1173       {
1174         \int_compare:nNnT { #4 } = { -1 }
1175         { \bool_set_true:N \l__nm_final_open_bool }
1176       }
1177       {
1178         \int_compare:nNnT \l__nm_final_j_int > \g__nm_col_int
1179         {
1180           \int_compare:nNnT { #4 } = 1
1181           { \bool_set_true:N \l__nm_final_open_bool }
1182         }
1183       }
1184     }
1185     \bool_if:NTF \l__nm_final_open_bool
```

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

```
1186     {
```

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).

```
1187       \int_sub:Nn \l__nm_final_i_int { #3 }
1188       \int_sub:Nn \l__nm_final_j_int { #4 }
1189       \bool_set_true:N \l__nm_stop_loop_bool
1190     }
```

If we are in the matrix, we test whether the cell is empty. If it’s not the case, we stop the loop because we have found the correct values for `\l_@@_final_i_int` and `\l_@@_final_j_int`.

```
1191     {
1192       \__nm_if_not_empty_cell:nnTF \l__nm_final_i_int \l__nm_final_j_int
1193       { \bool_set_true:N \l__nm_stop_loop_bool }
```

If the case is empty, we declare that the cell as non-empty. Indeed, we will draw a dotted line and the cell will be on that dotted line. All the cells of a dotted line have to be mark as “dotted” because we don’t want intersections between dotted lines.

```

1194         {
1195             \cs_set:cpn
1196             {
1197                 __nm _ dotted _
1198                 \int_use:N \l__nm_final_i_int -
1199                 \int_use:N \l__nm_final_j_int
1200             }
1201             { }
1202         }
1203     }
1204 }

```

We test wether the initial extremity of the dotted line is an implicit cell already dotted (by another dotted line). In this case, we can’t draw the line because we have no Tikz node at the extremity of the arrow (and we can’t use the “medium node” or the “large node” because we should use the normal node since the extremity is not open).

```

1205     \cs_if_free:cT
1206     {
1207         pgf@sh@ns@nm -
1208         \int_use:N \g__nm_env_int -
1209         \int_use:N \l__nm_final_i_int -
1210         \int_use:N \l__nm_final_j_int
1211     }
1212     {
1213         \bool_if:NF \l__nm_final_open_bool
1214         {
1215             \msg_error:nxx { nicematrix } { Impossible~line }
1216             { \int_use:N \l__nm_final_i_int }
1217             \bool_set_true:N \l__nm_impossible_line_bool
1218         }
1219     }

```

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

```

1220     \bool_set_false:N \l__nm_stop_loop_bool
1221     \bool_do_until:Nn \l__nm_stop_loop_bool
1222     {
1223         \int_sub:Nn \l__nm_initial_i_int { #3 }
1224         \int_sub:Nn \l__nm_initial_j_int { #4 }
1225         \bool_set_false:N \l__nm_initial_open_bool
1226         \int_compare:nNnTF \l__nm_initial_i_int < 1
1227         {
1228             \int_compare:nNnT { #3 } = 1
1229             { \bool_set_true:N \l__nm_initial_open_bool }
1230         }
1231         {
1232             \int_compare:nNnTF \l__nm_initial_j_int < 1
1233             {
1234                 \int_compare:nNnT { #4 } = 1
1235                 { \bool_set_true:N \l__nm_initial_open_bool }
1236             }
1237             {
1238                 \int_compare:nNnT \l__nm_initial_j_int > \g__nm_col_int
1239                 {
1240                     \int_compare:nNnT { #4 } = { -1 }
1241                     { \bool_set_true:N \l__nm_initial_open_bool }
1242                 }
1243             }
1244         }

```

```

1245     \bool_if:NTF \l__nm_initial_open_bool
1246     {
1247         \int_add:Nn \l__nm_initial_i_int { #3 }
1248         \int_add:Nn \l__nm_initial_j_int { #4 }
1249         \bool_set_true:N \l__nm_stop_loop_bool
1250     }
1251     {
1252         \__nm_if_not_empty_cell:nmTF
1253         \l__nm_initial_i_int \l__nm_initial_j_int
1254         { \bool_set_true:N \l__nm_stop_loop_bool }
1255         {
1256             \cs_set:cpn
1257             {
1258                 __nm_dotted_
1259                 \int_use:N \l__nm_initial_i_int -
1260                 \int_use:N \l__nm_initial_j_int
1261             }
1262             { }
1263         }
1264     }
1265 }

```

We test whether the initial extremity of the dotted line is an implicit cell already dotted (by another dotted line). In this case, we can't draw the line because we have no Tikz node at the extremity of the arrow (and we can't use the "medium node" or the "large node" because we should use the normal node since the extremity is not open).

```

1266     \cs_if_free:cT
1267     {
1268         pgf@sh@ns@nm -
1269         \int_use:N \g__nm_env_int -
1270         \int_use:N \l__nm_initial_i_int -
1271         \int_use:N \l__nm_initial_j_int
1272     }
1273     {
1274         \bool_if:NF \l__nm_initial_open_bool
1275         {
1276             \msg_error:nmx { nicematrix } { Impossible~line }
1277             { \int_use:N \l__nm_initial_i_int }
1278             \bool_set_true:N \l__nm_impossible_line_bool
1279         }
1280     }

```

If we have at least one open extremity, we create the "medium nodes" in the matrix²⁷. We remind that, when used once, the command `\@@_create_extra_nodes:` becomes no-op in the current TeX group.

```

1281     \bool_if:NF \l__nm_initial_open_bool \__nm_create_extra_nodes:
1282     \bool_if:NT \l__nm_final_open_bool \__nm_create_extra_nodes:
1283 }

```

The command `\@@_retrieve_coords:nm` retrieves the Tikz coordinates of the two extremities of the dotted line we will have to draw²⁸. 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:nm` are the suffix 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}`.

```

1284 \cs_new_protected:Nn \__nm_retrieve_coords:nm

```

²⁷We should change this. Indeed, for an open extremity of an *horizontal* dotted line, we use the `w` node, if, it exists, and not the "medium node".

²⁸In fact, with diagonal lines, or vertical lines in columns of type L or R, an adjustment of one of the coordinates may be done.

```

1285 {
1286   \dim_gzero_new:N \g__nm_x_initial_dim
1287   \dim_gzero_new:N \g__nm_y_initial_dim
1288   \dim_gzero_new:N \g__nm_x_final_dim
1289   \dim_gzero_new:N \g__nm_y_final_dim
1290   \begin { tikzpicture } [ remember-picture ]
1291     \tikz@parse@node \pgfutil@firstofone
1292     ( nm - \int_use:N \g__nm_env_int -
1293       \int_use:N \l__nm_initial_i_int -
1294       \int_use:N \l__nm_initial_j_int #1 )
1295     \dim_gset:Nn \g__nm_x_initial_dim \pgf@x
1296     \dim_gset:Nn \g__nm_y_initial_dim \pgf@y
1297     \tikz@parse@node \pgfutil@firstofone
1298     ( nm - \int_use:N \g__nm_env_int -
1299       \int_use:N \l__nm_final_i_int -
1300       \int_use:N \l__nm_final_j_int #2 )
1301     \dim_gset:Nn \g__nm_x_final_dim \pgf@x
1302     \dim_gset:Nn \g__nm_y_final_dim \pgf@y
1303   \end { tikzpicture }
1304 }
1305 \cs_generate_variant:Nn \__nm_retrieve_coords:nn { x x }

```

```

1306 \cs_new_protected:Nn \__nm_draw_Ldots:nn
1307 {
1308   \cs_if_free:cT { __nm _ dotted _ #1 - #2 }
1309   {
1310     \bool_set_false:N \l__nm_impossible_line_bool
1311     \__nm_find_extremities_of_line:nnnn { #1 } { #2 } \c_zero_int \c_one_int
1312     \bool_if:NF \l__nm_impossible_line_bool \__nm_actually_draw_Ldots:
1313   }
1314 }

```

The command `\@@_actually_draw_Ldots:` 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`.

```

1315 \cs_new_protected:Nn \__nm_actually_draw_Ldots:
1316 {
1317   \__nm_retrieve_coords:xx
1318   {
1319     \bool_if:NTF \l__nm_initial_open_bool
1320     {

```

If a `w` node exists (created when the key `columns-width` is used), we use the `w` node for the extremity.

```

1321     \cs_if_exist:cTF
1322     {
1323       pgf@sh@ns@nm
1324       - \int_use:N \g__nm_env_int
1325       - \int_use:N \l__nm_initial_i_int
1326       - \int_use:N \l__nm_initial_j_int - w
1327     }
1328     { - w.base~west }
1329     { - medium.base~west }
1330   }
1331   { .base~east }
1332 }
1333 {
1334   \bool_if:NTF \l__nm_final_open_bool
1335   {
1336     \cs_if_exist:cTF
1337     {
1338       pgf@sh@ns@nm
1339       - \int_use:N \g__nm_env_int
1340       - \int_use:N \l__nm_final_i_int

```

```

1341         - \int_use:N \l__nm_final_j_int - w
1342     }
1343     { - w.base~east }
1344     { - medium.base~east }
1345 }
1346 { .base~west }
1347 }
1348 \bool_if:NT \l__nm_initial_open_bool
1349   { \dim_gset_eq:NN \g__nm_y_initial_dim \g__nm_y_final_dim }
1350 \bool_if:NT \l__nm_final_open_bool
1351   { \dim_gset_eq:NN \g__nm_y_final_dim \g__nm_y_initial_dim }

```

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

```

1352 \dim_gadd:Nn \g__nm_y_initial_dim { 0.53 pt }
1353 \dim_gadd:Nn \g__nm_y_final_dim { 0.53 pt }
1354 \__nm_draw_tikz_line:
1355 }

```

```

1356 \cs_new_protected:Nn \__nm_draw_Cdots:nn
1357 {
1358   \cs_if_free:cT { __nm _ dotted _ #1 - #2 }
1359   {
1360     \bool_set_false:N \l__nm_impossible_line_bool
1361     \__nm_find_extremities_of_line:nnnn { #1 } { #2 } \c_zero_int \c_one_int
1362     \bool_if:NF \l__nm_impossible_line_bool
1363     {
1364       \__nm_retrieve_coords:xx
1365       {
1366         \bool_if:NTF \l__nm_initial_open_bool
1367         {
1368           \cs_if_exist:cTF
1369           {
1370             pgf@sh@ns@nm
1371             - \int_use:N \g__nm_env_int
1372             - \int_use:N \l__nm_initial_i_int
1373             - \int_use:N \l__nm_initial_j_int - w
1374           }
1375           { - w.mid~west }
1376           { - medium.mid~west }
1377         }
1378         { .mid~east }
1379       }
1380     }
1381     \bool_if:NTF \l__nm_final_open_bool
1382     {
1383       \cs_if_exist:cTF
1384       {
1385         pgf@sh@ns@nm
1386         - \int_use:N \g__nm_env_int
1387         - \int_use:N \l__nm_final_i_int
1388         - \int_use:N \l__nm_final_j_int - w
1389       }
1390       { - w.mid~east }
1391       { - medium.mid~east }
1392     }
1393     { .mid~west }
1394   }
1395   \bool_if:NT \l__nm_initial_open_bool
1396     { \dim_gset_eq:NN \g__nm_y_initial_dim \g__nm_y_final_dim }
1397   \bool_if:NT \l__nm_final_open_bool
1398     { \dim_gset_eq:NN \g__nm_y_final_dim \g__nm_y_initial_dim }
1399   \__nm_draw_tikz_line:

```



```

1400     }
1401   }
1402 }

```

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?).

```

1403 \cs_new_protected:Nn \__nm_draw_Vdots:nn
1404   {
1405     \cs_if_free:cT { __nm _ dotted _ #1 - #2 }
1406     {
1407       \bool_set_false:N \l__nm_impossible_line_bool
1408       \__nm_find_extremities_of_line:nnnn { #1 } { #2 } \c_one_int \c_zero_int
1409       \bool_if:NF \l__nm_impossible_line_bool
1410       {
1411         \__nm_retrieve_coords:xx
1412         {
1413           \bool_if:NTF \l__nm_initial_open_bool
1414             { - medium.north-west }
1415             { .south~west }
1416         }
1417         {
1418           \bool_if:NTF \l__nm_final_open_bool
1419             { - medium.south-west }
1420             { .north~west }
1421         }
1422       }
1423     }
1424   }

```

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

```

1422     \bool_set:Nn \l_tmpa_bool
1423       { \dim_compare_p:nNn \g__nm_x_initial_dim = \g__nm_x_final_dim }
1424     \__nm_retrieve_coords:xx
1425     {
1426       \bool_if:NTF \l__nm_initial_open_bool
1427         { - medium.north }
1428         { .south }
1429     }
1430     {
1431       \bool_if:NTF \l__nm_final_open_bool
1432         { - medium.south }
1433         { .north }
1434     }
1435   }

```

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

```

1435     \bool_set:Nn \l_tmpb_bool
1436       { \dim_compare_p:nNn \g__nm_x_initial_dim = \g__nm_x_final_dim }
1437     \bool_if:NF \l_tmpb_bool
1438     {
1439       \dim_gset:Nn \g__nm_x_initial_dim
1440       {
1441         \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
1442           \g__nm_x_initial_dim \g__nm_x_final_dim
1443       }
1444       \dim_gset_eq:NN \g__nm_x_final_dim \g__nm_x_initial_dim
1445     }
1446     \__nm_draw_tikz_line:
1447   }
1448 }
1449 }

```

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.

```

1450 \cs_new_protected:Nn \__nm_draw_Ddots:nn
1451 {
1452   \cs_if_free:cT { __nm _ dotted _ #1 - #2 }
1453   {
1454     \bool_set_false:N \l__nm_impossible_line_bool
1455     \__nm_find_extremities_of_line:nnnn { #1 } { #2 } \c_one_int \c_one_int
1456     \bool_if:NF \l__nm_impossible_line_bool
1457     {
1458       \__nm_retrieve_coords:xx
1459       {
1460         \bool_if:NTF \l__nm_initial_open_bool
1461         { - medium.north~west }
1462         { .south~east }
1463       }
1464       {
1465         \bool_if:NTF \l__nm_final_open_bool
1466         { - medium.south~east }
1467         { .north~west }
1468       }
1469     }
1470   }

```

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.

```

1469     \bool_if:NT \l__nm_parallelize_diags_bool
1470     {
1471       \int_incr:N \l__nm_ddots_int

```

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

```

1472     \int_compare:nNnTF \l__nm_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 Δ_x and the Δ_y of the line because these values will be used to draw the others diagonal lines parallels to the first one.

```

1473     {
1474       \dim_set:Nn \l__nm_delta_x_one_dim
1475       { \g__nm_x_final_dim - \g__nm_x_initial_dim }
1476       \dim_set:Nn \l__nm_delta_y_one_dim
1477       { \g__nm_y_final_dim - \g__nm_y_initial_dim }
1478     }

```

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`.

```

1479     {
1480       \dim_gset:Nn \g__nm_y_final_dim
1481       {
1482         \g__nm_y_initial_dim +
1483         ( \g__nm_x_final_dim - \g__nm_x_initial_dim ) *
1484         \dim_ratio:nn \l__nm_delta_y_one_dim \l__nm_delta_x_one_dim
1485       }
1486     }
1487   }

```

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

```

1488     \__nm_draw_tikz_line:
1489   }
1490 }
1491 }

```

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

```

1492 \cs_new_protected:Nn \__nm_draw_Iddots:nn
1493 {
1494   \cs_if_free:cT { __nm _ dotted _ #1 - #2 }

```

```

1495 {
1496   \bool_set_false:N \l__nm_impossible_line_bool
1497   \__nm_find_extremities_of_line:nmm { #1 } { #2 } 1 { -1 }
1498   \bool_if:NF \l__nm_impossible_line_bool
1499   {
1500     \__nm_retrieve_coords:xx
1501     {
1502       \bool_if:NTF \l__nm_initial_open_bool
1503         { - medium.north-east }
1504         { .south-west }
1505     }
1506     {
1507       \bool_if:NTF \l__nm_final_open_bool
1508         { - medium.south-west }
1509         { .north-east }
1510     }
1511     \bool_if:NT \l__nm_parallelize_diags_bool
1512     {
1513       \int_incr:N \l__nm_iddots_int
1514       \int_compare:nNnTF \l__nm_iddots_int = \c_one_int
1515       {
1516         \dim_set:Nn \l__nm_delta_x_two_dim
1517           { \g__nm_x_final_dim - \g__nm_x_initial_dim }
1518         \dim_set:Nn \l__nm_delta_y_two_dim
1519           { \g__nm_y_final_dim - \g__nm_y_initial_dim }
1520       }
1521       {
1522         \dim_gset:Nn \g__nm_y_final_dim
1523           {
1524             \g__nm_y_initial_dim +
1525             ( \g__nm_x_final_dim - \g__nm_x_initial_dim ) *
1526             \dim_ratio:nn \l__nm_delta_y_two_dim \l__nm_delta_x_two_dim
1527           }
1528       }
1529     }
1530     \__nm_draw_tikz_line:
1531   }
1532 }
1533 }

```

15.10 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`.

```

1534 \cs_new_protected:Nn \__nm_draw_tikz_line:
1535 {

```

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

```

1536   \dim_zero_new:N \l__nm_l_dim
1537   \dim_set:Nn \l__nm_l_dim
1538     {
1539     \fp_to_dim:n
1540       {
1541         sqrt
1542         (
1543           ( \dim_use:N \g__nm_x_final_dim
1544             - \dim_use:N \g__nm_x_initial_dim
1545           ) ^ 2
1546         +

```

```

1547         ( \dim_use:N \g__nm_y_final_dim
1548         - \dim_use:N \g__nm_y_initial_dim
1549         ) ^ 2
1550     )
1551 }
1552 }

```

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

```

1553 \dim_compare:nNnF \l__nm_l_dim = \c_zero_dim

```

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

```

1554 {
1555     \bool_if:NTF \l__nm_initial_open_bool
1556     {
1557         \bool_if:NTF \l__nm_final_open_bool
1558         {
1559             \int_set:Nn \l_tmpa_int
1560             { \dim_ratio:nn \l__nm_l_dim \l__nm_inter_dots_dim }
1561         }
1562         {
1563             \int_set:Nn \l_tmpa_int
1564             { \dim_ratio:nn { \l__nm_l_dim - 0.3 em } \l__nm_inter_dots_dim }
1565         }
1566     }
1567     {
1568         \bool_if:NTF \l__nm_final_open_bool
1569         {
1570             \int_set:Nn \l_tmpa_int
1571             { \dim_ratio:nn { \l__nm_l_dim - 0.3 em } \l__nm_inter_dots_dim }
1572         }
1573         {
1574             \int_set:Nn \l_tmpa_int
1575             { \dim_ratio:nn { \l__nm_l_dim - 0.6 em } \l__nm_inter_dots_dim }
1576         }
1577     }

```

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

```

1578     \dim_set:Nn \l_tmpa_dim
1579     {
1580         ( \g__nm_x_final_dim - \g__nm_x_initial_dim ) *
1581         \dim_ratio:nn \l__nm_inter_dots_dim \l__nm_l_dim
1582     }
1583     \dim_set:Nn \l_tmpb_dim
1584     {
1585         ( \g__nm_y_final_dim - \g__nm_y_initial_dim ) *
1586         \dim_ratio:nn \l__nm_inter_dots_dim \l__nm_l_dim
1587     }

```

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

```

1588     \int_set:Nn \l_tmpb_int
1589     {
1590         \bool_if:NTF \l__nm_initial_open_bool
1591         { \bool_if:NTF \l__nm_final_open_bool 1 0 }
1592         { \bool_if:NTF \l__nm_final_open_bool 2 1 }
1593     }

```

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.

```

1594     \dim_gadd:Nn \g__nm_x_initial_dim

```

```

1595     {
1596       ( \g__nm_x_final_dim - \g__nm_x_initial_dim ) *
1597       \dim_ratio:nn
1598       { \l__nm_l_dim - \l__nm_inter_dots_dim * \l_tmpa_int } { \l__nm_l_dim * 2 } *
1599       \l_tmpb_int
1600     }

```

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

```

1601     \dim_gadd:Nn \g__nm_y_initial_dim
1602     {
1603       ( \g__nm_y_final_dim - \g__nm_y_initial_dim ) *
1604       \dim_ratio:nn
1605       { \l__nm_l_dim - \l__nm_inter_dots_dim * \l_tmpa_int }
1606       { \l__nm_l_dim * 2 } *
1607       \l_tmpb_int
1608     }
1609     \begin { tikzpicture } [ overlay ]
1610       \int_step_inline:nnnn 0 1 \l_tmpa_int
1611       {
1612         \pgfpathcircle
1613         { \pgfpoint { \g__nm_x_initial_dim } { \g__nm_y_initial_dim } }
1614         { \l__nm_radius_dim }
1615         \pgfusepath { fill }
1616         \dim_gadd:Nn \g__nm_x_initial_dim \l_tmpa_dim
1617         \dim_gadd:Nn \g__nm_y_initial_dim \l_tmpb_dim
1618       }
1619     \end { tikzpicture }
1620   }
1621 }

```

15.11 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).

```

1622 \cs_set_eq:NN \__nm_ldots \ldots
1623 \cs_set_eq:NN \__nm_cdots \cdots
1624 \cs_set_eq:NN \__nm_vdots \vdots
1625 \cs_set_eq:NN \__nm_ddots \ddots
1626 \cs_set_eq:NN \__nm_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).

```

1627 \cs_new_protected:Nn \__nm_add_to_empty_cells:
1628   {
1629     \cs_gset:cpx
1630     { __nm _ empty _ \int_use:N \g__nm_row_int - \int_use:N \g__nm_col_int }
1631     { \int_use:N \g__nm_env_int }
1632   }

```

The commands `\@@_Ldots`, `\@@_Cdots`, `\@@_Vdots`, `\@@_Ddots` and `\@@_Iddots` will be linked to `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots` and `\Iddots` in the environments `{NiceArray}` (the other environments of `nicematrix` rely upon `{NiceArray}`).

The starred versions of these commands are deprecated since version 3.1 but they are still available.

```

1633 \NewDocumentCommand \__nm_Ldots { s }
1634   {
1635     \bool_if:nF { #1 } { \__nm_instruction_of_type:n { Ldots } }
1636     \bool_if:NF \l__nm_nullify_dots_bool { \phantom \__nm_ldots }
1637     \__nm_add_to_empty_cells:
1638   }

```

```

1639 \NewDocumentCommand \_nm_Cdots { s }
1640 {
1641   \bool_if:nF { #1 } { \_nm_instruction_of_type:n { Cdots } }
1642   \bool_if:NF \l\_nm_nullify_dots_bool { \phantom \_nm_cdots }
1643   \_nm_add_to_empty_cells:
1644 }

1645 \NewDocumentCommand \_nm_Vdots { s }
1646 {
1647   \bool_if:nF { #1 } { \_nm_instruction_of_type:n { Vdots } }
1648   \bool_if:NF \l\_nm_nullify_dots_bool { \phantom \_nm_vdots }
1649   \_nm_add_to_empty_cells:
1650 }

1651 \NewDocumentCommand \_nm_Ddots { s }
1652 {
1653   \bool_if:nF { #1 } { \_nm_instruction_of_type:n { Ddots } }
1654   \bool_if:NF \l\_nm_nullify_dots_bool { \phantom \_nm_ddots }
1655   \_nm_add_to_empty_cells:
1656 }

1657 \NewDocumentCommand \_nm_Iddots { s }
1658 {
1659   \bool_if:nF { #1 } { \_nm_instruction_of_type:n { Iddots } }
1660   \bool_if:NF \l\_nm_nullify_dots_bool { \phantom \_nm_iddots }
1661   \_nm_add_to_empty_cells:
1662 }

```

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

```

1663 \cs_new_protected:Nn \_nm_Hspace:
1664 {
1665   \_nm_add_to_empty_cells:
1666   \hspace
1667 }

```

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

```

1668 \cs_set_eq:NN \_nm_old_multicolumn \multicolumn
1669 \cs_new:Npn \_nm_multicolumn:nnn #1 #2 #3
1670 {
1671   \_nm_old_multicolumn { #1 } { #2 } { #3 }
1672   \int_compare:nNnT #1 > 1
1673   {
1674     \seq_gput_left:Nx \g\_nm_multicolumn_cells_seq
1675     { \int_eval:n \g\_nm_row_int - \int_use:N \g\_nm_col_int }
1676     \seq_gput_left:Nn \g\_nm_multicolumn_sizes_seq { #1 }
1677   }
1678   \int_gadd:Nn \g\_nm_col_int { #1 - 1 }
1679 }

```

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`.

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

```

1680 \cs_new:Npn \_nm_Hdotsfor:
1681 {
1682   \multicolumn { 1 } { C } { }
1683   \_nm_Hdotsfor_i
1684 }

```

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

```

1685 \bool_if:NTF \c_nm_draft_bool
1686   {
1687     \NewDocumentCommand \_nm_Hdotsfor_i { 0 { } m }
1688       { \prg_replicate:nm { #2 - 1 } { & \multicolumn { 1 } { C } { } } }
1689   }
1690   {
1691     \NewDocumentCommand \_nm_Hdotsfor_i { 0 { } m }
1692       {
1693         \tl_gput_right:Nx \g__nm_Hdotsfor_lines_tl
1694           {
1695             \_nm_draw_Hdotsfor:nnn
1696               { \int_use:N \g__nm_row_int }
1697               { \int_use:N \g__nm_col_int }
1698               { #2 }
1699           }
1700         \prg_replicate:nm { #2 - 1 } { & \multicolumn { 1 } { C } { } }
1701       }
1702   }

1703 \cs_new_protected:Nn \_nm_draw_Hdotsfor:nnn
1704   {
1705     \bool_set_false:N \l__nm_initial_open_bool
1706     \bool_set_false:N \l__nm_final_open_bool

```

For the row, it's easy.

```

1707   \int_set:Nn \l__nm_initial_i_int { #1 }
1708   \int_set:Nn \l__nm_final_i_int { #1 }

```

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

```

1709   \int_compare:nNnTF #2 = 1
1710     {
1711       \int_set:Nn \l__nm_initial_j_int 1
1712       \bool_set_true:N \l__nm_initial_open_bool
1713     }
1714     {
1715       \int_set:Nn \l_tmpa_int { #2 - 1 }
1716       \_nm_if_not_empty_cell:nnTF \l__nm_initial_i_int \l_tmpa_int
1717         { \int_set:Nn \l__nm_initial_j_int { #2 - 1 } }
1718         {
1719           \int_set:Nn \l__nm_initial_j_int {#2}
1720           \bool_set_true:N \l__nm_initial_open_bool
1721         }
1722     }
1723   \int_compare:nNnTF { #2 + #3 - 1 } = \g__nm_col_int
1724     {
1725       \int_set:Nn \l__nm_final_j_int { #2 + #3 - 1 }
1726       \bool_set_true:N \l__nm_final_open_bool
1727     }
1728     {
1729       \int_set:Nn \l_tmpa_int { #2 + #3 }
1730       \_nm_if_not_empty_cell:nnTF \l__nm_final_i_int \l_tmpa_int
1731         { \int_set:Nn \l__nm_final_j_int { #2 + #3 } }
1732         {
1733           \int_set:Nn \l__nm_final_j_int { #2 + #3 - 1 }
1734           \bool_set_true:N \l__nm_final_open_bool
1735         }
1736     }
1737   \bool_if:nT { \l__nm_initial_open_bool || \l__nm_final_open_bool }
1738     \_nm_create_extra_nodes:
1739     \_nm_actually_draw_Ldots:

```

We declare all the cells concerned by the `\Hdotsfor` as “dotted” (for the dotted lines created by `\Cdots`, `\Ldots`, etc., this job is done by `\@@_find_extremities_of_line:nnnn`). This declaration is done by defining a special control sequence (to nil).

```

1740   \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
1741     { \cs_set:cpn { __nm _ dotted _ #1 - ##1 } { } }
1742   }

```

15.12 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.

```

1743 \cs_new_protected:Nn \__nm_line:nn
1744   {
1745     \dim_zero_new:N \g__nm_x_initial_dim
1746     \dim_zero_new:N \g__nm_y_initial_dim
1747     \dim_zero_new:N \g__nm_x_final_dim
1748     \dim_zero_new:N \g__nm_y_final_dim
1749     \bool_set_false:N \l__nm_initial_open_bool
1750     \bool_set_false:N \l__nm_final_open_bool
1751     \begin { tikzpicture }
1752       \path~(#1)~---(#2)~node[at~start]~(i)~{}~node[at~end]~(f)~{} ;
1753       \tikz@parse@node \pgfutil@firstofone ( i )
1754       \dim_gset:Nn \g__nm_x_initial_dim \pgf@x
1755       \dim_gset:Nn \g__nm_y_initial_dim \pgf@y
1756       \tikz@parse@node \pgfutil@firstofone ( f )
1757       \dim_gset:Nn \g__nm_x_final_dim \pgf@x
1758       \dim_gset:Nn \g__nm_y_final_dim \pgf@y
1759     \end { tikzpicture }
1760     \__nm_draw_tikz_line:
1761   }

```

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).

15.13 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 position²⁹ 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.

```

1762 \cs_generate_variant:Nn \dim_set:Nn { N v }

```

Some extensions, like the extension `doc`, do a redefinition of the command `\dotfill` of LaTeX. That’s why we define a command `\@@_dotfill:` as we wish. We test whether we are in draft mode because, in this case, we don’t draw the dotted lines.

```

1763 \bool_if:NTF \c__nm_draft_bool
1764   { \cs_set_eq:NN \__nm_dotfill: \prg_do_nothing: }
1765   {
1766     \cs_set:Npn \__nm_dotfill:
1767       {

```

²⁹In 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`.

If the option `small` is used, we change the space between two dots (we can't use `\l_@@_inter_dots_dim` which will be set after the construction of the array). We can't put the `\bool_if:NT` in the first argument of `\hbox_to_wd:nn` because `\cleaders` is a special TeX primitive.

```

1768     \bool_if:NT \l__nm_small_bool
1769     { \dim_set:Nn \l__nm_inter_dots_dim { 0.25 em } }
1770     \cleaders
1771     \hbox_to_wd:nn
1772     { \l__nm_inter_dots_dim }
1773     {
1774         \c_math_toggle_token
1775         \bool_if:NT \l__nm_small_bool \scriptstyle
1776         \hss . \hss
1777         \c_math_toggle_token
1778     }
1779     \hfill
1780     \skip_horizontal:n \c_zero_dim
1781 }
1782 }

```

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

```

1783 \cs_new:Npn \__nm_hdottedline:
1784 {
1785     \noalign
1786     {
1787         \bool_gset_true:N \g__nm_extra_nodes_bool
1788         \cs_if_exist:cTF { __nm_width_ \int_use:N \g__nm_env_int }
1789         { \dim_set:Nv \l_tmpa_dim { __nm_width_ \int_use:N \g__nm_env_int } }
1790         { \dim_set:Nn \l_tmpa_dim { 5 mm } }
1791         \hbox_overlap_right:n
1792         {
1793             \hbox_to_wd:nn
1794             {
1795                 \l_tmpa_dim + 2 \arraycolsep
1796                 - \l__nm_left_margin_dim - \g__nm_right_margin_dim
1797             }
1798             \__nm_dotfill:
1799         }
1800     }
1801 }

```

```

1802 \cs_new_protected:Nn \__nm_vdottedline:n
1803 {

```

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

```

1804     \int_compare:nNnTF #1 = \c_zero_int
1805     { \__nm_error:n { Use-of~::~in~first~position } }
1806     {
1807         \__nm_create_extra_nodes:
1808         \bool_if:NF \c__nm_draft_bool
1809         {
1810             \dim_zero_new:N \g__nm_x_initial_dim
1811             \dim_zero_new:N \g__nm_y_initial_dim
1812             \dim_zero_new:N \g__nm_x_final_dim
1813             \dim_zero_new:N \g__nm_y_final_dim
1814             \bool_set_true:N \l__nm_initial_open_bool
1815             \bool_set_true:N \l__nm_final_open_bool

```

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

```

1816     \begin { tikzpicture } [ remember~picture ]
1817     \tikz@parse@node\pgfutil@firstofone
1818     ( 1 - #1 - large .north-east )
1819     \dim_gset:Nn \g__nm_x_initial_dim \pgf@x

```

```

1820     \dim_gset:Nn \g__nm_y_initial_dim \pgf@y
1821     \tikz@parse@node\pgfutil@firstofone
1822     ( \int_use:N \g__nm_row_int - #1 - large .south~east )
1823     \dim_gset:Nn \g__nm_x_final_dim \pgf@x
1824     \dim_gset:Nn \g__nm_y_final_dim \pgf@y
1825     \end { tikzpicture }

```

However, if the w-nodes are created in the previous column (that is if the previous column was constructed explicitly or implicitly³⁰ 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`.

```

1826     \cs_if_exist:cT
1827     { pgf@sh@ns@nm -\int_use:N \g__nm_env_int - 1 - #1 - w }
1828     {
1829         \begin { tikzpicture } [ remember~picture ]
1830         \tikz@parse@node\pgfutil@firstofone
1831         ( 1 - #1 - w .north~east )
1832         \dim_gset:Nn \g__nm_x_initial_dim \pgf@x
1833         \tikz@parse@node\pgfutil@firstofone
1834         ( \int_use:N \g__nm_row_int - #1 - w .south~east )
1835         \dim_gset:Nn \g__nm_x_final_dim \pgf@x
1836         \end { tikzpicture }
1837         \dim_gadd:Nn \g__nm_x_initial_dim \arraycolsep
1838         \dim_gadd:Nn \g__nm_x_final_dim \arraycolsep
1839     }
1840     \__nm_draw_tikz_line:
1841 }
1842 }
1843 }

```

15.14 The vertical rules

We don't want that a vertical rule drawn by the specifier “|” extends in the eventual “first row” and “last row” of the array.

The natural way to do that would be to redefine the specifier “|” with `\newcolumntype`:

```

\newcolumntype { | }
{ ! { \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 specifiers “|” (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 tests must be effective in each row and not once when the preamble is constructed).

```

1844 \cs_new_protected:Npn \__nm_vline:
1845 {
1846     \int_compare:nNnTF \l__nm_first_col_int = \c_zero_int
1847     {
1848         \int_compare:nNnTF \g__nm_col_int = \c_zero_int
1849         {
1850             \int_compare:nNnTF \l__nm_first_row_int = \c_zero_int
1851             {
1852                 \int_compare:nNnF \g__nm_row_int = \c_zero_int
1853                 {
1854                     \int_compare:nNnF \g__nm_row_int = \l__nm_last_row_int
1855                     \__nm_vline_i:

```

³⁰A 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`.

```

1856     }
1857   }
1858   {
1859     \int_compare:nNnF \g__nm_row_int = \c_zero_int
1860     {
1861       \int_compare:nNnF \g__nm_row_int = \l__nm_last_row_int
1862       \__nm_vline_i:
1863     }
1864   }
1865 }
1866 {
1867   \int_compare:nNnF \g__nm_row_int = \c_zero_int
1868   {
1869     \int_compare:nNnF \g__nm_row_int = \l__nm_last_row_int
1870     \__nm_vline_i:
1871   }
1872 }
1873 }
1874 {
1875   \int_compare:nNnTF \g__nm_col_int = \c_zero_int
1876   {
1877     \int_compare:nNnF \g__nm_row_int = { -1 }
1878     {
1879       \int_compare:nNnF \g__nm_row_int = { \l__nm_last_row_int - 1 }
1880       \__nm_vline_i:
1881     }
1882   }
1883   {
1884     \int_compare:nNnF \g__nm_row_int = \c_zero_int
1885     {
1886       \int_compare:nNnF \g__nm_row_int = \l__nm_last_row_int
1887       \__nm_vline_i:
1888     }
1889   }
1890 }
1891 }

```

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

```

1892 \cs_set_eq:NN \__nm_vline_i: \vline

```

15.15 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.

```

1893 \bool_new:N \l__nm_block_auto_columns_width_bool

```

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

```

1894 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
1895   {
1896     auto-columns-width .code:n =
1897     {
1898       \bool_set_true:N \l__nm_block_auto_columns_width_bool
1899       \dim_gzero_new:N \g__nm_max_cell_width_dim
1900       \bool_set_true:N \l__nm_auto_columns_width_bool
1901     }
1902   }

```

```

1903 \NewDocumentEnvironment { NiceMatrixBlock } { ! 0 { } }

```

```

1904 {
1905   \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
1906   \int_zero_new:N \l__nm_first_env_block_int
1907   \int_set:Nn \l__nm_first_env_block_int { \g__nm_env_int + 1 }
1908 }

```

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`).

```

1909 {
1910   \bool_if:NT \l__nm_block_auto_columns_width_bool
1911   {
1912     \iow_now:Nn \@mainaux \ExplSyntaxOn
1913     \int_step_inline:nnnn \l__nm_first_env_block_int 1 \g__nm_env_int
1914     {
1915       \iow_now:Nx \@mainaux
1916       {
1917         \cs_gset:cpn { __nm_max_cell_width_ ##1 }
1918         { \dim_use:N \g__nm_max_cell_width_dim }
1919       }
1920     }
1921     \iow_now:Nn \@mainaux \ExplSyntaxOff
1922   }
1923 }

```

15.16 The extra nodes

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

```

1924 \cs_generate_variant:Nn \dim_min:nn { v n }
1925 \cs_generate_variant:Nn \dim_max:nn { v n }

```

The macro `\@@_create_extra_nodes:` must *not* be used in the `code-after` because the `code-after` is executed in a scope of `prefix name`.

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

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

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

```

1926 \cs_new_protected:Nn \__nm_create_extra_nodes:
1927 {
1928   \begin { tikzpicture } [ remember-picture , overlay ]
1929   \int_step_variable:nnNn \g__nm_first_row_int \g__nm_row_total_int \__nm_i:
1930   {
1931     \dim_zero_new:c { l__nm_row_\__nm_i: _min_dim }
1932     \dim_set_eq:cN { l__nm_row_\__nm_i: _min_dim } \c_max_dim
1933     \dim_zero_new:c { l__nm_row_\__nm_i: _max_dim }
1934     \dim_set:cn { l__nm_row_\__nm_i: _max_dim } { - \c_max_dim }
1935   }
1936   \int_step_variable:nnNn \g__nm_first_col_int \g__nm_col_total_int \__nm_j:
1937   {
1938     \dim_zero_new:c { l__nm_column_\__nm_j: _min_dim }
1939     \dim_set_eq:cN { l__nm_column_\__nm_j: _min_dim } \c_max_dim
1940     \dim_zero_new:c { l__nm_column_\__nm_j: _max_dim }
1941     \dim_set:cn { l__nm_column_\__nm_j: _max_dim } { - \c_max_dim }
1942   }

```

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

```

1943     \int_step_variable:nnNn \g__nm_first_row_int \g__nm_row_total_int \_nm_i:
1944     {
1945         \int_step_variable:nnNn
1946         \g__nm_first_col_int \g__nm_col_total_int \_nm_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.

```

1947         { \cs_if_exist:cT
1948           { pgf@sh@ns@nm - \int_use:N \g__nm_env_int - \_nm_i: - \_nm_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`.

```

1949         {
1950             \tikz@parse@node \pgfutil@firstofone
1951             ( nm - \int_use:N \g__nm_env_int
1952               - \_nm_i: - \_nm_j: .south-west )
1953         \dim_set:cn { l__nm_row\_nm_i: _min_dim}
1954         { \dim_min:vn { l__nm_row _ \_nm_i: _min_dim } \pgf@y }
1955         \seq_if_in:NxF \g__nm_multicolumn_cells_seq { \_nm_i: - \_nm_j: }
1956         {
1957             \dim_set:cn { l__nm_column _ \_nm_j: _min_dim}
1958             { \dim_min:vn { l__nm_column _ \_nm_j: _min_dim } \pgf@x }
1959         }

```

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`.

```

1960             \tikz@parse@node \pgfutil@firstofone
1961             ( nm - \int_use:N \g__nm_env_int - \_nm_i: - \_nm_j: .north-east )
1962         \dim_set:cn { l__nm_row _ \_nm_i: _ max_dim }
1963         { \dim_max:vn { l__nm_row _ \_nm_i: _ max_dim } \pgf@y }
1964         \seq_if_in:NxF \g__nm_multicolumn_cells_seq { \_nm_i: - \_nm_j: }
1965         {
1966             \dim_set:cn { l__nm_column _ \_nm_j: _ max_dim }
1967             { \dim_max:vn { l__nm_column _ \_nm_j: _ max_dim } \pgf@x }
1968         }
1969     }
1970 }
1971 }

```

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`).

```

1972     \tikzset { name~suffix = -medium }
1973     \_nm_create_nodes:

```

For “large nodes”, the exterior rows and columns 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__@@_col_int` (and not `\g__@@_col_total_int`). Idem for the rows.

```

1974     \int_set:Nn \g__nm_first_row_int 1
1975     \int_set:Nn \g__nm_first_col_int 1

```

We have to change the values of all the dimensions `l__@@_row_i_min_dim`, `l__@@_row_i_max_dim`, `l__@@_column_j_min_dim` and `l__@@_column_j_max_dim`.

```

1976     \int_step_variable:nNn { \g__nm_row_int - 1 } \_nm_i:
1977     {
1978         \dim_set:cn { l__nm_row _ \_nm_i: _ min _ dim }
1979         {
1980             (
1981                 \dim_use:c { l__nm_row _ \_nm_i: _ min _ dim } +
1982                 \dim_use:c { l__nm_row _ \int_eval:n { \_nm_i: + 1 } _ max _ dim }
1983             )
1984             / 2
1985         }
1986         \dim_set_eq:cc { l__nm_row _ \int_eval:n { \_nm_i: + 1 } _ max _ dim }

```

```

1987     { l__nm_row\___nm_i: _min_dim }
1988   }
1989   \int_step_variable:nNn { \g__nm_col_int - 1 } \__nm_j:
1990   {
1991     \dim_set:cn { l__nm_column _ \__nm_j: _ max _ dim }
1992     {
1993       (
1994         \dim_use:c
1995         { l__nm_column _ \__nm_j: _ max _ dim } +
1996         \dim_use:c
1997         { l__nm_column _ \int_eval:n { \__nm_j: + 1 } _ min _ dim }
1998       )
1999       / 2
2000     }
2001     \dim_set_eq:cc { l__nm_column _ \int_eval:n { \__nm_j: + 1 } _ min _ dim }
2002     { l__nm_column _ \__nm_j: _ max _ dim }
2003   }
2004   \dim_sub:cn
2005   { l__nm_column _ 1 _ min _ dim }
2006   \g__nm_left_margin_dim
2007   \dim_add:cn
2008   { l__nm_column _ \int_use:N \g__nm_col_int _ max _ dim }
2009   \g__nm_right_margin_dim

```

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

```

2010   \tikzset { name~suffix = -large }
2011   \__nm_create_nodes:
2012   \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.

```

2013   \cs_set:Npn \__nm_create_extra_nodes: { }

```

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

```

2014   \begin { tikzpicture } [ remember~picture , overlay ]
2015     \tikz@parse@node \pgfutil@firstofone
2016     ( nm - \int_use:N \g__nm_env_int - 1 - 1 - large .north-west )
2017     \dim_gset:Nn \g_tmpa_dim \pgf@x
2018     \tikz@parse@node \pgfutil@firstofone
2019     ( nm - \int_use:N \g__nm_env_int - 1 -
2020       \int_use:N \g__nm_col_int - large .north-east )
2021     \dim_gset:Nn \g_tmpb_dim \pgf@x
2022   \end { tikzpicture }
2023   \iow_now:Nn \@mainaux \ExplSyntaxOn
2024   \iow_now:Nx \@mainaux
2025   {
2026     \cs_gset:cpn { __nm_width_ \int_use:N \g__nm_env_int }
2027     { \dim_eval:n { \g_tmpb_dim - \g_tmpa_dim } }
2028   }
2029   \iow_now:Nn \@mainaux \ExplSyntaxOff
2030 }

```

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.

```

2031 \cs_new_protected:Nn \__nm_create_nodes:
2032 {
2033   \int_step_variable:nnNn \g__nm_first_row_int \g__nm_row_total_int \__nm_i:
2034   {
2035     \int_step_variable:nnNn \g__nm_first_col_int \g__nm_col_total_int \__nm_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.

```

2036     {
2037         \coordinate ( __nm~south~west )
2038             at ( \dim_use:c { l__nm_column_ \__nm_j: _min_dim } ,
2039                 \dim_use:c { l__nm_row_ \__nm_i: _min_dim } ) ;
2040         \coordinate ( __nm~north~east )
2041             at ( \dim_use:c { l__nm_column_ \__nm_j: _max_dim } ,
2042                 \dim_use:c { l__nm_row_ \__nm_i: _max_dim } ) ;

```

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

```

2043     \node
2044     [
2045         node~contents = { } ,
2046         fit = ( __nm~south~west ) ( __nm~north~east ) ,
2047         inner~sep = \c_zero_dim ,
2048         name = nm - \int_use:N \g__nm_env_int - \__nm_i: - \__nm_j: ,
2049         alias =
2050             \str_if_empty:NF \g__nm_name_str
2051                 { \g__nm_name_str - \__nm_i: - \__nm_j: }
2052     ]
2053     ;
2054 }
2055 }

```

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 correspondant values of n .

```

2056     \__nm_seq_mapthread_function:NNN
2057     \g__nm_multicolumn_cells_seq
2058     \g__nm_multicolumn_sizes_seq
2059     \__nm_node_for_multicolumn:n
2060 }

2061 \cs_new_protected:Npn \__nm_extract_coords: #1 - #2 \q_stop
2062 {
2063     \cs_set:Npn \__nm_i: { #1 }
2064     \cs_set:Npn \__nm_j: { #2 }
2065 }

```

The command `\@@_node_for_multicolumn:n` takes two arguments. The first is the position of the cell where the command `\multicolumn{n}{...}{...}` was issued in the format $i-j$ and the second is the value of n (the length of the "multi-cell").

```

2066 \cs_new_protected:Nn \__nm_node_for_multicolumn:n
2067 {
2068     \__nm_extract_coords: #1 \q_stop
2069     \coordinate ( __nm~south~west ) at
2070     (
2071         \dim_use:c { l__nm_column_ \__nm_j: _min_dim } ,
2072         \dim_use:c { l__nm_row_ \__nm_i: _min_dim }
2073     ) ;
2074     \coordinate ( __nm~north~east ) at
2075     (
2076         \dim_use:c { l__nm_column_ \int_eval:n { \__nm_j: + #2 - 1 } _max_dim } ,
2077         \dim_use:c { l__nm_row_ \__nm_i: _max_dim }
2078     ) ;
2079     \node
2080     [
2081         node~contents = { } ,
2082         fit = ( __nm~south~west ) ( __nm~north~east ) ,

```

```

2083     inner-sep = \c_zero_dim ,
2084     name = nm - \int_use:N \g__nm_env_int - \__nm_i: - \__nm_j: ,
2085     alias =
2086         \str_if_empty:NF \g__nm_name_str { \g__nm_name_str - \__nm_i: - \__nm_j: }
2087     ]
2088     ;
2089 }

```

15.17 Block matrices

The code in this section is for the construction of *block matrices*. It has no direct link with the environment `{NiceMatrixBlock}`.

The following command will be linked to `\Block` in the environments of `nicematrix`. We define it with `\NewExpandableDocumentCommand` of `xparse` because it has an optional argument between `<` and `>` (for TeX instructions put before the math mode of the label) and because it must be expandable since it reduces (in the case of a block of only one row) to a command `\multicolumn`.

```

2090 \NewExpandableDocumentCommand \__nm_Block: { m D < > { } m }
2091 {
2092     \__nm_Block_i #1 \q_stop { #2 } { #3 }
2093 }

```

The first argument of `\@@_Block:` (which is required) has a special syntax. It must be of the form i - j where i and j are the size (in rows and columns) of the block.

```

2094 \cs_new:Npn \__nm_Block_i #1-#2 \q_stop
2095 {
2096     \__nm_Block_ii:nnnn { #1 } { #2 }
2097 }

```

Now, the arguments have been extracted: `#1` is i (the number of rows of the block), `#2` is j (the number of columns of the block), `#3` are the tokens to put before the math mode and `#4` is the label of the block. The following command must *not* be protected because it contains a command `\multicolumn` (in the case of a block of only one row).

```

2098 \cs_new:Npn \__nm_Block_ii:nnnn #1 #2 #3 #4
2099 {

```

In the case of a block of only one row, we use a `\multicolumn` and not the general technique because, in this case, we want the label perfectly aligned with the base line of that row of the array.

```

2100     \int_compare:nNnTF { #1 } = 1
2101     {
2102         \multicolumn { #2 } { C } { \hbox:n { #3 $#4$ } }
2103         \__nm_gobble_ampersands:n { #2 - 1 }
2104     }
2105     { \__nm_Block_iii:nnnn { #1 } { #2 } { #3 } { #4 } }
2106 }

```

The command `\@@_Block_iii:nnnn` is for the case of a block of n rows with $n > 1$.

```

2107 \cs_new_protected:Npn \__nm_Block_iii:nnnn #1 #2 #3 #4
2108 {
2109     \bool_gset_true:N \g__nm_extra_nodes_bool

```

We write an instruction in the `code-after`. We write the instruction in the beginning of the `code-after` (the left in `\tl_gput_left:Nx`) because we want the Tikz nodes corresponding of the block created *before* potential instructions written by the user in the `code-after` (these instructions may use the Tikz node of the created block).

```

2110     \tl_gput_left:Nx \g__nm_code_after_tl
2111     {
2112         \__nm_Block_iv:nnnnn
2113         { \int_use:N \g__nm_row_int }
2114         { \int_use:N \g__nm_col_int }
2115         { \int_eval:n { \g__nm_row_int + #1 - 1 } }
2116         { \int_eval:n { \g__nm_col_int + #2 - 1 } }

```



```

2117     \exp_not:n { { #3 $ #4 $ } }
2118   }
2119 }

```

The command `\@@_gobble_ampersands:n` will gobble n ampersands (and also the spaces) where n is the argument of the command. This command is fully expandable and we need this feature.

```

2120 \group_begin:
2121   \char_set_catcode_letter:N \&
2122   \cs_new:Npn \_nm_gobble_ampersands:n #1
2123     {
2124       \int_compare:nNnT { #1 } > 0
2125         {
2126           \peek_charcode_remove_ignore_spaces:NT &
2127             { \_nm_gobble_ampersands:n { #1 - 1 } }
2128         }
2129     }
2130 \group_end:

```

The following command `\@@_Block_iii:nnnnn` will be used in the `code-after`. It's necessary to create two Tikz nodes because we want the label #5 really drawn in the *center* of the node.

```

2131 \cs_new_protected:Npn \_nm_Block_iv:nnnnn #1 #2 #3 #4 #5
2132   {
2133     \bool_if:nTF
2134       {
2135         \int_compare_p:nNn { #3 } > \g__nm_row_int
2136         || \int_compare_p:nNn { #4 } > \g__nm_col_int
2137       }
2138     { \msg_error:nnnn { nicematrix } { Block-too-large } { #1 } { #2 } }
2139     {
2140       \begin{tikzpicture}
2141         \node
2142           [
2143             fit = ( #1 - #2 - medium . north-west )
2144                 ( #3 - #4 - medium . south-east ) ,
2145             inner~sep = 0 pt ,
2146           ]

```

We don't forget the name of the node because the user may wish to use it.

```

2147         (#1-#2) { } ;
2148         \node at (#1-#2.center) { #5 } ;
2149       \end{tikzpicture}
2150     }
2151   }

```

15.18 We process the options

We process the options when the package is loaded (with `\usepackage`) but we recommend to use `\NiceMatrixOptions` instead.

We must process these options after the definition of the environment `{NiceMatrix}` because the option `renew-matrix` executes the code `\cs_set_eq:NN \env@matrix \NiceMatrix`.

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

```

2152 \ProcessKeysOptions { NiceMatrix }

```

15.19 Error messages of the package

```

2153 \_nm_msg_new:nn { Block-too-large }
2154   {
2155     You-try-to-draw-a-block-in-the-cell-#1-#2-of-your-matrix-but-the-matrix-is-
2156     too-small-for-that-block.\
2157     If-you-go-on,~this-command-line-will-be-ignored.
2158   }

```

```

2159 \_nm_msg_new:nn { Impossible~line }
2160 {
2161     A~dotted~line~can't~be~drawn~because~you~have~not~put~
2162     all~the~ampersands~required~on~the~row~#1.\
2163     If~you~go~on,~this~dotted~line~will~be~ignored.
2164 }
2165 \_nm_msg_new:nn { Wrong~last~row }
2166 {
2167     You~have~used~'last~row=\int_use:N \g__nm_last_row_int'~but~your~environment~
2168     \{\g__nm_type_env_str\}~seems~to~have~\int_use:N \g__nm_row_int\
2169     rows.~If~you~go~on,~the~value~of~\int_use:N \g__nm_row_int\
2170     will~be~used~for~last~row.~You~can~avoid~this~problem~by~using~'last~row'~
2171     without~value~(more~compilations~might~be~necessary).
2172 }
2173 \_nm_msg_new:nn { Draft~mode }
2174 { The~compilation~is~in~draft~mode:~the~dotted~lines~won't~be~drawn. }
2175 \_nm_msg_new:nn { Yet~in~env }
2176 {
2177     Environments~\{NiceArray\}~(or~\{NiceMatrix\},~etc.)~can't~be~
2178     nested.\
2179     This~error~is~fatal.
2180 }
2181 \_nm_msg_new:nn { Outside~math~mode }
2182 {
2183     The~environment~\{\g__nm_type_env_str\}~can~be~used~only~in~math~mode~
2184     (and~not~in~\token_to_str:N \vcenter).\
2185     This~error~is~fatal.
2186 }
2187 \_nm_msg_new:nn { Option~Transparent~suppressed }
2188 {
2189     The~option~'Transparent'~has~been~renamed~'transparent'.\
2190     However,~you~can~go~on~for~this~time.
2191 }
2192 \_nm_msg_new:nn { Option~RenewMatrix~suppressed }
2193 {
2194     The~option~'RenewMatrix'~has~been~renamed~'renew~matrix'.\
2195     However,~you~can~go~on~for~this~time.
2196 }
2197 \_nm_msg_new:nn { Bad~value~for~letter~for~dotted~lines }
2198 {
2199     The~value~of~key~'\tl_use:N\l_keys_key_tl'~must~be~of~length~1.\
2200     If~you~go~on,~it~will~be~ignored.
2201 }
2202 \_nm_msg_new:nnn { Unknown~key~for~NiceMatrixOptions }
2203 {
2204     The~key~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~command~
2205     \token_to_str:N \NiceMatrixOptions. \
2206     If~you~go~on,~it~will~be~ignored. \
2207     For~a~list~of~the~available~keys,~type~H~<return>.
2208 }
2209 {
2210     The~available~options~are~(in~alphabetic~order):~
2211     allow~duplicate~names,~
2212     code~for~first~col,~
2213     code~for~first~row,~
2214     code~for~last~col,~
2215     code~for~last~row,~
2216     exterior~arraycolsep,~
2217     hlines,~
2218     left~margin,~
2219     letter~for~dotted~lines,~

```

```

2220 nullify-dots,~
2221 parallelize-diags,~
2222 renew-dots,~
2223 renew-matrix,~
2224 right-margin,~
2225 small,~
2226 and-transparent
2227 }

2228 \_nm_msg_new:nnn { Unknown-option-for-NiceArray }
2229 {
2230 The-option-'\tl_use:N\l_keys_key_tl'~is-unknown-for-the-environment-
2231 \{NiceArray\}. \\
2232 If-you-go-on,~it-will-be-ignored. \\
2233 For-a-list-of-the-available-options,~type-H-<return>.
2234 }
2235 {
2236 The-available-options-are~(in-alphabetic-order):~
2237 b,~
2238 c,~
2239 code-after,~
2240 code-for-first-col,~
2241 code-for-first-row,~
2242 code-for-last-col,~
2243 code-for-last-row,~
2244 columns-width,~
2245 create-extra-nodes,~
2246 extra-left-margin,~
2247 extra-right-margin,~
2248 hlines,~
2249 left-margin,~
2250 name,~
2251 nullify-dots,~
2252 parallelize-diags,~
2253 renew-dots,~
2254 right-margin,~
2255 small,~
2256 and-t.
2257 }

2258 \_nm_msg_new:nnn { Unknown-option-for-NiceMatrix }
2259 {
2260 The-option-'\tl_use:N\l_keys_key_tl'~is-unknown-for-the-environment-
2261 \{NiceMatrix\}~and-its-variants. \\
2262 If-you-go-on,~it-will-be-ignored. \\
2263 For-a-list-of-the-available-options,~type-H-<return>.
2264 }
2265 {
2266 The-available-options-are~(in-alphabetic-order):~
2267 code-after,~
2268 columns-width,~
2269 create-extra-nodes,~
2270 extra-left-margin,~
2271 extra-right-margin,~
2272 hlines,~
2273 left-margin,~
2274 name,~
2275 nullify-dots,~
2276 parallelize-diags,~
2277 renew-dots,~
2278 right-margin~
2279 and-small.
2280 }

```

Despite its name, the following set of keys will be used for `{pNiceArray}` but also `{vNiceArray}`,

```

{VNiceArray}, etc. but not for {NiceArray}.
2281 \_nm_msg_new:nnn { Unknown-option-for-pNiceArray }
2282 {
2283   The-option-\tl_use:N\l_keys_key_tl'-is-unknown-for-the-environment-
2284   \{g_nm_type_env_str\}. \\
2285   If-you-go-on,-it-will-be-ignored. \\
2286   For-a-list-of-the-available-options,-type-H<return>.
2287 }
2288 {
2289   The-available-options-are-(in-alphabetic-order):~
2290   code-after,~
2291   code-for-first-col,~
2292   code-for-first-row,~
2293   code-for-last-col,~
2294   code-for-last-row,~
2295   columns-width,~
2296   create-extra-nodes,~
2297   extra-left-margin,~
2298   extra-right-margin,~
2299   first-col,~
2300   first-row,~
2301   last-col,~
2302   last-row,~
2303   hlines,~
2304   left-margin,~
2305   name,~
2306   nullify-dots,~
2307   parallelize-diags,~
2308   renew-dots,~
2309   right-margin~
2310   and~small.
2311 }
2312 \_nm_msg_new:nnn { Duplicate-name }
2313 {
2314   The-name-\l_keys_value_tl'-is-already-used-and-you-shouldn't-use~
2315   the-same-environment-name-twice.~You-can-go-on,-but,~
2316   maybe,-you-will-have-incorrect-results-especially~
2317   if-you-use-'columns-width=auto'.~If-you-use-nicematrix-inside-some~
2318   environments-of~amsmath,-this-error-may-be-an-artefact.~In-this-case,~
2319   use-the-option-'allow-duplicate-names'.\\
2320   For-a-list-of-the-names-already-used,-type-H<return>. \\
2321 }
2322 {
2323   The-names-already-defined-in-this-document-are:~
2324   \seq_use:Nnnn \g_nm_names_seq { ,~ } { ,~ } { ~and~ }.
2325 }
2326 \_nm_msg_new:nn { Option-auto-for-columns-width }
2327 {
2328   You-can't-give-the-value-'auto'~to-the-option-'columns-width'~here.~
2329   If-you-go-on,-the-option-will-be-ignored.
2330 }
2331 \_nm_msg_new:nn { Zero-row }
2332 {
2333   There-is-a-problem.~Maybe-your-environment-\{g_nm_type_env_str\}-is-empty.~
2334   Maybe-you-have-used-l,~c~and~r~instead-of~L,~C~and~R~in-the-preamble~
2335   of-your-environment. \\
2336   If-you-go-on,-the-result-may-be-incorrect.
2337 }
2338 \_nm_msg_new:nn { Use-of~:~in-first-position }
2339 {
2340   You-can't-use-the-column-specifier-\l_nm_letter_for_dotted_lines_str'-in-the~
2341   first-position-of-the-preamble-of-the-environment-\{g_nm_type_env_str\}. \\

```

```

2342   If~you~go~on,~this~dotted~line~will~be~ignored.
2343   }

```

15.20 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`.

```

2344 \cs_new:Npn \__nm_seq_mapthread_function:NNN #1 #2 #3
2345   {
2346   \group_begin:

```

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

```

2347   \int_step_inline:nm { \seq_count:N #1 }
2348   {
2349     \seq_pop:NN #1 \l_tmpa_tl
2350     \seq_pop:NN #2 \l_tmpb_tl
2351     \exp_args:NVV #3 \l_tmpa_tl \l_tmpb_tl
2352   }
2353   \group_end:
2354   }

```

```

2355 \cs_set_protected:Npn \__nm_renew_matrix:
2356   {
2357     \RenewDocumentEnvironment { pmatrix } { } {
2358       { \pNiceMatrix }
2359       { \endpNiceMatrix }
2360     \RenewDocumentEnvironment { vmatrix } { } {
2361       { \vNiceMatrix }
2362       { \endvNiceMatrix }
2363     \RenewDocumentEnvironment { Vmatrix } { } {
2364       { \VNiceMatrix }
2365       { \endVNiceMatrix }
2366     \RenewDocumentEnvironment { bmatrix } { } {
2367       { \bNiceMatrix }
2368       { \endbNiceMatrix }
2369     \RenewDocumentEnvironment { Bmatrix } { } {
2370       { \BNiceMatrix }
2371       { \endBNiceMatrix }
2372   }

```

15.21 Obsolete environments

```

2373 \NewDocumentEnvironment { pNiceArrayC } { } {
2374   {
2375     \bool_set_true:N \l__nm_last_col_bool
2376     \pNiceArray
2377   }
2378   { \endpNiceArray }
2379 \NewDocumentEnvironment { bNiceArrayC } { } {
2380   {
2381     \bool_set_true:N \l__nm_last_col_bool
2382     \bNiceArray
2383   }
2384   { \endbNiceArray }
2385 \NewDocumentEnvironment { BNiceArrayC } { } {
2386   {
2387     \bool_set_true:N \l__nm_last_col_bool
2388     \BNiceArray
2389   }
2390   { \endBNiceArray }

```

```

2391 \NewDocumentEnvironment { vNiceArrayC } { }
2392 {
2393   \bool_set_true:N \l__nm_last_col_bool
2394   \vNiceArray
2395 }
2396 { \endvNiceArray }

2397 \NewDocumentEnvironment { VNiceArrayC } { }
2398 {
2399   \bool_set_true:N \l__nm_last_col_bool
2400   \VNiceArray
2401 }
2402 { \endVNiceArray }

2403 \NewDocumentEnvironment { pNiceArrayRC } { }
2404 {
2405   \bool_set_true:N \l__nm_last_col_bool
2406   \int_set:Nn \l__nm_first_row_int \c_zero_int
2407   \pNiceArray
2408 }
2409 { \endpNiceArray }

2410 \NewDocumentEnvironment { bNiceArrayRC } { }
2411 {
2412   \bool_set_true:N \l__nm_last_col_bool
2413   \int_set:Nn \l__nm_first_row_int \c_zero_int
2414   \bNiceArray
2415 }
2416 { \endbNiceArray }

2417 \NewDocumentEnvironment { BNiceArrayRC } { }
2418 {
2419   \bool_set_true:N \l__nm_last_col_bool
2420   \int_set:Nn \l__nm_first_row_int \c_zero_int
2421   \BNiceArray
2422 }
2423 { \endBNiceArray }

2424 \NewDocumentEnvironment { vNiceArrayRC } { }
2425 {
2426   \bool_set_true:N \l__nm_last_col_bool
2427   \int_set:Nn \l__nm_first_row_int \c_zero_int
2428   \vNiceArray
2429 }
2430 { \endvNiceArray }

2431 \NewDocumentEnvironment { VNiceArrayRC } { }
2432 {
2433   \bool_set_true:N \l__nm_last_col_bool
2434   \int_set:Nn \l__nm_first_row_int \c_zero_int
2435   \VNiceArray
2436 }
2437 { \endVNiceArray }

2438 \NewDocumentEnvironment { NiceArrayCwithDelims } { }
2439 {
2440   \bool_set_true:N \l__nm_last_col_bool
2441   \NiceArrayWithDelims
2442 }
2443 { \endNiceArrayWithDelims }

2444 \NewDocumentEnvironment { NiceArrayRCwithDelims } { }
2445 {
2446   \bool_set_true:N \l__nm_last_col_bool
2447   \int_set:Nn \l__nm_first_row_int \c_zero_int
2448   \NiceArrayWithDelims
2449 }
2450 { \endNiceArrayWithDelims }

```

16 History

Changes between versions 1.0 and 1.1

The dotted lines are no longer drawn with Tikz nodes but with Tikz circles (for efficiency).
Modification of the code which is now twice faster.

Changes between versions 1.1 and 1.2

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

Changes between version 1.2 and 1.3

New environment `{pNiceArrayC}` and its variants.
Correction of a bug in the definition of `{BNiceMatrix}`, `{vNiceMatrix}` and `{VNiceMatrix}` (in fact, it was a typo).
Options are now available locally in `{pNiceMatrix}` and its variants.
The names of the options are changed. The old names were names in “camel style”.

Changes between version 1.3 and 1.4

The column types `w` and `W` can now be used in the environments `{NiceArray}`, `{pNiceArrayC}` and its variants with the same meaning as in the package `array`.
New option `columns-width` to fix the same width for all the columns of the array.

Changes between version 1.4 and 2.0

The versions 1.0 to 1.4 of `nicematrix` were focused on the continuous dotted lines whereas the version 2.0 of `nicematrix` provides different features to improve the typesetting of mathematical matrices.

Changes between version 2.0 and 2.1

New implementation of the environment `{pNiceArrayRC}`. With this new implementation, there is no restriction on the width of the columns.
The package `nicematrix` no longer loads `mathtools` but only `amsmath`.
Creation of “medium nodes” and “large nodes”.

Changes between version 2.1 and 2.1.1

Small corrections: for example, the option `code-for-first-row` is now available in the command `\NiceMatrixOptions`.
Following a discussion on TeX StackExchange³¹, Tikz externalization is now deactivated in the environments of the extension `nicematrix`.³²

Changes between version 2.1 and 2.1.2

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

³¹cf. tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package

³²Before this version, there was an error when using `nicematrix` with Tikz externalization. In any case, it's not possible to externalize the Tikz elements constructed by `nicematrix` because they use the options `overlay` and `remember picture`.

Changes between version 2.1.2 and 2.1.3

When searching the end of a dotted line from a command like `\Cdots` issued in the “main matrix” (not in the 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`).

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

Changes between version 2.1.3 and 2.1.4

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

See www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end

Changes between version 2.1.4 and 2.1.5

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

Option `allow-duplicate-names`.

Changes between version 2.1.5 and 2.2

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

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

Changes between version 2.2 and 2.2.1

Improvement of the vertical dotted lines drawn by the specifier “:” in the preamble.

Modification of the position of the dotted lines drawn by `\hdottedline`.

Changes between version 2.2.1 and 2.3

Compatibility with the column type `S` of `siunitx`.

Option `hlines`.

A warning is issued when the `draft` mode is used. In this case, the dotted lines are not drawn.

Changes between version 2.3 and 3.0

Modification of `\Hdotsfor`. Now `\Hdotsfor` erases the `\vlines` (of “|”) as `\hdotsfor` does.

Composition of exterior rows and columns of the four sides of the matrix (and not only on two sides) with the options `first-row`, `last-row`, `first-col` and `last-col`.

Changes between version 3.0 and 3.1

Command `\Block` to draw block matrices.

Error message when the user gives an incorrect value for `last-row`.

A dotted line can no longer cross another dotted line (except the dotted lines drawn by `\cdottedline`, `:` (in the preamble of the array) and `\line` in `code-after`).

The starred versions of `\Cdots`, `\Ldots`, etc. are now deprecated because, with the new implementation, they become pointless. These starred versions are no longer documented.

The vertical rules in the matrices (drawn by `|`) are now compatible with the color fixed by `colortbl`.

Correction of a bug: it was not possible to use the colon `:` in the preamble of an array when `pdflatex` was used with `french-babel` (because `french-babel` activates the colon in the beginning of the document).

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>	2121
<code>\%</code>	2156, 2162, 2178, 2184, 2189, 2194, 2199, 2205, 2206, 2231, 2232, 2261, 2262, 2284, 2285, 2319, 2320, 2335, 2341
<code>\{</code>	926, 989, 2168, 2177, 2183, 2231, 2261, 2284, 2333, 2341
<code>\}</code>	926, 991, 2168, 2177, 2183, 2231, 2261, 2284, 2333, 2341
<code>\ </code>	940
<code>\</code>	2168, 2169
A	
<code>\array</code>	433
<code>\arraycolsep</code>	158, 160, 162, 458, 635, 636, 724, 762, 764, 778, 836, 864, 958, 967, 1795, 1837, 1838
<code>\arrayrulewidth</code>	447, 448
<code>\arraystretch</code>	457
<code>\AtBeginDocument</code>	62, 90
B	
<code>\begin</code>	975, 982, 989, 996, 1003, 1056, 1290, 1609, 1751, 1816, 1829, 1928, 2014, 2140
<code>\bgroup</code>	349
<code>\Block</code>	542
<code>\BNiceArray</code>	2388, 2421
<code>\bNiceArray</code>	2382, 2414
<code>\BNiceMatrix</code>	2370
<code>\bNiceMatrix</code>	2367
bool commands:	
<code>\bool_do_until:Nn</code>	1161, 1221
<code>\bool_gset_eq:NN</code>	478, 486, 487, 489
<code>\bool_gset_false:N</code>	584
<code>\bool_gset_true:N</code>	572, 843, 1787, 2109
<code>\bool_if:NTF</code>	34, 98, 287, 409, 425, 441, 455, 460, 492, 543, 582, 598, 604, 607, 633, 664, 666, 670, 673, 675, 693, 727, 738, 775, 788, 849, 948, 954, 1008, 1053, 1093, 1111, 1120, 1127, 1185, 1213, 1245, 1274, 1282, 1312, 1319, 1334, 1348, 1350, 1362, 1366, 1381, 1395, 1397, 1409, 1413, 1418, 1426, 1431, 1437, 1441, 1456, 1460, 1465, 1469, 1498, 1502, 1507, 1511, 1555, 1557, 1568, 1590, 1591, 1592, 1636, 1642, 1648, 1654, 1660, 1685, 1763, 1768, 1775, 1808, 1910
<code>\bool_if:nTF</code>	1089, 1281, 1635, 1641, 1647, 1653, 1659, 1737, 2133
<code>\bool_new:N</code>	11, 27, 51, 52, 53, 60, 61, 85, 86, 87, 88, 89, 130, 131, 133, 134, 135, 138, 139, 1893
<code>\bool_set:Nn</code>	1422, 1435
<code>\bool_set_false:N</code>	962, 1029, 1043, 1125, 1126, 1160, 1165, 1220, 1225, 1310, 1360, 1407, 1454, 1496, 1705, 1706, 1749, 1750
<code>\bool_set_true:N</code>	12, 29, 32, 66, 93, 132, 183, 232, 599, 613, 902, 949, 1051, 1169, 1175, 1181, 1189, 1193, 1217, 1229, 1235, 1241, 1249, 1254, 1278, 1712, 1720, 1726, 1734, 1814, 1815, 1898, 1900, 2375, 2381, 2387, 2393, 2399, 2405, 2412, 2419, 2426, 2433, 2440, 2446
<code>\l_tmpa_bool</code>	1029, 1043, 1051, 1053, 1422, 1441
<code>\l_tmpb_bool</code>	1435, 1437
box commands:	
<code>\box_clear_new:N</code>	659
<code>\box_dp:N</code> ..	307, 319, 369, 503, 509, 513, 772
<code>\box_ht:N</code> ..	309, 314, 317, 505, 507, 511, 771
<code>\box_move_down:nn</code>	370, 742
<code>\box_move_up:nn</code>	384, 733
<code>\box_set_dp:Nn</code>	772
<code>\box_set_ht:Nn</code>	771
<code>\box_use:N</code>	350, 368, 384, 828, 896
<code>\box_use_drop:N</code>	750, 763, 773
<code>\box_wd:N</code>	326, 374, 648, 656, 801, 862
<code>\l_tmpa_box</code>	285, 307, 309, 314, 317, 319, 326, 350, 359, 368, 641, 648, 649, 656, 753, 771, 772, 773, 786, 801, 828, 847, 862, 896
<code>\l_tmpb_box</code>	367, 369, 374, 384
C	
<code>\Cdots</code>	534
<code>\cdots</code>	546, 1623
Cdots internal commands:	
<code>_nm_Cdots</code>	534, 546, 1639

cdots internal commands:	
<code>_nm_cdots</code>	1623, 1642
char commands:	
<code>\char_set_catcode_letter:N</code>	2121
<code>\cleaders</code>	1770
<code>\coordinate</code> ...	378, 383, 2037, 2040, 2069, 2074
cs commands:	
<code>\cs_generate_variant:Nn</code>	
.....	353, 1305, 1762, 1924, 1925
<code>\cs_gset:Npn</code>	1013, 1020, 1098, 1105, 1917, 2026
<code>\cs_gset:Npx</code>	1629
<code>\cs_gset_eq:NN</code>	111, 496
<code>\cs_if_exist:NTF</code>	
.....	600, 616, 623, 950, 1030, 1044,
	1086, 1321, 1336, 1368, 1383, 1788, 1826, 1947
<code>\cs_if_free:NTF</code>	463, 471,
	1034, 1205, 1266, 1308, 1358, 1405, 1452, 1494
<code>\cs_new:Npn</code>	
	436, 1669, 1680, 1783, 2094, 2098, 2122, 2344
<code>\cs_new_protected:Nn</code>	275, 295,
	321, 354, 1006, 1077, 1083, 1153, 1284,
	1306, 1315, 1356, 1403, 1450, 1492, 1534,
	1627, 1663, 1703, 1743, 1802, 1926, 2031, 2066
<code>\cs_new_protected:Npn</code>	18,
	19, 20, 21, 22, 23, 24, 25, 54, 114, 302,
	412, 423, 438, 453, 1844, 2061, 2107, 2131
<code>\cs_set:Npn</code>	430, 457, 490, 514,
	1155, 1195, 1256, 1741, 1766, 2013, 2063, 2064
<code>\cs_set_eq:NN</code>	102,
	427, 428, 429, 533, 534, 535, 536, 537, 538,
	539, 540, 541, 542, 545, 546, 547, 548,
	549, 550, 551, 560, 561, 562, 564, 1147,
	1622, 1623, 1624, 1625, 1626, 1668, 1764, 1892
<code>\cs_set_protected:Npn</code>	67, 96, 410, 606, 2355

D

<code>\Ddots</code>	536
<code>\ddots</code>	548, 1625
Ddots internal commands:	
<code>_nm_Ddots</code>	536, 548, 1651
ddots internal commands:	
<code>_nm_ddots</code>	1625, 1654
<code>\DeclareOption</code>	12, 13
dim commands:	
<code>\dim_abs:n</code>	1070
<code>\dim_add:Nn</code>	2007
<code>\dim_compare:nNnTF</code>	522, 1069, 1553
<code>\dim_compare_p:nNn</code>	1423, 1436
<code>\dim_eval:n</code>	2027
<code>\dim_gadd:Nn</code>	
	1352, 1353, 1594, 1601, 1616, 1617, 1837, 1838
<code>\dim_gset:Nn</code>	306,
	308, 313, 316, 318, 325, 503, 505, 507,
	509, 511, 513, 635, 636, 648, 656, 797,
	858, 1065, 1067, 1295, 1296, 1301, 1302,
	1439, 1480, 1522, 1754, 1755, 1757, 1758,
	1819, 1820, 1823, 1824, 1832, 1835, 2017, 2021
<code>\dim_gset_eq:NN</code>	298,
	479, 480, 481, 1349, 1351, 1396, 1398, 1444
<code>\dim_gzero:N</code>	299, 300
<code>\dim_gzero_new:N</code>	502, 504, 506, 508, 510,
	512, 605, 955, 1286, 1287, 1288, 1289, 1899
<code>\dim_max:nn</code>	307, 309, 314,
	317, 319, 326, 799, 860, 1441, 1925, 1963, 1967

<code>\dim_min:nn</code>	1441, 1924, 1954, 1958
<code>\dim_new:N</code>	49, 71,
	73, 140, 141, 142, 143, 144, 145, 146, 147, 148
<code>\dim_ratio:nn</code>	1484, 1526,
	1560, 1564, 1571, 1575, 1581, 1586, 1597, 1604
<code>\dim_set:Nn</code>	72,
	74, 184, 237, 353, 369, 458, 466, 473, 704,
	714, 1129, 1130, 1474, 1476, 1516, 1518,
	1537, 1578, 1583, 1762, 1769, 1789, 1790,
	1934, 1941, 1953, 1957, 1962, 1966, 1978, 1991
<code>\dim_set_eq:NN</code>	640, 651, 1932, 1939, 1986, 2001
<code>\dim_sub:Nn</code>	2004
<code>\dim_use:N</code>	
	.. 529, 530, 531, 1014, 1021, 1543, 1544,
	1547, 1548, 1918, 1981, 1982, 1994, 1996,
	2038, 2039, 2041, 2042, 2071, 2072, 2076, 2077
<code>\dim_zero:N</code>	464, 711, 721
<code>\dim_zero_new:N</code> ...	631, 632, 1115, 1116,
	1117, 1118, 1536, 1745, 1746, 1747, 1748,
	1810, 1811, 1812, 1813, 1931, 1933, 1938, 1940
<code>\c_max_dim</code>	1932, 1934, 1939, 1941
<code>\g_tmpa_dim</code>	1065, 1070, 2017, 2027
<code>\l_tmpa_dim</code>	369, 370, 384, 704, 711,
	734, 759, 771, 1578, 1616, 1789, 1790, 1795
<code>\g_tmpb_dim</code>	1067, 1070, 2021, 2027
<code>\l_tmpb_dim</code>	714, 721, 744, 766, 772, 1583, 1617
<code>\c_zero_dim</code>	331, 332, 402, 522, 640,
	651, 808, 809, 876, 877, 1553, 1780, 2047, 2083
<code>\dots</code>	550

E

<code>\egroup</code>	351
else commands:	
<code>\else:</code>	56
<code>\end</code>	977, 984, 991, 998, 1005, 1068,
	1303, 1619, 1759, 1825, 1836, 2012, 2022, 2149
<code>\endarray</code>	686, 966
<code>\endBNiceArray</code>	2390, 2423
<code>\endbNiceArray</code>	2384, 2416
<code>\endBNiceMatrix</code>	2371
<code>\endbNiceMatrix</code>	2368
<code>\endNiceArrayWithDelims</code>	
	... 907, 914, 921, 928, 935, 942, 2443, 2450
<code>\endpNiceArray</code>	2378, 2409
<code>\endpNiceMatrix</code>	2359
<code>\endVNiceArray</code>	2402, 2437
<code>\endvNiceArray</code>	2396, 2430
<code>\endVNiceMatrix</code>	2365
<code>\endvNiceMatrix</code>	2362
<code>\everycr</code>	500, 516
exp commands:	
<code>\exp_after:wN</code>	118
<code>\exp_args:Nv</code>	567, 683
<code>\exp_args:NvV</code>	2351
<code>\exp_not:n</code>	2117
<code>\ExplSyntaxOff</code>	1024, 1109, 1921, 2029
<code>\ExplSyntaxOn</code>	1010, 1095, 1912, 2023

F

fi commands:	
<code>\fi:</code>	58
fp commands:	
<code>\fp_to_dim:n</code>	1539

G

group commands:
`\group_begin:` 100, 639, 1085, 2120, 2346
`\group_end:` 105, 657, 1150, 2130, 2353
`\group_insert_after:N` 462, 602, 952

H

`\halign` 518, 520
hbox commands:
`\hbox:n` 41, 43, 45, 380, 760, 2102
`\hbox_overlap_left:n` 803
`\hbox_overlap_right:n` 865, 1791
`\hbox_set:Nn` 367, 641, 649, 753
`\hbox_set:Nw` 285, 359, 679, 786, 847
`\hbox_set_end:` 324, 365, 690, 795, 856
`\hbox_to_wd:nn` 374, 1771, 1793
`\hdotsfor` 540
`\hdotsfor` 551
`\hdottedline` 538
`\hfil` 376, 564
`\hfill` 1779
`\hrule` 447
`\Hspace` 539
`\hspace` 1666
`\hss` 564, 1776

I

`\ialign` 490, 514
`\iddots` 537
`\iddots` 36, 549, 1626
ldots internal commands:
`_nm_iddots` 537, 549, 1657
iddots internal commands:
`_nm_iddots` 1626, 1660
if commands:
`\if_mode_math:` 56
int commands:
`\int_add:Nn` 1163, 1164, 1247, 1248
`\int_compare:nNnTF` 278,
280, 288, 291, 304, 311, 443, 445, 573, 611,
661, 691, 695, 702, 712, 722, 729, 1047,
1079, 1091, 1166, 1168, 1172, 1174, 1178,
1180, 1226, 1228, 1232, 1234, 1238, 1240,
1472, 1514, 1672, 1709, 1723, 1804, 1846,
1848, 1850, 1852, 1854, 1859, 1861, 1867,
1869, 1875, 1877, 1879, 1884, 1886, 2100, 2124
`\int_compare:nTF` 243
`\int_compare_p:nNn` 2135, 2136
`\int_eval:n`
1675, 1982, 1986, 1997, 2001, 2076, 2115, 2116
`\int_gadd:Nn` 1678
`\int_gdecr:N` 1089
`\int_gincr:N` 277, 297, 603, 844, 953
`\int_gset:Nn` 283, 556, 583, 845
`\int_gset_eq:NN`
. 482, 484, 485, 575, 698, 1088, 1090
`\int_gsub:Nn` 1092
`\int_gzero:N` 440
`\int_gzero_new:N` 555, 557, 558, 559, 581
`\int_incr:N` 1471, 1513
`\int_max:nn` 284, 846
`\int_new:N` 48, 76, 78, 79, 81, 82, 84
`\int_set:Nn` 77, 80, 83, 618,
625, 1156, 1157, 1158, 1159, 1559, 1563,

1570, 1574, 1588, 1707, 1708, 1711, 1715,
1717, 1719, 1725, 1729, 1731, 1733, 1907,
1974, 1975, 2406, 2413, 2420, 2427, 2434, 2447
`\int_step_inline:nn` 2347
`\int_step_inline:nnn` 1740
`\int_step_inline:nnnn` 1610, 1913
`\int_step_variable:nNn` 1976, 1989
`\int_step_variable:nnNn`
. 1929, 1936, 1943, 1945, 2033, 2035
`\int_sub:Nn` 1187, 1188, 1223, 1224
`\int_use:N` 338, 339, 340, 345, 346,
392, 393, 394, 399, 400, 418, 419, 463, 467,
577, 616, 619, 817, 818, 824, 885, 886, 887,
892, 893, 1013, 1031, 1037, 1038, 1039,
1045, 1048, 1060, 1061, 1062, 1098, 1099,
1106, 1144, 1198, 1199, 1208, 1209, 1210,
1216, 1259, 1260, 1269, 1270, 1271, 1277,
1292, 1293, 1294, 1298, 1299, 1300, 1324,
1325, 1326, 1339, 1340, 1341, 1371, 1372,
1373, 1386, 1387, 1388, 1630, 1631, 1675,
1696, 1697, 1788, 1789, 1822, 1827, 1834,
1948, 1951, 1961, 2008, 2016, 2019, 2020,
2026, 2048, 2084, 2113, 2114, 2167, 2168, 2169
`\int_zero:N` 259, 261, 268, 270
`\int_zero_new:N`
. . . 1113, 1114, 1121, 1122, 1123, 1124, 1906
`\c_one_int` 243, 278, 280,
311, 1092, 1311, 1361, 1408, 1455, 1472, 1514
`\l_tmpa_int` 1559, 1563, 1570,
1574, 1598, 1605, 1610, 1715, 1716, 1729, 1730
`\l_tmpb_int` 1588, 1599, 1607
`\c_zero_int` 288, 304,
661, 702, 722, 729, 1079, 1311, 1361, 1408,
1804, 1846, 1848, 1850, 1852, 1859, 1867,
1875, 1884, 2406, 2413, 2420, 2427, 2434, 2447
iow commands:
`\iow_now:Nn`
. . . . 1010, 1011, 1018, 1024, 1095, 1096,
1103, 1109, 1912, 1915, 1921, 2023, 2024, 2029

K

`\kern` 45
keys commands:
`\keys_define:nn`
. . . 149, 168, 179, 197, 222, 252, 254, 266, 1894
`\l_keys_key_tl` 2199, 2204, 2230, 2260, 2283
`\keys_set:nn` 251, 608, 609, 956, 1905
`\l_keys_value_tl` 2314

L

`\Ldots` 533
`\ldots` 545, 1622
ldots internal commands:
`_nm_Ldots` 533, 545, 550, 1633
ldots internal commands:
`_nm_ldots` 1622, 1636
`\left` 644, 653, 756, 975, 982, 989, 996, 1003
`\line` 1147
`\lineskip` 707, 717
`\lVert` 1003
`\lvert` 996

M

`\makebox` 368

math commands:

- `\c_math_toggle_token` 286, 323, 643, 645, 652, 654, 682, 687, 755, 769, 787, 794, 848, 855, 1774, 1777
- `\mathinner` 38
- `\mkern` 40, 42, 44, 45

msg commands:

- `\msg_error:nn` 18, 19
- `\msg_error:nnn` 20, 1215, 1276
- `\msg_error:nnnn` 2138
- `\msg_fatal:nn` 21, 22
- `\msg_new:nnn` 23
- `\msg_new:nnnn` 24
- `\msg_redirect_name:nnn` 26
- `\msg_warning:nn` 35
- `\multicolumn` .. 541, 1668, 1682, 1688, 1700, 2102
- `\myfiledate` 8
- `\myfileversion` 9

N

- `\newcolumnstype` 356, 524, 525, 526, 529, 530, 531, 567
- `\NewDocumentCommand` 250, 1633, 1639, 1645, 1651, 1657, 1687, 1691
- `\NewDocumentEnvironment` 592, 900, 908, 915, 922, 929, 936, 943, 971, 978, 985, 992, 999, 1903, 2373, 2379, 2385, 2391, 2397, 2403, 2410, 2417, 2424, 2431, 2438, 2444
- `\NewExpandableDocumentCommand` 2090
- `\NiceArrayWithDelims` 905, 912, 919, 926, 933, 940, 2441, 2448
- `\NiceMatrixOptions` 250, 2205

nm internal commands:

- `__nm_actualization_for_first_and_last_row:` 302, 327, 796, 857
- `__nm_actually_draw_Ldots:` 1312, 1315, 1739
- `__nm_adapt_S_column:` 96, 111, 596
- `__nm_add_to_empty_cells:` 1627, 1637, 1643, 1649, 1655, 1661, 1665
- `__nm_after_array:` 602, 952, 1077
- `__nm_after_array_i:` 1080, 1083
- `__nm_array:` 423, 683, 963
- `\l_nm_auto_columns_width_bool` 135, 183, 460, 1900
- `__nm_begin_of_row:` 281, 295, 785
- `__nm_Block:` 542, 2090
- `\l_nm_block_auto_columns_width_bool` 604, 954, 1008, 1893, 1898, 1910
- `__nm_Block_i` 2092, 2094
- `__nm_Block_ii:nnnn` 2096, 2098
- `__nm_Block_iii:nnnn` 2105, 2107
- `__nm_Block_iv:nnnnn` 2112, 2131
- `\g_nm_Cdots_lines_tl` 585, 1136
- `__nm_Cell:` 121, 275, 360, 524, 525, 526
- `\g_nm_code_after_tl` 193, 576, 1148, 1149, 2110
- `\l_nm_code_for_first_col_tl` 170, 789
- `\l_nm_code_for_first_row_tl` 174, 289
- `\l_nm_code_for_last_col_tl` 172, 850
- `\l_nm_code_for_last_row_tl` 176, 292
- `\g_nm_col_int` 277, 278, 284, 340, 346, 394, 400, 419, 440, 558, 573, 575, 577, 844, 846, 887, 893, 1088, 1089,

- 1178, 1238, 1630, 1675, 1678, 1697, 1723, 1848, 1875, 1989, 2008, 2020, 2114, 2116, 2136
- `\g_nm_col_total_int` 283, 284, 559, 845, 846, 1088, 1936, 1946, 2035
- `\c_nm_colortbl_loaded_bool` ... 61, 66, 492
- `\l_nm_columns_width_dim` 49, 184, 237, 464, 466, 473, 522, 529, 530, 531
- `__nm_create_extra_nodes:` 1120, 1281, 1282, 1738, 1807, 1926, 2013
- `__nm_create_nodes:` 1973, 2011, 2031
- `\l_nm_ddots_int` 1113, 1471, 1472
- `\g_nm_Ddots_lines_tl` 588, 1134
- `\l_nm_delta_x_one_dim` ... 1115, 1474, 1484
- `\l_nm_delta_x_two_dim` ... 1117, 1516, 1526
- `\l_nm_delta_y_one_dim` ... 1116, 1476, 1484
- `\l_nm_delta_y_two_dim` ... 1118, 1518, 1526
- `__nm_dotfill:` 1764, 1766, 1798
- `\g_nm_dp_ante_last_row_dim` 298, 508, 509, 716, 718, 745
- `\g_nm_dp_last_row_dim` 298, 299, 318, 319, 512, 513, 718, 745
- `\g_nm_dp_row_zero_dim` 306, 307, 502, 503, 706
- `\c_nm_draft_bool` 11, 12, 34, 409, 1685, 1763, 1808
- `__nm_draw_Cdots:nn` 1356
- `__nm_draw_Ddots:nn` 1450
- `__nm_draw_Hdotsfor:nnn` 1695, 1703
- `__nm_draw_Iddots:nn` 1492
- `__nm_draw_Ldots:nn` 1306
- `__nm_draw_tikz_line:` 1354, 1399, 1446, 1488, 1530, 1534, 1760, 1840
- `__nm_draw_Vdots:nn` 1403
- `__nm_end_Cell:` . 123, 321, 364, 524, 525, 526
- `\g_nm_env_int` 48, 338, 392, 463, 467, 603, 616, 619, 817, 885, 953, 1013, 1037, 1050, 1060, 1098, 1144, 1208, 1269, 1292, 1298, 1324, 1339, 1371, 1386, 1631, 1788, 1789, 1827, 1907, 1913, 1948, 1951, 1961, 2016, 2019, 2026, 2048, 2084
- `__nm_error:n` 18, 226, 230, 236, 245, 249, 253, 264, 273, 697, 1081, 1805
- `__nm_error:nn` 19, 189
- `__nm_error:nnn` 20
- `__nm_everycr:` 436, 497, 500
- `__nm_everycr_i:` 437, 438
- `\l_nm_exterior_arraycolsep_bool` 130, 233, 666, 675, 962
- `\l_nm_extra_left_margin_dim` 146, 163, 681, 833, 960
- `\g_nm_extra_nodes_bool` 139, 478, 572, 1120, 1787, 2109
- `\l_nm_extra_nodes_bool` 138, 156, 478
- `\g_nm_extra_right_margin_dim` 148, 481, 689, 969
- `\l_nm_extra_right_margin_dim` 147, 164, 481, 871
- `__nm_extract_coords:` 2061, 2068
- `__nm_fatal:n` 21, 57, 598, 948
- `__nm_fatal:nn` 22
- `\l_nm_final_i_int` 1123, 1158, 1163, 1166, 1187, 1192, 1198, 1209, 1216, 1299, 1340, 1387, 1708, 1730

<code>\l_nm_final_j_int</code>	<code>_nm_j:</code>
1124, 1159, 1164, 1172, 1178, 1188, 1192,	1936, 1938,
1199, 1210, 1300, 1341, 1388, 1725, 1731, 1733	1939, 1940, 1941, 1946, 1948, 1952, 1955,
<code>\l_nm_final_open_bool</code>	1957, 1958, 1961, 1964, 1966, 1967, 1989,
..... 1126, 1165, 1169, 1175,	1991, 1995, 1997, 2001, 2002, 2035, 2038,
1181, 1185, 1213, 1282, 1334, 1350, 1381,	2041, 2048, 2051, 2064, 2071, 2076, 2084, 2086
1397, 1418, 1431, 1465, 1507, 1557, 1568,	<code>\l_nm_l_dim</code> 1536, 1537, 1553, 1560,
1591, 1592, 1706, 1726, 1734, 1737, 1750, 1815	1564, 1571, 1575, 1581, 1586, 1598, 1605, 1606
<code>_nm_find_extremities_of_line:nnnn</code> ..	<code>\l_nm_last_col_bool</code>
..... 1153, 1311, 1361, 1408, 1455, 1497 87,
<code>\g_nm_first_col_int</code>	260, 269, 670, 2375, 2381, 2387, 2393,
..... 81, 485, 722, 1936, 1946, 1975, 2035	2399, 2405, 2412, 2419, 2426, 2433, 2440, 2446
<code>\l_nm_first_col_int</code>	<code>\g_nm_last_col_found_bool</code>
..... 79, 80, 259, 268, 280, 485, 661, 1846 88, 584, 775, 843, 1089
<code>\l_nm_first_env_block_int</code> 1906, 1907, 1913	<code>\g_nm_last_row_int</code>
<code>\g_nm_first_row_int</code> 84, 445, 482, 691, 695, 698, 738, 1091, 2167
..... 78, 484, 729, 1929, 1943, 1974, 2033	<code>\l_nm_last_row_int</code>
<code>\l_nm_first_row_int</code> 82, 83, 262, 271, 291, 482, 611,
..... 76, 77, 261, 270, 484, 556,	618, 625, 712, 1854, 1861, 1869, 1879, 1886
702, 1850, 2406, 2413, 2420, 2427, 2434, 2447	<code>\g_nm_last_row_without_value_bool</code> ..
<code>_nm_gobble_ampersands:n</code> 2103, 2122, 2127 86, 487, 693, 1093
<code>_nm_Hdotsfor:</code>	<code>\l_nm_last_row_without_value_bool</code> ..
..... 540, 551, 1680 85, 488, 613
<code>_nm_Hdotsfor_i</code>	<code>\g_nm_last_vdotted_col_int</code>
..... 1683, 1687, 1691 573, 575, 581, 583
<code>\g_nm_Hdotsfor_lines_tl</code> .. 590, 1132, 1693	<code>\g_nm_Ldots_lines_tl</code>
<code>_nm_hdottedline:</code> 586, 1137
..... 538, 1783	<code>\g_nm_left_delim_dim</code> ... 631, 635, 648, 831
<code>\l_nm_hlines_bool</code>	<code>\g_nm_left_margin_dim</code> 142, 479, 2006
..... 133, 152, 441	<code>\l_nm_left_margin_dim</code>
<code>_nm_Hspace:</code> 140, 157, 479, 680, 832, 959, 1796
..... 539, 1663	<code>\l_nm_letter_for_dotted_lines_str</code> ..
<code>\g_nm_ht_last_row_dim</code> 244, 566, 567, 2340
..... 300, 316, 317, 510, 511, 716	<code>_nm_line:nn</code>
<code>\g_nm_ht_row_one_dim</code> 1147, 1743
..... 313, 314, 506, 507, 706, 708, 734	<code>\g_nm_max_cell_width_dim</code>
<code>\g_nm_ht_row_zero_dim</code> 325, 326, 605, 955, 1014, 1021, 1899, 1918
..... 308, 309, 504, 505, 708, 734	<code>_nm_msg_new:nn</code>
<code>_nm_i:</code> 23, 2153, 2159, 2165, 2173,
..... 1929, 1931,	2175, 2181, 2187, 2192, 2197, 2326, 2331, 2338
1932, 1933, 1934, 1943, 1948, 1952, 1953,	<code>_nm_msg_new:nnn</code>
1954, 1955, 1961, 1962, 1963, 1964, 1976, 24, 2202, 2228, 2258, 2281, 2312
1978, 1981, 1982, 1986, 1987, 2033, 2039,	<code>_nm_msg_redirect_name:nn</code>
2042, 2048, 2051, 2063, 2072, 2077, 2084, 2086 25, 239
<code>\l_nm_iddots_int</code>	<code>_nm_multicolumn:nnn</code>
..... 1114, 1513, 1514 541, 1669
<code>\g_nm_Iddots_lines_tl</code>	<code>\g_nm_multicolumn_cells_seq</code>
..... 589, 1135 553, 1674, 1955, 1964, 2057
<code>_nm_if_not_empty_cell:nn</code>	<code>\g_nm_multicolumn_sizes_seq</code> 554, 1676, 2058
..... 1027	<code>\g_nm_name_str</code> 137, 483, 614, 623,
<code>_nm_if_not_empty_cell:nnTF</code>	626, 1016, 1020, 1101, 1105, 2050, 2051, 2086
..... 1192, 1252, 1716, 1730	<code>\l_nm_name_str</code> 136, 191, 342, 344,
<code>\l_nm_impossible_line_bool</code>	396, 398, 469, 471, 474, 483, 821, 823, 889, 891
..... 60, 1217, 1278, 1310, 1312,	<code>\g_nm_names_seq</code>
1360, 1362, 1407, 1409, 1454, 1456, 1496, 1498 50, 188, 190, 2324
<code>\l_nm_in_env_bool</code> .. 51, 598, 599, 948, 949	<code>\g_nm_NiceArray_bool</code>
<code>\l_nm_initial_i_int</code> 53, 486, 727
..... 1121, 1156, 1223, 1226, 1247, 1253,	<code>\l_nm_NiceArray_bool</code>
1259, 1270, 1277, 1293, 1325, 1372, 1707, 1716 52, 486, 607, 633, 664, 673, 902
<code>\l_nm_initial_j_int</code>	<code>_nm_node_for_multicolumn:nn</code> .. 2059, 2066
..... 1122, 1157, 1224, 1232, 1238, 1248, 1253,	<code>\l_nm_nullify_dots_bool</code>
1260, 1271, 1294, 1326, 1373, 1711, 1717, 1719 134, 155, 1636, 1642, 1648, 1654, 1660
<code>\l_nm_initial_open_bool</code> .. 1125, 1225,	<code>_nm_old_multicolumn</code>
1229, 1235, 1241, 1245, 1274, 1281, 1319, 1668, 1671
1348, 1366, 1395, 1413, 1426, 1460, 1502,	<code>\l_nm_parallelize_diags_bool</code>
1555, 1590, 1705, 1712, 1720, 1737, 1749, 1814 131, 132, 153, 1111, 1469, 1511
<code>_nm_instruction_of_type:n</code>	<code>\l_nm_pos_env_str</code>
..... 410, 412, 1635, 1641, 1647, 1653, 1659 128, 129, 256, 257, 258, 434, 731, 740, 961
<code>\l_nm_inter_dots_dim</code>	<code>_nm_pre_array:</code>
..... 71, 72, 1130, 1560, 1564, 453, 630, 957
1571, 1575, 1581, 1586, 1598, 1605, 1769, 1772	<code>\c_nm_preamble_first_col_tl</code> 662, 781
	<code>\c_nm_preamble_last_col_tl</code> 671, 839
	<code>\l_nm_radius_dim</code>
 73, 74, 1129, 1614

`\l_nm_renew_dots_bool` 154, 232, 543
`_nm_renew_matrix:` 224, 227, 231, 2355
`_nm_renew_NC@rewrite@S:` 114, 582
`_nm_renewcolumnntype:nn` 354, 563, 564
`_nm_retrieve_coords:nn`
1284, 1305, 1317, 1364, 1411, 1424, 1458, 1500
`\c_nm_revtex_bool` 27, 29, 32, 425
`\g_nm_right_delim_dim` 632, 636, 656, 869
`\g_nm_right_margin_dim`
. 143, 480, 688, 968, 1796, 2009
`\l_nm_right_margin_dim` 141, 159, 480, 870
`\g_nm_row_int` 288, 291, 297, 304, 311, 339,
345, 393, 399, 418, 443, 445, 555, 556, 695,
698, 818, 824, 886, 892, 1079, 1090, 1092,
1166, 1630, 1675, 1696, 1822, 1834, 1852,
1854, 1859, 1861, 1867, 1869, 1877, 1879,
1884, 1886, 1976, 2113, 2115, 2135, 2168, 2169
`\g_nm_row_total_int`
. 557, 1090, 1099, 1106, 1929, 1943, 2033
`_nm_seq_mapthread_function:NNN` 2056, 2344
`\c_nm_siunitx_loaded_bool` 89, 93, 98, 582
`\g_nm_small_bool` 489, 1127
`\l_nm_small_bool`
. 151, 287, 455, 489, 788, 849, 1768, 1775
`\l_nm_stop_loop_bool`
1160, 1161, 1189, 1193, 1220, 1221, 1249, 1254
`\c_nm_table_collect_begin_tl` 106, 108, 121
`\c_nm_table_print_tl` 109, 110, 123
`_nm_test_if_math_mode:`
. 54, 597, 911, 918,
925, 932, 939, 947, 974, 981, 988, 995, 1002
`\l_nm_the_array_box` 659, 679, 750, 763
`\g_nm_type_env_str`
. 75, 594, 595, 903, 904, 910,
917, 924, 931, 938, 945, 946, 973, 980, 987,
994, 1001, 1151, 2168, 2183, 2284, 2333, 2341
`\g_nm_Vdots_lines_tl` 587, 1133
`_nm_vdottedline:n` 577, 1802
`_nm_vline:` 606, 1844
`_nm_vline_i:`
. 67, 1855, 1862, 1870, 1880, 1887, 1892
`\g_nm_width_first_col_dim` 145, 725, 797, 800
`\g_nm_width_last_col_dim` 144, 777, 858, 861
`_nm_write_max_cell_width:` 462, 1006
`\g_nm_x_final_dim` 1288, 1301, 1423, 1436,
1442, 1444, 1475, 1483, 1517, 1525, 1543,
1580, 1596, 1747, 1757, 1812, 1823, 1835, 1838
`\g_nm_x_initial_dim` 1286,
1295, 1423, 1436, 1439, 1442, 1444, 1475,
1483, 1517, 1525, 1544, 1580, 1594, 1596,
1613, 1616, 1745, 1754, 1810, 1819, 1832, 1837
`\g_nm_y_final_dim` 1289, 1302, 1349,
1351, 1353, 1396, 1398, 1477, 1480, 1519,
1522, 1547, 1585, 1603, 1748, 1758, 1813, 1824
`\g_nm_y_initial_dim`
. 1287, 1296, 1349, 1351, 1352, 1396,
1398, 1477, 1482, 1519, 1524, 1548, 1585,
1601, 1603, 1613, 1617, 1746, 1755, 1811, 1820
`\noalign` 437, 496, 1785
`\node` 335, 389, 812, 880, 2043, 2079, 2141, 2148
`\nulldelimiterspace` 640, 651

P

`\path` 1752

peek commands:
`\peek_charcode_remove_ignore_spaces:NTF`
. 2126
`\pgfpathcircle` 1612
`\pgfpoint` 1613
`\pgfpointanchor` 1064, 1066
`\pgfusepath` 1615
`\phantom` 1636, 1642, 1648, 1654, 1660
`\pNiceArray` 2376, 2407
`\pNiceMatrix` 2358
prg commands:
`\prg_do_nothing:` 102, 111, 496, 1764
`\prg_replicate:nn` 1688, 1700
`\prg_return_false:` 1041, 1054, 1071
`\prg_return_true:` 1032, 1072
`\prg_set_conditional:Npnn` 1027
`\ProcessKeysOptions` 2152
`\ProcessOptions` 14
`\ProvideDocumentCommand` 36
`\ProvidesExplPackage` 6

Q

quark commands:
`\q_stop` 2061, 2068, 2092, 2094

R

`\raise` 41, 43, 45
`\relax` 14, 561, 562
`\renewcommand` 116
`\RenewDocumentEnvironment`
. 2357, 2360, 2363, 2366, 2369
`\RequirePackage` 2, 4, 5, 15, 16, 17
`\right` 644, 653, 768, 977, 984, 991, 998, 1005
`\rVert` 1005
`\rvert` 998

S

`\scriptstyle` 287, 788, 849, 1775
seq commands:
`\seq_count:N` 2347
`\seq_gclear_new:N` 553, 554
`\seq_gput_left:Nn` 190, 1674, 1676
`\seq_if_in:NnTF` 188, 1955, 1964
`\seq_new:N` 50
`\seq_pop:NN` 2349, 2350
`\seq_use:Nnnn` 2324
skip commands:
`\skip_horizontal:n` 571, 680, 681, 688,
689, 724, 725, 762, 764, 777, 778, 829, 836,
864, 867, 958, 959, 960, 967, 968, 969, 1780
`\skip_vertical:n` 448, 759, 766
`\c_zero_skip` 501, 517
str commands:
`\c_colon_str` 248
`\str_gclear:N` 1151
`\str_gset:Nn` 595, 904, 910, 917,
924, 931, 938, 946, 973, 980, 987, 994, 1001
`\str_if_empty:NTF` 342, 396, 469, 594,
614, 821, 889, 903, 945, 1016, 1101, 2050, 2086
`\str_if_eq:nnTF` 182, 235, 731, 740
`\str_new:N` 75, 128, 136, 137
`\str_set:Nn` 129, 187, 244, 256, 257, 258, 961
`\str_set_eq:NN` 191
`\l_tmpa_str` 187, 188, 190, 191

T	
<code>\tabskip</code>	501, 517
T _E X and L ^A T _E X 2 _ε commands:	
<code>\@acol</code>	429
<code>\@acoll</code>	427
<code>\@acolr</code>	428
<code>\@addtopreamble</code>	606
<code>\@array@array</code>	431
<code>\@arrayacol</code>	427, 428, 429
<code>\@arrayrule</code>	606
<code>\@arstrutbox</code>	503, 505, 507, 509, 511, 513
<code>\@halignto</code>	430
<code>\@height</code>	447
<code>\@ifclassloaded</code>	28, 31
<code>\@ifnextchar</code>	560
<code>\@ifpackageloaded</code>	64, 92
<code>\@mainaux</code>	
.....	1010, 1011, 1018, 1024, 1095, 1096,
.....	1103, 1109, 1912, 1915, 1921, 2023, 2024, 2029
<code>\@temptokena</code>	101, 104, 118, 120
<code>\c@MaxMatrixCols</code>	963
<code>\CT@arc@</code>	67
<code>\CT@everycr</code>	494
<code>\CT@row@color</code>	496
<code>\NC@find</code>	102, 125
<code>\NC@find@W</code>	562
<code>\NC@find@w</code>	561
<code>\NC@rewrite@S</code>	103, 116
<code>\new@ifnextchar</code>	560
<code>\p@</code>	41, 43, 45
<code>\pgf@x</code>	1065, 1067, 1295, 1301, 1754, 1757,
.....	1819, 1823, 1832, 1835, 1958, 1967, 2017, 2021
<code>\pgf@y</code>	
.....	1296, 1302, 1755, 1758, 1820, 1824, 1954, 1963
<code>\pgfutil@firstofone</code>	
.....	1291, 1297, 1753, 1756,
.....	1817, 1821, 1830, 1833, 1950, 1960, 2015, 2018
<code>\tikz@library@external@loaded</code>	600, 950, 1086
<code>\tikz@parse@node</code>	1291, 1297, 1753, 1756,
.....	1817, 1821, 1830, 1833, 1950, 1960, 2015, 2018
tex commands:	
<code>\tex_the:D</code>	120
<code>\tikz</code>	328, 377, 382, 388, 805, 873
<code>\tikzset</code>	601, 951, 1087, 1138, 1972, 2010
tl commands:	
<code>\c_empty_tl</code>	194
<code>\tl_const:Nn</code>	781, 839
<code>\tl_count:n</code>	243
<code>\tl_gclear:N</code>	1149
<code>\tl_gclear_new:N</code>	585, 586, 587, 588, 589, 590
<code>\tl_gput_left:Nn</code>	2110
<code>\tl_gput_right:Nn</code>	414, 576, 1693
<code>\tl_gset:Nn</code>	104, 108, 110
<code>\tl_gset_eq:NN</code>	483
<code>\tl_item:Nn</code>	107, 108, 110
<code>\tl_new:N</code>	106, 109
<code>\tl_put_left:Nn</code>	662, 667
<code>\tl_put_right:Nn</code>	671, 676
<code>\tl_set:Nn</code>	107, 660, 1057
<code>\tl_set_rescan:Nnn</code>	565
<code>\tl_use:N</code>	2199, 2204, 2230, 2260, 2283
<code>\g_tmpa_tl</code>	104, 107, 110
<code>\l_tmpa_tl</code>	107, 108, 660, 662, 667,
.....	671, 676, 683, 1057, 1064, 1066, 2349, 2351
<code>\l_tmpb_tl</code>	2350, 2351
token commands:	
<code>\token_to_str:N</code>	2184, 2205
U	
use commands:	
<code>\use:N</code>	417, 467, 474, 619, 626, 1048
<code>\usetikzlibrary</code>	3
V	
<code>\vbox</code>	45
vbox commands:	
<code>\vbox:n</code>	372
<code>\vcenter</code>	644, 653, 757, 2184
<code>\Vdots</code>	535
<code>\vdots</code>	547, 1624
Vdots internal commands:	
<code>_nm_Vdots</code>	535, 547, 1645
vdots internal commands:	
<code>_nm_vdots</code>	1624, 1648
<code>\vline</code>	67, 1892
<code>\VNiceArray</code>	2400, 2435
<code>\vNiceArray</code>	2394, 2428
<code>\VNiceMatrix</code>	2364
<code>\vNiceMatrix</code>	2361

Contents

1	Presentation	1
2	The environments of this extension	2
3	The continuous dotted lines	2
3.1	The option <code>nullify-dots</code>	3
3.2	The command <code>\Hdotsfor</code>	4
3.3	How to generate the continuous dotted lines transparently	5
4	The Tikz nodes created by <code>nicematrix</code>	5

5	The code-after	6
6	The environment <code>{NiceArray}</code>	7
7	The dotted lines to separate rows or columns	9
8	The width of the columns	9
9	Block matrices	10
10	The option <code>small</code>	11
11	The option <code>hlines</code>	12
12	Utilisation of the column type <code>S</code> of <code>siunitx</code>	12
13	Technical remarks	12
	13.1 Intersections of dotted lines	12
	13.2 Diagonal lines	13
	13.3 The “empty” cells	13
	13.4 The option <code>exterior-arraycolsep</code>	14
	13.5 The class option <code>draft</code>	14
	13.6 A technical problem with the argument of <code>\\</code>	14
	13.7 Obsolete environments	14
14	Examples	15
	14.1 Dotted lines	15
	14.2 Width of the columns	17
	14.3 How to highlight cells of the matrix	18
	14.4 Direct utilisation of the Tikz nodes	20
15	Implementation	21
	15.1 Declaration of the package and extensions loaded	22
	15.2 Technical definitions	22
	15.2.1 Variables for the exterior rows and columns	24
	15.2.2 The column <code>S</code> of <code>siunitx</code>	25
	15.3 The options	27
	15.4 Code common to <code>{NiceArrayWithDelims}</code> and <code>{NiceMatrix}</code>	31
	15.5 The environment <code>{NiceArrayWithDelims}</code>	39
	15.6 The environment <code>{NiceMatrix}</code> and its variants	46
	15.7 Automatic width of the cells	47
	15.8 How to know whether a cell is “empty”	48
	15.9 After the construction of the array	49
	15.10The actual instructions for drawing the dotted line with Tikz	59
	15.11User commands available in the new environments	61
	15.12The command <code>\line</code> accessible in code-after	64
	15.13The commands to draw dotted lines to separate columns and rows	64
	15.14The vertical rules	66
	15.15The environment <code>{NiceMatrixBlock}</code>	67
	15.16The extra nodes	68
	15.17Block matrices	72
	15.18We process the options	73
	15.19Error messages of the package	73
	15.20Code for <code>\seq_mapthread_function:NNN</code>	77
	15.21Obsolete environments	77
16	History	79
	Index	81