

The package `nicematrix`*

F. Pantigny
fpantigny@wanadoo.fr

October 13, 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.

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

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`.

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

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

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 3.5 of `nicematrix`, at the date of 2019/10/13.

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>
<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}`.

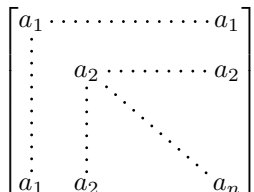
The environment `{NiceArray}` is similar to the environment `{array}` of the package `{array}`. However, for technical reasons, in the preamble of the environment `{NiceArray}`, the user must use the letters `L`, `C` and `R` instead of `l`, `c` and `r`. It's possible to use the constructions `w{...}{...}`,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

3 The continuous dotted lines

Inside the environments of the extension `nicematrix`, new commands are defined: `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, and `\Iddots`. These commands are intended to be used in place of `\dots`, `\cdots`, `\vdots`, `\ddots` and `\iddots`.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 & \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 & 0</code>
<code>\Vdots</code>	<code>& & \Vdots</code>
<code>0</code>	<code>& \Cdots & 0</code>
<code>\end{bNiceMatrix}</code>	



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

<code>\begin{bNiceMatrix}</code>	
<code>0</code>	<code>& \Cdots & \Cdots & 0</code>
<code>\Vdots</code>	<code>& & & \Vdots</code>
<code>\Vdots</code>	<code>& & & \Vdots</code>
<code>0</code>	<code>& \Cdots & \Cdots & 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: $\cdot\cdot\cdot$. 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).

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 & \\
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}
```

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

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.

⁶In `nicematrix`, one should use `\hspace*` and not `\hspace` for such an usage because `nicematrix` loads `array`. One may also remark that it's possible to fix the width of a column by using the environment `{NiceArray}` (or one of its variants) with a column of type `w` or `W`: see p. 10

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

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

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

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

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

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

3.2 The command `\Hdotsfor`

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

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

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

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

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

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

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

3.3 How to generate the continuous dotted lines transparently

The package `nicematrix` provides an option called `transparent` for using existing code transparently in the environments of the `amsmath` : `{matrix}`, `{pmatrix}`, `{bmatrix}`, etc. In fact, this option is an alias for the conjunction of two options: `renew-dots` and `renew-matrix`.⁷

- 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”.

⁷The options `renew-dots`, `renew-matrix` and `transparent` can be fixed with the command `\NiceMatrixOptionsn` like the other options. However, they can also be fixed as options of the command `\usepackage` (it's an exception for these three specific options.)

The 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.⁸

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

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`.⁹

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

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}
```

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

⁸There is no “large nodes” created in the exterior rows and columns (for these rows and columns, cf. p. 7).

⁹The 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 R¹⁰ instead of l, c and r. It's possible to use the constructions `w{...}{...}`, `W{...}{...}`, `l, >{...}`, `<{...}`, `@{...}`, `!{...}` and `*{n}{...}` but the letters p, m and b should not be used.¹¹

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).

```


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


```

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

```


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


```

7 The exterior row and columns

The options `first-row`, `last-row`, `first-col` and `last-col` allow the composition of exterior rows and columns in the environments of `nicematrix`.

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

¹⁰The column types L, C and R are defined locally inside `{NiceArray}` with `\newcolumnntype` of `array`. This definition overrides an eventual previous definition. In fact, the column types `w` and `W` are also redefined.

¹¹In a command `\multicolumn`, one should also use the letters L, C, R.

```

 $\begin{pNiceMatrix}[first-row,last-row=5,first-col,last-col=5]$ 
& C_1 & & C_2 & & C_3 & & C_4 & & \\
L_1 & a_{11} & & a_{12} & & a_{13} & & a_{14} & & L_1 \\
L_2 & a_{21} & & a_{22} & & a_{23} & & a_{24} & & L_2 \\
L_3 & a_{31} & & a_{32} & & a_{33} & & a_{34} & & L_3 \\
L_4 & a_{41} & & a_{42} & & a_{43} & & a_{44} & & L_4 \\
& C_1 & & C_2 & & C_3 & & C_4 & & \\
\end{pNiceMatrix}

```

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

We have several remarks to do.

- For the environments with an explicit preamble (i.e. `{NiceArray}` and its variants), no letter must be given in that preamble for the potential first column and the potential last column: the first column will be automatically (and necessarily) of type `R` and the last column will be automatically of type `L`.
- In an environment with an explicit preamble, the option `last-col` must be used *without* value: the number of columns will be automatically computed from the preamble of the array.
- For the potential last row, the option `last-row` may, in fact, be used without value. In this case, `nicematrix` computes, during the first compilation, the number of row of the array and writes that information in the `.aux` file for the second run. In the following example, the option `last-row` will be used without value.

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

```

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

```

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

Remarks

- As shown in the previous example, an horizontal rule (drawn by `\hline`) doesn't extend in the exterior columns and a vertical rule (specified by a `|` in the preamble of the array) doesn't extend in the exterior rows.¹²
- Logically, the potential option `columns-width` (described p. 10) doesn't apply to the “first column” and “last column”.
- For technical reasons, it's not possible to use the option of the command `\` after the “first row” or before the “last row” (the placement of the delimiters would be wrong).

8 The dotted lines to separate rows or columns

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

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

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

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

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

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

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

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

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

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

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

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

¹²The latter is not true when the extension `arydshln` is loaded besides `nicematrix`. In fact, `nicematrix` and `arydshln` are not totally compatible because `arydshln` redefines many internals of `array`. On another 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.

9 The width of the columns

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

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

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

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

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

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

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

It's possible to give the special value `auto` to the option `columns-width`: all the columns of the array will have a width equal to the widest cell of the array.¹³

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

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

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

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

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

But it's also possible to fix a zone where all the matrices will have their columns of the same width, equal to the widest cell of all the matrices. This construction uses the environment `{NiceMatrixBlock}` with the option `auto-columns-width`.¹⁴

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

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

Several compilations may be necessary to achieve the job.

¹³The result is achieved with only one compilation (but Tikz will have written informations in the `.aux` file and a message requiring a second compilation will appear).

¹⁴At this time, this is the only usage of the environment `{NiceMatrixBlock}` but it may have other usages in the future.

10 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).

```
\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}
```

$$\left[\begin{array}{ccc|c} A & & & 0 \\ & \hspace*{1cm} & & \vdots \\ & & & 0 \\ \hline 0 & \cdots & 0 & 0 \end{array} \right]$$

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

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

$$\left[\begin{array}{ccc|c} A & & & 0 \\ & \hspace*{1cm} & & \vdots \\ & & & 0 \\ \hline 0 & \cdots & 0 & 0 \end{array} \right]$$

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

11 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`).

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

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

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.

12 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>

13 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 `dcolum` are not supported by `nicematrix`.

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

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

14 Technical remarks

14.1 Intersections of dotted lines

Since the version 3.1 of `nicematrix`, the dotted lines created by `\Cdots`, `\Ldots`, `\Vdots`, etc. can’t intersect.¹⁵

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.

¹⁵Of the contrary, dotted lines created by `\hdottedline`, the letter “:” in the preamble of the array and the command `\line` in the `code-after` can have intersections with other dotted lines.

```

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

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

14.2 Diagonal lines

By default, all the diagonal lines¹⁶ of a same array are “parallelized”. That means that the first diagonal line is drawn and, then, the other lines are drawn parallel to the first one (by rotation around the left-most extremity of the line). That’s why the position of the instructions `\Ddots` in the array can have a marked effect on the final result.

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

Example with parallelization (default):

```

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

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

```

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

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

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

The same example without parallelization:

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

14.3 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}
```

¹⁶We speak of the lines created by `\Ddots` and not the lines created by a command `\line` in `code-after`.

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

- Each cell whose TeX output has a width 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`.

14.4 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 idea¹⁷. The environment `{matrix}` of `amsmath` and its variants (`{pmatrix}`, `{vmatrix}`, etc.) of `amsmath` prefer to delete these spaces with explicit instructions `\hskip -\arraycolsep`. The extension `nicematrix` does the same in all its environments, `{NiceArray}` included. However, if the user wants the environment `{NiceArray}` behaving by default like the environment `{array}` (for example, when adapting an existing document) it's possible to control this behaviour with the option `exterior-arraycolsep`, set by the command `\NiceMatrixOptions`. With this option, exterior spaces of length `\arraycolsep` will be inserted in the environments `{NiceArray}` (the other environments of `nicematrix` are not affected).

14.5 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 `:`).¹⁸ That's why, when the class option `draft` is used, the dotted lines are not drawn, for a faster compilation.

14.6 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.

<pre>\begin{pNiceMatrix} a & \frac{A}{B} \\ b & c \end{pNiceMatrix}</pre>	$\begin{pmatrix} a & \frac{A}{B} \\ b & c \end{pmatrix}$
---	--

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

<pre>\begin{pNiceMatrix} a & \frac{A}{B} \\ \noalign{\kern2mm} b & c \end{pNiceMatrix}</pre>	$\begin{pmatrix} a & \frac{A}{B} \\ b & c \end{pmatrix}$
--	--

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

¹⁷In the documentation of `{amsmath}`, we can read: *The extra space of `\arraycolsep` that `array` adds on each side is a waste so we remove it [in `{matrix}`] (perhaps we should instead remove it from `array` in general, but that's a harder task).* It's possible to suppress these spaces for a given environment `{array}` with a construction like `\begin{array}{@{}cccc@{}}... \end{array}`.

¹⁸The main reason is that we want dotted lines with round dots (and not square dots) with the same space on both extremities of the lines. To achieve this goal, we have to construct our own system of dotted lines.

```

\begin{pNiceMatrix}
a & \multicolumn{1}{\frac{AB}}{\hspace{2mm}}}
b & c
\end{pNiceMatrix}

```

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

14.7 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}` ;
- `{NiceArrayRCwithDelims}` ;
- `{pNiceArrayRC}`, `{bNiceArrayRC}`, `{BNiceArrayRC}`, `{vNiceArrayRC}`, `{VNiceArrayRC}`.

They might be deleted in a future version of `nicematrix`.

15 Examples

15.1 Dotted lines

A tridiagonal matrix:

```

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

```

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

A permutation matrix:

```

$\begin{pNiceMatrix}
0      & 1 & 0 & & & \Cdots & 0      & \\
\Vdots & & & & & \Ddots & & \Vdots \\
      & & & & & \Ddots & & \\
      & & & & & \Ddots & & 0 \\
0      & 0 & & & & & 1      & \\
1      & 0 & & & \Cdots & & 0      & \\
\end{pNiceMatrix}$

```

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

An example with `\Iddots`:

```

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

```

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

An example with `\multicolumn`:

```

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

```

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

An example with `\Hdotsfor`:

```

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

```

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

An example for the resultant of two polynomials:

```

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

```


$$\left| \begin{array}{cccc} a_0 & & & \\ & \ddots & & \\ a_1 & & & \\ & \ddots & & \\ & & a_0 & \\ & & & \ddots \\ a_p & & a_1 & \\ & & & \ddots \\ & & & a_p \end{array} \right| \begin{array}{c} \vdots \\ b_0 \\ \vdots \\ b_1 \\ \vdots \\ b_q \\ \vdots \\ b_q \end{array}$$

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}
```

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

15.2 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}
```

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

15.3 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`.

```

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

```

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

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}}

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

```

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

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

```

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

```

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

```

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

```

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

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}
```

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

15.4 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}
&
```

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

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

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

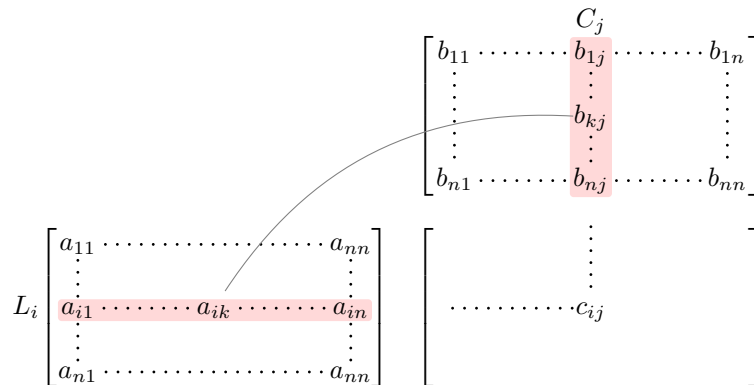
```
\begin{bNiceArray}{CC>{\strut}CCC}[name=A,first-col]
& a_{11} & \Cdots & & a_{nn} \\\
& \Vdots & & & \Vdots \\\
L_i & a_{i1} & \Cdots & a_{ik} & \Cdots & a_{in} \\\
& \Vdots & & & \Vdots \\\
& a_{n1} & \Cdots & & a_{nn} \\\
\end{bNiceArray}
&
```

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

```
\begin{bNiceArray}{CC>{\strut}CCC}
& & & \\\
& & \Vdots & \\\
\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}
```



16 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.

16.1 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.¹⁹

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

We give the traditionnal 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:nnn { nicematrix } }
20 \cs_new_protected:Npn \__nm_error:nnn { \msg_error:nnnn { 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 } }
```

¹⁹cf. tex.stackexchange.com/questions/57424/using-of-usetikzlibrary-in-an-expl3-package-fails

16.2 Technical definitions

We test whether the current class is `revtex4-1` or `revtex4-2` because these classes redefine `\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:n { nicematrix } { Draft-mode } }

```

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

```

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

```

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

```

49 \int_new:N \g__nm_NiceMatrixBlock_int

```

The dimension `\l_@@_columns_width_dim` will be used when the options specify that all the columns must have the same width (but, if the key `columns-width` is used with the special value `auto`, the boolean `\l_@@_auto_columns_width_bool` will be raised).

```

50 \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`.

```

51 \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.

```

52 \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`).

```

53 \bool_new:N \l__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.²⁰

```

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__nm_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__nm_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
76 \tl_new:N \g__nm_code_after_tl

```

²⁰Of course, the user should solve the problem by adding the lacking ampersands.

16.2.1 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`.

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

- **First column**

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

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

- **Last row**

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

```
81 \int_new:N \l__nm_last_row_int
82 \int_set:Nn \l__nm_last_row_int { -2 }
```

If, in an environment like `{pNiceArray}`, the option `last-row` is used without value, we will globally raise the following flag. It will be used to know if we have, after the construction of the array, to write in the `aux` file the number of the “last row”.²¹

```
83 \bool_new:N \l__nm_last_row_without_value_bool
```

- **Last column**

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

```
84 \int_new:N \l__nm_last_col_int
85 \int_set:Nn \l__nm_last_col_int { -1 }
```

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

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

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

```
86 \bool_new:N \g__nm_last_col_found_bool
```

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

²¹We 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.

16.2.2 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`.

```

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

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

```

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

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

```

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

However, we don't want to use explicitly any private command of `siunitx`. That's why we will extract the name of the two `__siunitx...` commands by their position in the code of `\NC@rewrite@S`.

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

```

94 \cs_set_protected:Npn \__nm_adapt_S_column:
95 {
```

In the preamble of the LaTeX document, the boolean `\c__nm_siunitx_loaded_bool` won't be known. That's why we test the existence of `\c__nm_siunitx_loaded_bool` and not its value.²²

```

96   \bool_if:NT \c__nm_siunitx_loaded_bool
97   {
98     \group_begin:
99     \@temptokena = { }
```

²²Indeed, `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`).

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

```
100 \cs_set_eq:NN \NC@find \prg_do_nothing:
101 \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`).

```
102 \tl_gset:NV \g_tmpa_tl \@temptokena
103 \group_end:
104 \tl_new:N \c__nm_table_collect_begin_tl
105 \tl_set:Nx \l_tmpa_tl { \tl_item:Nn \g_tmpa_tl 2 }
106 \tl_gset:Nx \c__nm_table_collect_begin_tl { \tl_item:Nn \l_tmpa_tl 1 }
107 \tl_new:N \c__nm_table_print_tl
108 \tl_gset:Nx \c__nm_table_print_tl { \tl_item:Nn \g_tmpa_tl { -1 } }
```

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

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

```
109 \cs_gset_eq:NN \__nm_adapt_S_column: \prg_do_nothing:
110 }
111 }
```

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).

```
112 \cs_new_protected:Npn \__nm_renew_NC@rewrite@S:
113 {
114   \renewcommand*{\NC@rewrite@S}[1][ ]
115   {
116     \@temptokena \exp_after:wN
117     {
118       \tex_the:D \@temptokena
119       > { \__nm_Cell: \c__nm_table_collect_begin_tl S {##1} }
120       c
121       < { \c__nm_table_print_tl \__nm_end_Cell: }
122     }
123     \NC@find
124   }
125 }
```

16.3 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.

```
126 \str_new:N \l__nm_pos_env_str
127 \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`).

```
128 \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`.

```
129 \bool_new:N \l__nm_parallelize_diags_bool
130 \bool_set_true:N \l__nm_parallelize_diags_bool
```

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

```
131 \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.).

```
132 \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).

```
133 \bool_new:N \l__nm_auto_columns_width_bool
```

The token list `\l_@@_name_str` will contain the optional name of the environment: this name can be used to access to the Tikz nodes created in the array from outside the environment.

```
134 \str_new:N \l__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.

```
135 \bool_new:N \l__nm_extra_nodes_bool
```

```
136 \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`.

```
137 \dim_new:N \l__nm_left_margin_dim
```

```
138 \dim_new:N \l__nm_right_margin_dim
```

```
139 \dim_new:N \g__nm_width_last_col_dim
```

```
140 \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`.

```
141 \dim_new:N \l__nm_extra_left_margin_dim
```

```
142 \dim_new:N \l__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.

```
143 \keys_define:nn { NiceMatrix / Global }
144 {
145   code-for-first-col .tl_set:N = \l__nm_code_for_first_col_tl ,
146   code-for-first-col .value_required:n = true ,
147   code-for-last-col .tl_set:N = \l__nm_code_for_last_col_tl ,
148   code-for-last-col .value_required:n = true ,
149   code-for-first-row .tl_set:N = \l__nm_code_for_first_row_tl ,
150   code-for-first-row .value_required:n = true ,
151   code-for-last-row .tl_set:N = \l__nm_code_for_last_row_tl ,
152   code-for-last-row .value_required:n = true ,
153   small .bool_set:N = \l__nm_small_bool ,
154   hlines .bool_set:N = \l__nm_hlines_bool ,
155   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`.

```
156   renew-dots .bool_set:N = \l__nm_renew_dots_bool ,
```

```
157   renew-dots .value_forbidden:n = true ,
```

```
158   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.²³

```

159   create-extra-nodes .bool_set:N = \l__nm_extra_nodes_bool ,
160   left-margin .dim_set:N = \l__nm_left_margin_dim ,
161   left-margin .default:n = \arraycolsep ,
162   right-margin .dim_set:N = \l__nm_right_margin_dim ,
163   right-margin .default:n = \arraycolsep ,
164   margin .meta:n = { left-margin = #1 , right-margin = #1 } ,
165   margin .default:n = \arraycolsep ,
166   extra-left-margin .dim_set:N = \l__nm_extra_left_margin_dim ,
167   extra-right-margin .dim_set:N = \l__nm_extra_right_margin_dim ,
168   extra-margin .meta:n =
169     { extra-left-margin = #1 , extra-right-margin = #1 } ,
170 }

```

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

```

171 \keys_define:nn { NiceMatrix / Env }
172 {
173   columns-width .code:n =
174     \str_if_eq:nnTF { #1 } { auto }
175     { \bool_set_true:N \l__nm_auto_columns_width_bool }
176     { \dim_set:Nn \l__nm_columns_width_dim { #1 } } ,
177   columns-width .value_required:n = true ,
178   name .code:n =
179     \str_set:Nn \l_tmpa_str { #1 }
180     \seq_if_in:NVTF \g__nm_names_seq \l_tmpa_str
181     { \__nm_error:nn { Duplicate-name } { #1 } }
182     { \seq_gput_left:NV \g__nm_names_seq \l_tmpa_str }
183     \str_set_eq:NN \l__nm_name_str \l_tmpa_str ,
184   name .value_required:n = true ,
185   code-after .tl_gset:N = \g__nm_code_after_tl ,
186   code-after .value_required:n = true ,
187   first-col .code:n = \int_zero:N \l__nm_first_col_int ,
188   first-row .code:n = \int_zero:N \l__nm_first_row_int ,
189   last-row .int_set:N = \l__nm_last_row_int ,
190   last-row .default:n = -1 ,
191 }

```

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

```

192 \keys_define:nn { NiceMatrix }
193 {
194   NiceMatrixOptions .inherit:n =
195     {
196       NiceMatrix / Global ,
197     } ,
198   NiceMatrix .inherit:n =
199     {
200       NiceMatrix / Global ,
201       NiceMatrix / Env
202     } ,
203   NiceArray .inherit:n =
204     {
205       NiceMatrix / Global ,
206       NiceMatrix / Env ,
207     } ,
208   pNiceArray .inherit:n =
209     {

```

²³In fact, we should not because, if there is a `w` node, the `w` node is used instead of the “medium” node.

```

210     NiceMatrix / Global ,
211     NiceMatrix / Env ,
212 }
213 }

```

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

```

214 \keys_define:nn { NiceMatrix / NiceMatrixOptions }
215 {

```

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

```

216     renew-matrix .code:n = \__nm_renew_matrix: ,
217     renew-matrix .value_forbidden:n = true ,
218     RenewMatrix .code:n = \__nm_error:n { Option~RenewMatrix~suppressed }
219         \__nm_renew_matrix: ,
220     transparent .meta:n = { renew-dots , renew-matrix } ,
221     transparent .value_forbidden:n = true,
222     Transparent .code:n = \__nm_error:n { Option~Transparent~suppressed }
223         \__nm_renew_matrix:
224         \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}`.

```

225     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.

```

226     columns-width .code:n =
227         \str_if_eq:nnTF { #1 } { auto }
228         { \__nm_error:n { Option~auto~for~columns~width } }
229         { \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 distincts environments of `nicematrix` (theses names are global and not local to the current TeX scope). However, the option `allow-duplicate-names` disables this feature.

```

230     allow-duplicate-names .code:n =
231         \__nm_msg_redirect_name:nn { Duplicate~name } { none } ,
232     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`).

```

233     letter-for-dotted-lines .code:n =
234     {
235         \int_compare:nTF { \tl_count:n { #1 } = \c_one_int }
236         { \str_set:Nx \l__nm_letter_for_dotted_lines_str { #1 } }
237         { \__nm_error:n { Bad~value~for~letter~for~dotted~lines } }
238     } ,
239     letter-for-dotted-lines .value_required:n = true ,

240     unknown .code:n = \__nm_error:n { Unknown~key~for~NiceMatrixOptions }
241 }

242 \str_new:N \l__nm_letter_for_dotted_lines_str
243 \str_set_eq:NN \l__nm_letter_for_dotted_lines_str \c_colon_str

```

`\NiceMatrixOptions` is the command of the `nicematrix` package to fix options at the document level. The scope of these specifications is the current TeX group.

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

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

```
246 \keys_define:nn { NiceMatrix / NiceMatrix }
247   {
248     last-col .code:n = \tl_if_empty:nTF {#1}
249                       { \__nm_error:n { last-col-empty-for-NiceMatrix } }
250                       { \int_set:Nn \l__nm_last_col_int { #1 } } ,
251     unknown .code:n = \__nm_error:n { Unknown~option~for~NiceMatrix }
252   }
```

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

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

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

```
255   c .code:n = \str_set:Nn \l__nm_pos_env_str c ,
256   t .code:n = \str_set:Nn \l__nm_pos_env_str t ,
257   b .code:n = \str_set:Nn \l__nm_pos_env_str b ,
```

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

```
258     last-col .code:n = \tl_if_empty:nF {#1}
259                   { \__nm_error:n { last-col-non-empty-for-NiceArray } }
260                   { \int_zero:N \l__nm_last_col_int ,
261     unknown .code:n = \__nm_error:n { Unknown~option~for~NiceArray }
262   }
263 \keys_define:nn { NiceMatrix / pNiceArray }
264   {
265     first-col .code:n = \int_zero:N \l__nm_first_col_int ,
266     last-col .code:n = \tl_if_empty:nF {#1}
267                       { \__nm_error:n { last-col-non-empty-for-NiceArray } }
268                       { \int_zero:N \l__nm_last_col_int ,
269     first-row .code:n = \int_zero:N \l__nm_first_row_int ,
270     last-row .int_set:N = \l__nm_last_row_int ,
271     last-row .default:n = -1 ,
272     unknown .code:n = \__nm_error:n { Unknown~option~for~NiceMatrix }
273   }
```

16.4 Important code used by `{NiceArrayWithDelims}`

The pseudo-environment `\@@_Cell:–\@@_end_Cell:` will be used to format the cells of the array. In the code, the affectations are global because this pseudo-environment will be used in the cells of a `\halign` (via an environment `{array}`).

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

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

```
276     \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.

```

277 \int_compare:nNnT \g__nm_col_int = \c_one_int
278 {
279   \int_compare:nNnT \l__nm_first_col_int = \c_one_int
280     \__nm_begin_of_row:
281 }
282 \int_gset:Nn \g__nm_col_total_int
283 { \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).

```

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

```

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.

```

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

```

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.

```

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

```


We want to compute in `\g_@@_max_cell_width_dim` the width of the widest cell of the array (except the cells of the “first column” and the “last column”).

```

324 \dim_gset:Nn \g__nm_max_cell_width_dim
325 { \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”.

```

326 \__nm_actualization_for_first_and_last_row:

```

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

```

327 \tikz
328 [
329   remember~picture ,
330   inner~sep = \c_zero_dim ,
331   minimum~width = \c_zero_dim ,
332   baseline
333 ]
334 \node
335 [
336   anchor = base ,
337   name = nm - \int_use:N \g__nm_env_int -
338           \int_use:N \g__nm_row_int -
339           \int_use:N \g__nm_col_int ,
340   alias =
341     \str_if_empty:NF \l__nm_name_str
342     {
343       \l__nm_name_str -
344       \int_use:N \g__nm_row_int -
345       \int_use:N \g__nm_col_int
346     }
347 ]
348 \bgroup
349 \box_use:N \l_tmpa_box
350 \egroup ;
351 }
352 \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.

```

353 \cs_new_protected:Nn \__nm_renewcolumnntype:nn
354 {
355   \newcolumnntype #1 [ 2 ]
356   {
357     > {
358       \hbox_set:Nw \l_tmpa_box
359       \__nm_Cell:
360     }
361     c
362     < {
363       \__nm_end_Cell:
364       \hbox_set_end:
365       #2
366       \hbox_set:Nn \l_tmpb_box
367       { \makebox [ ##2 ] [ ##1 ] { \box_use:N \l_tmpa_box } }
368       \dim_set:Nn \l_tmpa_dim { \box_dp:N \l_tmpb_box }
369       \box_move_down:nn \l_tmpa_dim
370       {
371         \vbox:n
372         {
373           \hbox_to_wd:nn { \box_wd:N \l_tmpb_box }

```

```

374         {
375             \hfil
376             \tikz [ remember~picture , overlay ]
377             \coordinate ( __nm~north~east ) ;
378         }
379     \hbox:n
380     {
381         \tikz [ remember~picture , overlay ]
382         \coordinate ( __nm~south~west ) ;
383         \box_move_up:nn \l_tmpa_dim { \box_use:N \l_tmpb_box }
384     }
385 }
386 }

```

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 `pdflatex` (why?).

```

387         \tikz [ remember~picture , overlay ]
388         \node
389         [
390             node~contents = { } ,
391             name = nm - \int_use:N \g__nm_env_int -
392                     \int_use:N \g__nm_row_int -
393                     \int_use:N \g__nm_col_int - w,
394             alias =
395                 \str_if_empty:NF \l__nm_name_str
396                 {
397                     \l__nm_name_str -
398                     \int_use:N \g__nm_row_int -
399                     \int_use:N \g__nm_col_int - w
400                 } ,
401             inner~sep = \c_zero_dim ,
402             fit = ( __nm~south~west ) ( __nm~north~east )
403         ]
404     ;
405 }
406 }
407 }

```

The argument of the following command `\@@_instruction_of_type:n` defined below is the type of the instruction (`Cdots`, `Vdots`, `Ddots`, etc.). This command writes in the corresponding `\g_@@_type_lines_tl` the instruction which will 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.

```

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

```

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).

```

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

```

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`.

```

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

```

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

```

433     [ \l__nm_pos_env_str ]
434 }

```

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

```

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

```

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.

```

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

```

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

```

452 \cs_new_protected:Npn \__nm_pre_array:
453 {

```

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}`).

```

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

```

We switch to a global version of the boolean `\g_@@_extra_nodes_bool`, because, in some circumstances, the boolean will be raised from inside a cell of the `\halign` (in particular in a column of type `w`).

```

459   \bool_gset_eq:NN \g__nm_extra_nodes_bool \l__nm_extra_nodes_bool

```

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

```

460   \cs_set:Npn \ialign
461   {
462       \bool_if:NTF \c__nm_colortbl_loaded_bool
463       {
464           \CT@everycr
465           {
466               \noalign { \cs_gset_eq:NN \CT@row@color \prg_do_nothing: }
467               \__nm_everycr:
468           }
469       }
470       { \everycr { \__nm_everycr: } }
471       \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`²⁴ 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`.

```

472   \dim_gzero_new:N \g__nm_dp_row_zero_dim
473   \dim_gset:Nn \g__nm_dp_row_zero_dim { \box_dp:N \@arstrutbox }
474   \dim_gzero_new:N \g__nm_ht_row_zero_dim
475   \dim_gset:Nn \g__nm_ht_row_zero_dim { \box_ht:N \@arstrutbox }
476   \dim_gzero_new:N \g__nm_ht_row_one_dim
477   \dim_gset:Nn \g__nm_ht_row_one_dim { \box_ht:N \@arstrutbox }
478   \dim_gzero_new:N \g__nm_dp_ante_last_row_dim
479   \dim_gset:Nn \g__nm_dp_ante_last_row_dim { \box_dp:N \@arstrutbox }
480   \dim_gzero_new:N \g__nm_ht_last_row_dim
481   \dim_gset:Nn \g__nm_ht_last_row_dim { \box_ht:N \@arstrutbox }
482   \dim_gzero_new:N \g__nm_dp_last_row_dim
483   \dim_gset:Nn \g__nm_dp_last_row_dim { \box_dp:N \@arstrutbox }

```

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`.²⁵

```

484   \cs_set:Npn \ialign
485   {
486       \everycr { }
487       \tabskip = \c_zero_skip
488       \halign
489   }

```

²⁴The option `small` of `nicematrix` changes (among other) the value of `\arraystretch`. This is done, of course, before the call of `{array}`.

²⁵The user will probably not employ directly `\ialign` in the array... but more likely environments that utilize `\ialign` internally (e.g.: `{substack}`).

```

490     \halign
491 }

```

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}.

```

492 \newcolumnntype L { > \_nm_Cell: l < \_nm_end_Cell: }
493 \newcolumnntype C { > \_nm_Cell: c < \_nm_end_Cell: }
494 \newcolumnntype R { > \_nm_Cell: r < \_nm_end_Cell: }

495 \cs_set_eq:NN \Ldots \_nm_Ldots
496 \cs_set_eq:NN \Cdots \_nm_Cdots
497 \cs_set_eq:NN \Vdots \_nm_Vdots
498 \cs_set_eq:NN \Ddots \_nm_Ddots
499 \cs_set_eq:NN \Iddots \_nm_Iddots
500 \cs_set_eq:NN \hdottedline \_nm_hdottedline:
501 \cs_set_eq:NN \Hspace \_nm_Hspace:
502 \cs_set_eq:NN \Hdotsfor \_nm_Hdotsfor:
503 \cs_set_eq:NN \multicolumn \_nm_multicolumn:nnn
504 \cs_set_eq:NN \Block \_nm_Block:
505 \bool_if:NT \l_nm_renew_dots_bool
506 {
507     \cs_set_eq:NN \ldots \_nm_Ldots
508     \cs_set_eq:NN \cdots \_nm_Cdots
509     \cs_set_eq:NN \vdots \_nm_Vdots
510     \cs_set_eq:NN \ddots \_nm_Ddots
511     \cs_set_eq:NN \iddots \_nm_Iddots
512     \cs_set_eq:NN \dots \_nm_Ldots
513     \cs_set_eq:NN \hdotsfor \_nm_Hdotsfor:
514 }

```

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

```

515 \seq_gclear_new:N \g_nm_multicolumn_cells_seq
516 \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).

```

517 \int_gzero_new:N \g_nm_row_int
518 \int_gset:Nn \g_nm_row_int { \l_nm_first_row_int - 1 }

```

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

```

519 \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.

```

520 \int_gzero_new:N \g_nm_col_int
521 \int_gzero_new:N \g_nm_col_total_int
522 \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:`.

```

523 \cs_set_eq:NN \NC@find@w \relax
524 \cs_set_eq:NN \NC@find@W \relax
525 \_nm_renewcolumnntype:nn w { }
526 \_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).

```

527 \tl_set_rescan:Nno
528 \l__nm_letter_for_dotted_lines_str { } \l__nm_letter_for_dotted_lines_str
529 \exp_args:NV \newcolumntype \l__nm_letter_for_dotted_lines_str
530 {
531 !
532 {
533 \skip_horizontal:n { 0.53 pt }
534 \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.

```

535 \int_compare:nNnT \g__nm_col_int > \g__nm_last_vdotted_col_int
536 {
537 \int_gset_eq:NN \g__nm_last_vdotted_col_int \g__nm_col_int
538 \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`.

```

539 { \__nm_vdottedline:n { \int_use:N \g__nm_col_int } }
540 }
541 }
542 }
543 \int_gzero_new:N \g__nm_last_vdotted_col_int
544 \bool_if:NT \c__nm_siunitx_loaded_bool \__nm_renew_NC@rewrite@S:
545 \int_gset:Nn \g__nm_last_vdotted_col_int { -1 }
546 \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.

```

547 \tl_gclear_new:N \g__nm_Cdots_lines_tl
548 \tl_gclear_new:N \g__nm_Ldots_lines_tl
549 \tl_gclear_new:N \g__nm_Vdots_lines_tl
550 \tl_gclear_new:N \g__nm_Ddots_lines_tl
551 \tl_gclear_new:N \g__nm_Iddots_lines_tl
552 \tl_gclear_new:N \g__nm_Hdotsfor_lines_tl
553 }

```

16.5 The environment `{NiceArrayWithDelims}`

```

554 \NewDocumentEnvironment { NiceArrayWithDelims } { m m O { } m ! O { } }
555 {
556 \str_if_empty:NT \g__nm_type_env_str
557 { \str_gset:Nn \g__nm_type_env_str { NiceArrayWithDelims } }

```

```

558 \__nm_adapt_S_column:
559 \__nm_test_if_math_mode:
560 \bool_if:NT \l__nm_in_env_bool { \__nm_fatal:n { Yet~in~env } }
561 \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).

```

562 \cs_if_exist:NT \tikz@library@external@loaded
563 {
564   \tikzset { external / export = false }
565   \cs_if_exist:NT \ifstandalone
566     { \tikzset { external / optimize = false } }
567 }

```

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

```

568 \int_gincr:N \g__nm_env_int
569 \bool_if:NF \l__nm_block_auto_columns_width_bool
570 { \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.

```

571 \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`.

```

572 \bool_if:NTF \l__nm_NiceArray_bool
573 { \keys_set:nn { NiceMatrix / NiceArray } }
574 { \keys_set:nn { NiceMatrix / pNiceArray } }
575 { #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).

```

576 \int_compare:nNnT \l__nm_last_row_int = { -1 }
577 {
578   \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.

```

579 \str_if_empty:NTF \l__nm_name_str
580 {
581   \cs_if_exist:cT { __nm_last_row_ \int_use:N \g__nm_env_int }
582   {
583     \int_set:Nn \l__nm_last_row_int
584       { \use:c { __nm_last_row_ \int_use:N \g__nm_env_int } }
585   }
586 }
587 {
588   \cs_if_exist:cT { __nm_last_row_ \l__nm_name_str }
589   {
590     \int_set:Nn \l__nm_last_row_int
591       { \use:c { __nm_last_row_ \l__nm_name_str } }
592   }
593 }
594 }

```

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

```

595 \__nm_pre_array:

```

We compute the width of the two delimiters.

```

596 \dim_gzero_new:N \g__nm_left_delim_dim
597 \dim_gzero_new:N \g__nm_right_delim_dim
598 \bool_if:NTF \l__nm_NiceArray_bool
599 {
600   \dim_gset:Nn \g__nm_left_delim_dim { 2 \arraycolsep }
601   \dim_gset:Nn \g__nm_right_delim_dim { 2 \arraycolsep }
602 }
603 {

```

```

604     \group_begin:
605     \dim_set_eq:Nn \nulldelimiterspace \c_zero_dim
606     \hbox_set:Nn \l_tmpa_box
607     {
608         \c_math_toggle_token
609         \left #1 \vcenter to 3 cm { } \right.
610         \c_math_toggle_token
611     }
612     \dim_gset:Nn \g__nm_left_delim_dim { \box_wd:N \l_tmpa_box }
613     \hbox_set:Nn \l_tmpa_box
614     {
615         \dim_set_eq:Nn \nulldelimiterspace \c_zero_dim
616         \c_math_toggle_token
617         \left. \vcenter to 3 cm { } \right #2
618         \c_math_toggle_token
619     }
620     \dim_gset:Nn \g__nm_right_delim_dim { \box_wd:N \l_tmpa_box }
621     \group_end:
622 }
623

```

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

```

624     \box_clear_new:N \l__nm_the_array_box

```

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

```

625     \tl_set:Nn \l_tmpa_tl { #4 }
626     \int_compare:nNnTF \l__nm_first_col_int = \c_zero_int
627     { \tl_put_left:NV \l_tmpa_tl \c__nm_preamble_first_col_tl }
628     {
629         \bool_if:NT \l__nm_NiceArray_bool
630         {
631             \bool_if:NF \l__nm_exterior_arraycolsep_bool
632             { \tl_put_left:Nn \l_tmpa_tl { @ { } } }
633         }
634     }
635     \int_compare:nNnTF \l__nm_last_col_int > { -1 }
636     { \tl_put_right:NV \l_tmpa_tl \c__nm_preamble_last_col_tl }
637     {
638         \bool_if:NT \l__nm_NiceArray_bool
639         {
640             \bool_if:NF \l__nm_exterior_arraycolsep_bool
641             { \tl_put_right:Nn \l_tmpa_tl { @ { } } }
642         }
643     }

```

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).

```

644     \hbox_set:Nw \l__nm_the_array_box
645     \skip_horizontal:n \l__nm_left_margin_dim
646     \skip_horizontal:n \l__nm_extra_left_margin_dim
647     \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`).

```

648     \exp_args:NV \__nm_array: \l_tmpa_tl
649 }

```

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


```

650 {
651   \bool_if:nT
652   {
653     ( \l__nm_auto_columns_width_bool && ! \l__nm_block_auto_columns_width_bool )
654     || \dim_compare_p:nNn \l__nm_columns_width_dim > \c_zero_dim
655   }
656   {
657     \crrc
658     \int_compare:nNnT \l__nm_first_col_int = 0 { \omit & }
659     \omit

```

First, we put a “col” node on the left of the first column (of course, we have to do that *after* the \omit).

```

660     \tikz [ remember~picture , overlay ]
661     \coordinate [ name = nm - \int_use:N \g__nm_env_int - col - 0 ] ;

```

We compute in \g_tmpa_dim the common width of the columns. We use a global variable because we are in a cell of an \halign and that we have to use this variable in other cells (of the same row). The affectation of \g_tmpa_dim, like all the affectations, must be done after the \omit of the cell.

```

662     \bool_if:nTF
663     {
664       \l__nm_auto_columns_width_bool
665       && ! \l__nm_block_auto_columns_width_bool
666     }
667     {
668       \dim_gset:Nn \g_tmpa_dim
669       { \g__nm_max_cell_width_dim + 2 \arraycolsep }
670     }
671     {
672       \dim_gset:Nn \g_tmpa_dim
673       { \l__nm_columns_width_dim + 2 \arraycolsep }
674     }
675     \skip_horizontal:N \g_tmpa_dim

```

We begin a loop over the columns. The integer \g_tmpa_int will be the number of columns of the current cell. This integer is not used to fix the width of the column (since all the columns have the same width equal to \g_@@_tmpa_dim) but for the Tikz nodes.

```

676     \int_gset:Nn \g_tmpa_int 1
677     \bool_if:nTF \g__nm_last_col_found_bool
678     { \prg_replicate:nn { \g__nm_col_total_int - 2 } }
679     { \prg_replicate:nn { \g__nm_col_total_int - 1 } }
680     {
681       &
682       \omit

```

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

```

683       \int_gincr:N \g_tmpa_int
684       \skip_horizontal:N \g_tmpa_dim

```

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

```

685       \tikz [ remember~picture , overlay ]
686       \coordinate
687       [
688         name = nm - \int_use:N \g__nm_env_int -
689         col - \int_use:N \g_tmpa_int
690       ] ;
691     }
692   }
693   \endarray
694   \c_math_toggle_token
695   \skip_horizontal:n \l__nm_right_margin_dim
696   \skip_horizontal:n \l__nm_extra_right_margin_dim
697   \hbox_set_end:

698   \int_compare:nNnT \l__nm_last_row_int > { -2 }
699   {

```

```

700     \bool_if:NF \l__nm_last_row_without_value_bool
701     {
702         \int_compare:nNnF \l__nm_last_row_int = \g__nm_row_int
703         {
704             \__nm_error:n { Wrong-last-row }
705             \int_gset_eq:NN \l__nm_last_row_int \g__nm_row_int
706         }
707     }
708 }

```

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).

```

709     \int_compare:nNnTF \l__nm_first_row_int = \c_zero_int
710     {
711         \dim_set:Nn \l_tmpa_dim
712         {
713             \g__nm_ht_row_one_dim + \g__nm_dp_row_zero_dim
714             + \lineskip
715             + \g__nm_ht_row_zero_dim - \g__nm_ht_row_one_dim
716         }
717     }
718     { \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”.²⁶

```

719     \int_compare:nNnTF \l__nm_last_row_int > { -2 }
720     {
721         \dim_set:Nn \l_tmpb_dim
722         {
723             \g__nm_ht_last_row_dim + \g__nm_dp_ante_last_row_dim
724             + \lineskip
725             + \g__nm_dp_last_row_dim - \g__nm_dp_ante_last_row_dim
726         }
727     }
728     { \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. 44).

```

729     \int_compare:nNnT \l__nm_first_col_int = \c_zero_int
730     {
731         \skip_horizontal:n \arraycolsep
732         \skip_horizontal:n \g__nm_width_first_col_dim
733     }

```

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}`.

```

734     \bool_if:NTF \l__nm_NiceArray_bool
735     {
736         \int_compare:nNnT \l__nm_first_row_int = \c_zero_int
737         {
738             \str_if_eq:VnTF \l__nm_pos_env_str { t }
739             {
740                 \box_move_up:nn
741                 { \l_tmpa_dim - \g__nm_ht_row_zero_dim + \g__nm_ht_row_one_dim }
742             }
743         }
744         {
745             \int_compare:nNnT \l__nm_last_row_int > 0
746             {

```

²⁶A 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`).

```

747         \str_if_eq:VnT \l__nm_pos_env_str { b }
748         {
749             \box_move_down:nn
750             {
751                 \l_tmpb_dim
752                 - \g__nm_dp_last_row_dim + \g__nm_dp_ante_last_row_dim
753             }
754         }
755     }
756 }
757 { \box_use_drop:N \l__nm_the_array_box }
758 }

```

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

```

759 {
760     \hbox_set:Nn \l_tmpa_box
761     {
762         \c_math_toggle_token
763         \left #1
764         \vcenter
765         {

```

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

```

766         \skip_vertical:n { - \l_tmpa_dim }
767         \hbox:n
768         {
769             \skip_horizontal:n { - \arraycolsep }
770             \box_use_drop:N \l__nm_the_array_box
771             \skip_horizontal:n { - \arraycolsep }
772         }

```

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

```

773         \skip_vertical:n { - \l_tmpb_dim }
774     }
775     \right #2
776     \c_math_toggle_token
777 }
778 \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }
779 \box_set_dp:Nn \l_tmpa_box { \box_dp:N \l_tmpa_box + \l_tmpb_dim }
780 \box_use_drop:N \l_tmpa_box
781 }

```

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. 45).

```

782     \bool_if:NT \g__nm_last_col_found_bool
783     {
784         \skip_horizontal:n \g__nm_width_last_col_dim
785         \skip_horizontal:n \arraycolsep
786     }
787     \__nm_after_array:
788 }

```

This is the end of the environment {NiceArrayWithDelims}.

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

```

789 \tl_const:Nn \c__nm_preamble_first_col_tl
790 {
791     >
792     {
793         \__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.

```

794         \hbox_set:Nw \l_tmpa_box
795         \c_math_toggle_token
796         \bool_if:NT \l__nm_small_bool \scriptstyle

```

```

797     \l__nm_code_for_first_col_tl
798   }
799   l
800   <
801   {
802     \c_math_toggle_token
803     \hbox_set_end:
804     \__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.

```

805     \dim_gset:Nn \g__nm_width_first_col_dim
806     {
807       \dim_max:nn
808       \g__nm_width_first_col_dim
809       { \box_wd:N \l_tmpa_box }
810     }

```

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

```

811     \hbox_overlap_left:n
812     {
813       \tikz
814       [
815         remember~picture ,
816         inner~sep = \c_zero_dim ,
817         minimum~width = \c_zero_dim ,
818         baseline
819       ]
820       \node
821       [
822         anchor = base ,
823         name =
824         nm -
825         \int_use:N \g__nm_env_int -
826         \int_use:N \g__nm_row_int -
827         0 ,
828         alias =
829         \str_if_empty:NF \l__nm_name_str
830         {
831           \l__nm_name_str -
832           \int_use:N \g__nm_row_int -
833           0
834         }
835       ]
836       { \box_use:N \l_tmpa_box } ;
837     \skip_horizontal:n
838     {
839       \g__nm_left_delim_dim +
840       \l__nm_left_margin_dim +
841       \l__nm_extra_left_margin_dim
842     }
843   }
844   \skip_horizontal:n { - 2 \arraycolsep }
845 }
846 }

```

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

```

847 \tl_const:Nn \c__nm_preamble_last_col_tl
848 {
849   >
850   {

```

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

```

851     \bool_gset_true:N \g__nm_last_col_found_bool
852     \int_gincr:N \g__nm_col_int
853     \int_gset:Nn \g__nm_col_total_int

```

```
854 { \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.

```
855 \hbox_set:Nw \l_tmpa_box
856 \c_math_toggle_token
857 \bool_if:NT \l__nm_small_bool \scriptstyle
858 \l__nm_code_for_last_col_tl
859 }
860 1
861 <
862 {
863 \c_math_toggle_token
864 \hbox_set_end:
865 \__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.

```
866 \dim_gset:Nn \g__nm_width_last_col_dim
867 {
868 \dim_max:nn
869 \g__nm_width_last_col_dim
870 { \box_wd:N \l_tmpa_box }
871 }
872 \skip_horizontal:n { - 2 \arraycolsep }
```

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

```
873 \hbox_overlap_right:n
874 {
875 \skip_horizontal:n
876 {
877 \g__nm_right_delim_dim +
878 \l__nm_right_margin_dim +
879 \l__nm_extra_right_margin_dim
880 }
881 \tikz
882 [
883 remember~picture ,
884 inner~sep = \c_zero_dim ,
885 minimum~width = \c_zero_dim ,
886 baseline
887 ]
888 \node
889 [
890 anchor = base ,
891 name =
892 nm -
893 \int_use:N \g__nm_env_int -
894 \int_use:N \g__nm_row_int -
895 \int_use:N \g__nm_col_int ,
896 alias =
897 \str_if_empty:NF \l__nm_name_str
898 {
899 \l__nm_name_str -
900 \int_use:N \g__nm_row_int -
901 \int_use:N \g__nm_col_int
902 }
903 ]
904 { \box_use:N \l_tmpa_box } ;
905 }
906 }
907 }
```

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.

```

908 \NewDocumentEnvironment { NiceArray } { }
909 {
910     \bool_set_true:N \l__nm_NiceArray_bool
911     \str_if_empty:NT \g__nm_type_env_str
912     { \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).

```

913     \NiceArrayWithDelims . .
914 }
915 { \endNiceArrayWithDelims }

```

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

```

916 \NewDocumentEnvironment { pNiceArray } { }
917 {
918     \str_if_empty:NT \g__nm_type_env_str
919     { \str_gset:Nn \g__nm_type_env_str { pNiceArray } }
920     \__nm_test_if_math_mode:
921     \NiceArrayWithDelims ( )
922 }
923 { \endNiceArrayWithDelims }

924 \NewDocumentEnvironment { bNiceArray } { }
925 {
926     \str_if_empty:NT \g__nm_type_env_str
927     { \str_gset:Nn \g__nm_type_env_str { NiceArray } }
928     \__nm_test_if_math_mode:
929     \NiceArrayWithDelims [ ]
930 }
931 { \endNiceArrayWithDelims }

932 \NewDocumentEnvironment { BNiceArray } { }
933 {
934     \str_if_empty:NT \g__nm_type_env_str
935     { \str_gset:Nn \g__nm_type_env_str { BNiceArray } }
936     \__nm_test_if_math_mode:
937     \NiceArrayWithDelims \{ \}
938 }
939 { \endNiceArrayWithDelims }

940 \NewDocumentEnvironment { vNiceArray } { }
941 {
942     \str_if_empty:NT \g__nm_type_env_str
943     { \str_gset:Nn \g__nm_type_env_str { vNiceArray } }
944     \__nm_test_if_math_mode:
945     \NiceArrayWithDelims | |
946 }
947 { \endNiceArrayWithDelims }

948 \NewDocumentEnvironment { VNiceArray } { }
949 {
950     \str_if_empty:NT \g__nm_type_env_str
951     { \str_gset:Nn \g__nm_type_env_str { VNiceArray } }
952     \__nm_test_if_math_mode:
953     \NiceArrayWithDelims \| \|
954 }
955 { \endNiceArrayWithDelims }

```

16.6 The environment `{NiceMatrix}` and its variants

```

956 \cs_new_protected:Npn \__nm_define_env:n #1
957 {

```

```

958 \NewDocumentEnvironment { #1 NiceMatrix } { ! O { } }
959 {
960   \str_gset:Nn \g__nm_type_env_str { #1 NiceMatrix }
961   \keys_set:nn { NiceMatrix / NiceMatrix } { ##1 }
962   \begin { #1 NiceArray }
963     {
964       *
965       {
966         \int_compare:nNnTF \l__nm_last_col_int = { -1 }
967         \c@MaxMatrixCols
968         { \int_eval:n { \l__nm_last_col_int - 1 } }
969       }
970       C
971     }
972   }
973   { \end { #1 NiceArray } }
974 }
975 \__nm_define_env:n { }
976 \__nm_define_env:n p
977 \__nm_define_env:n b
978 \__nm_define_env:n B
979 \__nm_define_env:n v
980 \__nm_define_env:n V

```

16.7 Automatic width of the cells

16.8 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.

```

981 \prg_set_conditional:Npnn \__nm_if_not_empty_cell:nn #1 #2 { T , TF }
982 {

```

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.

```

983   \bool_set_false:N \l_tmpa_bool
984   \cs_if_exist:cTF
985     { __nm _dotted _ \int_use:N #1 - \int_use:N #2 }
986     \prg_return_true:
987     {

```

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.

```

988   \cs_if_free:cTF
989     {
990       pgf@sh@ns@nm -
991       \int_use:N \g__nm_env_int -
992       \int_use:N #1 -
993       \int_use:N #2
994     }
995     { \prg_return_false: }
996     {

```

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.

```

997         \bool_set_false:N \l_tmpa_bool
998         \cs_if_exist:cT
999             { __nm _ empty _ \int_use:N #1 - \int_use:N #2 }
1000             {
1001                 \int_compare:nNnT
1002                     { \use:c { __nm _ empty _ \int_use:N #1 - \int_use:N #2 } }
1003                     =
1004                     \g__nm_env_int
1005                     { \bool_set_true:N \l_tmpa_bool }
1006             }
1007         \bool_if:NTF \l_tmpa_bool
1008         \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.

```

1009         {
1010             \begin { pgfpicture }

```

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

```

1011         \tl_set:Nx \l_tmpa_tl
1012             {
1013                 nm -
1014                 \int_use:N \g__nm_env_int -
1015                 \int_use:N #1 -
1016                 \int_use:N #2
1017             }
1018         \pgfpointanchor \l_tmpa_tl { east }
1019         \dim_gset:Nn \g_tmpa_dim \pgf@x
1020         \pgfpointanchor \l_tmpa_tl { west }
1021         \dim_gset:Nn \g_tmpb_dim \pgf@x
1022         \end { pgfpicture }
1023         \dim_compare:nNnTF
1024             { \dim_abs:n { \g_tmpb_dim - \g_tmpa_dim } } < { 0.5 pt }
1025             \prg_return_false:
1026             \prg_return_true:
1027         }
1028     }
1029 }
1030 }

```

16.9 After the construction of the array

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

```

1031 \cs_new_protected:Nn \__nm_after_array:
1032 {
1033     \int_compare:nNnTF \g__nm_row_int > \c_zero_int
1034         \__nm_after_array_i:
1035         { \__nm_error:n { Zero-row } }
1036 }

```


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

```

1037 \cs_new_protected:Nn \__nm_after_array_i:
1038 {
1039   \group_begin:
1040   \cs_if_exist:NT \tikz@library@external@loaded
1041     { \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”.²⁷

```

1042   \int_gset_eq:NN \g__nm_col_int \g__nm_col_total_int
1043   \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.

```

1044   \int_gset_eq:NN \g__nm_row_total_int \g__nm_row_int
1045   \int_compare:nNnT \l__nm_last_row_int > { -1 }
1046     { \int_gsub:Nn \g__nm_row_int \c_one_int }

```

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

```

1047   \bool_if:NT \l__nm_last_row_without_value_bool
1048   {
1049     \iow_now:Nn \@mainaux \ExplSyntaxOn
1050     \iow_now:Nx \@mainaux
1051     {
1052       \cs_gset:cpn { __nm_last_row_ \int_use:N \g__nm_env_int }
1053       { \int_use:N \g__nm_row_total_int }
1054     }

```

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.

```

1055     \str_if_empty:NF \l__nm_name_str
1056     {
1057       \iow_now:Nx \@mainaux
1058       {
1059         \cs_gset:cpn { __nm_last_row_ \l__nm_name_str }
1060         { \int_use:N \g__nm_row_total_int }
1061       }
1062     }
1063     \iow_now:Nn \@mainaux \ExplSyntaxOff
1064   }

```

By default, the diagonal lines will be parallelized²⁸. 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.

```

1065   \bool_if:NT \l__nm_parallelize_diags_bool
1066   {
1067     \int_zero_new:N \l__nm_ddots_int
1068     \int_zero_new:N \l__nm_iddots_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.

```

1069     \dim_zero_new:N \l__nm_delta_x_one_dim
1070     \dim_zero_new:N \l__nm_delta_y_one_dim
1071     \dim_zero_new:N \l__nm_delta_x_two_dim
1072     \dim_zero_new:N \l__nm_delta_y_two_dim
1073   }

```

²⁷We remind that the potential “first column” has the number 0.

²⁸It’s possible to use the option `parallelize-diags` to disable this parallelization.

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).

```

1074 \bool_if:NT \g__nm_extra_nodes_bool \__nm_create_extra_nodes:
1075 \int_zero_new:N \l__nm_initial_i_int
1076 \int_zero_new:N \l__nm_initial_j_int
1077 \int_zero_new:N \l__nm_final_i_int
1078 \int_zero_new:N \l__nm_final_j_int
1079 \bool_set_false:N \l__nm_initial_open_bool
1080 \bool_set_false:N \l__nm_final_open_bool

```

If the option `small` is used, the values `\l_@@_radius_dim` and `\l_@@_inter_dots_dim` (used to draw the dotted lines) are changed.

```

1081 \bool_if:NT \l__nm_small_bool
1082 {
1083   \dim_set:Nn \l__nm_radius_dim { 0.37 pt }
1084   \dim_set:Nn \l__nm_inter_dots_dim { 0.25 em }
1085 }

```

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

```

1086 \g__nm_Hdotsfor_lines_tl
1087 \g__nm_Vdots_lines_tl
1088 \g__nm_Ddots_lines_tl
1089 \g__nm_Iddots_lines_tl
1090 \g__nm_Cdots_lines_tl
1091 \g__nm_Ldots_lines_tl

```

Now, the code-after.

```

1092 \tikzset
1093 {
1094   every-picture / .style =
1095   {
1096     overlay ,
1097     remember-picture ,
1098     name-prefix = nm - \int_use:N \g__nm_env_int -
1099   }
1100 }
1101 \cs_set_eq:NN \line \__nm_line:nn
1102 \g__nm_code_after_tl
1103 \tl_gclear:N \g__nm_code_after_tl
1104 \group_end:
1105 \str_gclear:N \g__nm_type_env_str
1106 }

```

A dotted line will be said *open* in one of its extremities when it stops on the edge of the matrix and *closed* otherwise. In the following matrix, the dotted line is closed on its left extremity and open on its right.

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

For a closed extremity, we use the normal node and for a open one, we use the “medium node” or, if it exists, the `w` node (the medium and large nodes are created with `\@@_create_extra_nodes:` if they have not been created yet).

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

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

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

This command computes:

- `\l_@@_initial_i_int` and `\l_@@_initial_j_int` which are the coordinates of one extremity of the line;
- `\l_@@_final_i_int` and `\l_@@_final_j_int` which are the coordinates of the other extremity of the line;
- `\l_@@_initial_open_bool` and `\l_@@_final_open_bool` to indicate whether the extremities are open or not.

```
1107 \cs_new_protected:Nn \__nm_find_extremities_of_line:nnnn
1108 {
```

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

```
1109 \cs_set:cpn { __nm _ dotted _ #1 - #2 } { }
```

Initialization of variables.

```
1110 \int_set:Nn \l__nm_initial_i_int { #1 }
1111 \int_set:Nn \l__nm_initial_j_int { #2 }
1112 \int_set:Nn \l__nm_final_i_int { #1 }
1113 \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.

```
1114 \bool_set_false:N \l__nm_stop_loop_bool
1115 \bool_do_until:Nn \l__nm_stop_loop_bool
1116 {
1117   \int_add:Nn \l__nm_final_i_int { #3 }
1118   \int_add:Nn \l__nm_final_j_int { #4 }
```

We test if we are still in the matrix.

```
1119   \bool_set_false:N \l__nm_final_open_bool
1120   \int_compare:nNnTF \l__nm_final_i_int > \g__nm_row_int
1121   {
1122     \int_compare:nNnT { #3 } = 1
1123     { \bool_set_true:N \l__nm_final_open_bool }
1124   }
1125   {
1126     \int_compare:nNnTF \l__nm_final_j_int < 1
1127     {
1128       \int_compare:nNnT { #4 } = { -1 }
1129       { \bool_set_true:N \l__nm_final_open_bool }
1130     }
1131     {
1132       \int_compare:nNnT \l__nm_final_j_int > \g__nm_col_int
1133       {
1134         \int_compare:nNnT { #4 } = 1
1135         { \bool_set_true:N \l__nm_final_open_bool }
1136       }
1137     }
1138   }
1139   \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.

```
1140   {
```

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).

```

1141         \int_sub:Nn \l__nm_final_i_int { #3 }
1142         \int_sub:Nn \l__nm_final_j_int { #4 }
1143         \bool_set_true:N \l__nm_stop_loop_bool
1144     }

```

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`.

```

1145     {
1146         \__nm_if_not_empty_cell:nnTF \l__nm_final_i_int \l__nm_final_j_int
1147         { \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.

```

1148         {
1149             \cs_set:cpn
1150             {
1151                 __nm_dotted_
1152                 \int_use:N \l__nm_final_i_int -
1153                 \int_use:N \l__nm_final_j_int
1154             }
1155             { }
1156         }
1157     }
1158 }

```

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).

```

1159     \cs_if_free:cT
1160     {
1161         pgf@sh@ns@nm -
1162         \int_use:N \g__nm_env_int -
1163         \int_use:N \l__nm_final_i_int -
1164         \int_use:N \l__nm_final_j_int
1165     }
1166     {
1167         \bool_if:NF \l__nm_final_open_bool
1168         {
1169             \msg_error:nnx { nicematrix } { Impossible~line }
1170             { \int_use:N \l__nm_final_i_int }
1171             \bool_set_true:N \l__nm_impossible_line_bool
1172         }
1173     }

```

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

```

1174     \bool_set_false:N \l__nm_stop_loop_bool
1175     \bool_do_until:Nn \l__nm_stop_loop_bool
1176     {
1177         \int_sub:Nn \l__nm_initial_i_int { #3 }
1178         \int_sub:Nn \l__nm_initial_j_int { #4 }
1179         \bool_set_false:N \l__nm_initial_open_bool
1180         \int_compare:nNnTF \l__nm_initial_i_int < 1
1181         {
1182             \int_compare:nNnT { #3 } = 1
1183             { \bool_set_true:N \l__nm_initial_open_bool }
1184         }
1185     }

```

```

1186 \int_compare:nNnTF \l__nm_initial_j_int < 1
1187 {
1188     \int_compare:nNnT { #4 } = 1
1189     { \bool_set_true:N \l__nm_initial_open_bool }
1190 }
1191 {
1192     \int_compare:nNnT \l__nm_initial_j_int > \g__nm_col_int
1193     {
1194         \int_compare:nNnT { #4 } = { -1 }
1195         { \bool_set_true:N \l__nm_initial_open_bool }
1196     }
1197 }
1198 }
1199 \bool_if:NTF \l__nm_initial_open_bool
1200 {
1201     \int_add:Nn \l__nm_initial_i_int { #3 }
1202     \int_add:Nn \l__nm_initial_j_int { #4 }
1203     \bool_set_true:N \l__nm_stop_loop_bool
1204 }
1205 {
1206     \__nm_if_not_empty_cell:nNnTF
1207     \l__nm_initial_i_int \l__nm_initial_j_int
1208     { \bool_set_true:N \l__nm_stop_loop_bool }
1209     {
1210         \cs_set:cpn
1211         {
1212             __nm_dotted_
1213             \int_use:N \l__nm_initial_i_int -
1214             \int_use:N \l__nm_initial_j_int
1215         }
1216         { }
1217     }
1218 }
1219 }

```

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).

```

1220 \cs_if_free:cT
1221 {
1222     pgf@sh@ns@nm -
1223     \int_use:N \g__nm_env_int -
1224     \int_use:N \l__nm_initial_i_int -
1225     \int_use:N \l__nm_initial_j_int
1226 }
1227 {
1228     \bool_if:NF \l__nm_initial_open_bool
1229     {
1230         \msg_error:nnx { nicematrix } { Impossible~line }
1231         { \int_use:N \l__nm_initial_i_int }
1232         \bool_set_true:N \l__nm_impossible_line_bool
1233     }
1234 }

```

If we have at least one open extremity, we create the “medium nodes” in the matrix²⁹. We remind that, when used once, the command `\@@_create_extra_nodes:` becomes no-op in the current TeX group.

```

1235 \bool_if:nT \l__nm_initial_open_bool \__nm_create_extra_nodes:
1236 \bool_if:NT \l__nm_final_open_bool \__nm_create_extra_nodes:
1237 }

```

²⁹We 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”.

The command `\@@_retrieve_coords:nn` retrieves the Tikz coordinates of the two extremities of the dotted line we will have to draw³⁰. This command has four implicit arguments which are `\l_@@_initial_i_int`, `\l_@@_initial_j_int`, `\l_@@_final_i_int` and `\l_@@_final_j_int`. The two arguments of the command `\@@_retrieve_coords:nn` are the suffix and the anchor that must be used for the two nodes.

The coordinates are stored in `\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}`.

```

1238 \cs_new_protected:Nn \__nm_retrieve_coords:nn
1239 {
1240   \dim_gzero_new:N \g__nm_x_initial_dim
1241   \dim_gzero_new:N \g__nm_y_initial_dim
1242   \dim_gzero_new:N \g__nm_x_final_dim
1243   \dim_gzero_new:N \g__nm_y_final_dim
1244   \begin { tikzpicture } [ remember-picture ]
1245     \tikz@parse@node \pgfutil@firstofone
1246       ( nm - \int_use:N \g__nm_env_int -
1247         \int_use:N \l__nm_initial_i_int -
1248         \int_use:N \l__nm_initial_j_int #1 )
1249     \dim_gset:Nn \g__nm_x_initial_dim \pgf@x
1250     \dim_gset:Nn \g__nm_y_initial_dim \pgf@y
1251     \tikz@parse@node \pgfutil@firstofone
1252       ( nm - \int_use:N \g__nm_env_int -
1253         \int_use:N \l__nm_final_i_int -
1254         \int_use:N \l__nm_final_j_int #2 )
1255     \dim_gset:Nn \g__nm_x_final_dim \pgf@x
1256     \dim_gset:Nn \g__nm_y_final_dim \pgf@y
1257   \end { tikzpicture }
1258 }
1259 \cs_generate_variant:Nn \__nm_retrieve_coords:nn { x x }

```

```

1260 \cs_new_protected:Nn \__nm_draw_Ldots:nn
1261 {
1262   \cs_if_free:cT { __nm _ dotted _ #1 - #2 }
1263   {
1264     \bool_set_false:N \l__nm_impossible_line_bool
1265     \__nm_find_extremities_of_line:nnnn { #1 } { #2 } \c_zero_int \c_one_int
1266     \bool_if:NF \l__nm_impossible_line_bool \__nm_actually_draw_Ldots:
1267   }
1268 }

```

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`.

```

1269 \cs_new_protected:Nn \__nm_actually_draw_Ldots:
1270 {
1271   \__nm_retrieve_coords:xx
1272   {
1273     \bool_if:NTF \l__nm_initial_open_bool
1274     {

```

If a `w` node exists we use the `w` node for the extremity.

```

1275       \cs_if_exist:cTF
1276       {
1277         pgf@sh@ns@nm
1278         - \int_use:N \g__nm_env_int
1279         - \int_use:N \l__nm_initial_i_int
1280         - \int_use:N \l__nm_initial_j_int - w
1281       }

```

³⁰In fact, with diagonal lines, or vertical lines in columns of type L or R, an adjustment of one of the coordinates may be done.

```

1282         { - w.base~west }
1283         { - medium.base~west }
1284     }
1285     { .base~east }
1286 }
1287 {
1288     \bool_if:NTF \l__nm_final_open_bool
1289     {
1290         \cs_if_exist:cTF
1291         {
1292             pgf@sh@ns@nm
1293             - \int_use:N \g__nm_env_int
1294             - \int_use:N \l__nm_final_i_int
1295             - \int_use:N \l__nm_final_j_int - w
1296         }
1297         { - w.base~east }
1298         { - medium.base~east }
1299     }
1300     { .base~west }
1301 }
1302 \bool_if:NT \l__nm_initial_open_bool
1303 { \dim_gset_eq:NN \g__nm_y_initial_dim \g__nm_y_final_dim }
1304 \bool_if:NT \l__nm_final_open_bool
1305 { \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.

```

1306 \dim_gadd:Nn \g__nm_y_initial_dim { 0.53 pt }
1307 \dim_gadd:Nn \g__nm_y_final_dim { 0.53 pt }
1308 \__nm_draw_tikz_line:
1309 }

```

```

1310 \cs_new_protected:Nn \__nm_draw_Cdots:nn
1311 {
1312     \cs_if_free:cT { __nm _ dotted _ #1 - #2 }
1313     {
1314         \bool_set_false:N \l__nm_impossible_line_bool
1315         \__nm_find_extremities_of_line:nnnn { #1 } { #2 } \c_zero_int \c_one_int
1316         \bool_if:NF \l__nm_impossible_line_bool
1317         {
1318             \__nm_retrieve_coords:xx
1319             {
1320                 \bool_if:NTF \l__nm_initial_open_bool
1321                 {
1322                     \cs_if_exist:cTF
1323                     {
1324                         pgf@sh@ns@nm
1325                         - \int_use:N \g__nm_env_int
1326                         - \int_use:N \l__nm_initial_i_int
1327                         - \int_use:N \l__nm_initial_j_int - w
1328                     }
1329                     { - w.mid~west }
1330                     { - medium.mid~west }
1331                 }
1332                 { .mid~east }
1333             }
1334             {
1335                 \bool_if:NTF \l__nm_final_open_bool
1336                 {
1337                     \cs_if_exist:cTF
1338                     {
1339                         pgf@sh@ns@nm
1340                         - \int_use:N \g__nm_env_int

```

```

1341         - \int_use:N \l__nm_final_i_int
1342         - \int_use:N \l__nm_final_j_int - w
1343     }
1344     { - w.mid~east }
1345     { - medium.mid~east }
1346 }
1347 { .mid~west }
1348 }
1349 \bool_if:NT \l__nm_initial_open_bool
1350 { \dim_gset_eq:NN \g__nm_y_initial_dim \g__nm_y_final_dim }
1351 \bool_if:NT \l__nm_final_open_bool
1352 { \dim_gset_eq:NN \g__nm_y_final_dim \g__nm_y_initial_dim }
1353 \__nm_draw_tikz_line:
1354 }
1355 }
1356 }

```

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?).

```

1357 \cs_new_protected:Nn \__nm_draw_Vdots:nn
1358 {
1359     \cs_if_free:cT { __nm _ dotted _ #1 - #2 }
1360     {
1361         \bool_set_false:N \l__nm_impossible_line_bool
1362         \__nm_find_extremities_of_line:nnnn { #1 } { #2 } \c_one_int \c_zero_int
1363         \bool_if:NF \l__nm_impossible_line_bool
1364         {
1365             \__nm_retrieve_coords:xx
1366             {
1367                 \bool_if:NTF \l__nm_initial_open_bool
1368                 { - medium.north~west }
1369                 { .south~west }
1370             }
1371             {
1372                 \bool_if:NTF \l__nm_final_open_bool
1373                 { - medium.south~west }
1374                 { .north~west }
1375             }
1376         }
1377     }
1378 }

```

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

```

1376 \bool_set:Nn \l_tmpa_bool
1377 { \dim_compare_p:nNn \g__nm_x_initial_dim = \g__nm_x_final_dim }
1378 \__nm_retrieve_coords:xx
1379 {
1380     \bool_if:NTF \l__nm_initial_open_bool
1381     { - medium.north }
1382     { .south }
1383 }
1384 {
1385     \bool_if:NTF \l__nm_final_open_bool
1386     { - medium.south }
1387     { .north }
1388 }

```

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

```

1389 \bool_set:Nn \l_tmpb_bool
1390 { \dim_compare_p:nNn \g__nm_x_initial_dim = \g__nm_x_final_dim }
1391 \bool_if:NF \l_tmpb_bool
1392 {
1393     \dim_gset:Nn \g__nm_x_initial_dim

```



```

1394         {
1395             \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
1396             \g__nm_x_initial_dim \g__nm_x_final_dim
1397         }
1398         \dim_gset_eq:NN \g__nm_x_final_dim \g__nm_x_initial_dim
1399     }
1400     \__nm_draw_tikz_line:
1401 }
1402 }
1403 }

```

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.

```

1404 \cs_new_protected:Nn \__nm_draw_Ddots:nn
1405 {
1406     \cs_if_free:cT { __nm _ dotted _ #1 - #2 }
1407     {
1408         \bool_set_false:N \l__nm_impossible_line_bool
1409         \__nm_find_extremities_of_line:nnnn { #1 } { #2 } \c_one_int \c_one_int
1410         \bool_if:NF \l__nm_impossible_line_bool
1411         {
1412             \__nm_retrieve_coords:xx
1413             {
1414                 \bool_if:NTF \l__nm_initial_open_bool
1415                 { - medium.north-west }
1416                 { .south-east }
1417             }
1418             {
1419                 \bool_if:NTF \l__nm_final_open_bool
1420                 { - medium.south-east }
1421                 { .north-west }
1422             }
1423         }
1424     }
1425 }

```

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.

```

1423     \bool_if:NT \l__nm_parallelize_diags_bool
1424     {
1425         \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).

```

1426         \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.

```

1427     {
1428         \dim_set:Nn \l__nm_delta_x_one_dim
1429         { \g__nm_x_final_dim - \g__nm_x_initial_dim }
1430         \dim_set:Nn \l__nm_delta_y_one_dim
1431         { \g__nm_y_final_dim - \g__nm_y_initial_dim }
1432     }

```

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`.

```

1433     {
1434         \dim_gset:Nn \g__nm_y_final_dim
1435         {
1436             \g__nm_y_initial_dim +
1437             ( \g__nm_x_final_dim - \g__nm_x_initial_dim ) *
1438             \dim_ratio:nn \l__nm_delta_y_one_dim \l__nm_delta_x_one_dim
1439         }
1440     }
1441 }

```

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

```

1442         \__nm_draw_tikz_line:
1443     }
1444 }
1445 }

```

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

```

1446 \cs_new_protected:Nn \__nm_draw_Iddots:nn
1447 {
1448     \cs_if_free:cT { __nm _ dotted _ #1 - #2 }
1449     {
1450         \bool_set_false:N \l__nm_impossible_line_bool
1451         \__nm_find_extremities_of_line:nnnn { #1 } { #2 } 1 { -1 }
1452         \bool_if:NF \l__nm_impossible_line_bool
1453         {
1454             \__nm_retrieve_coords:xx
1455             {
1456                 \bool_if:NTF \l__nm_initial_open_bool
1457                 { - medium.north~east }
1458                 { .south~west }
1459             }
1460             {
1461                 \bool_if:NTF \l__nm_final_open_bool
1462                 { - medium.south~west }
1463                 { .north~east }
1464             }
1465             \bool_if:NT \l__nm_parallelize_diags_bool
1466             {
1467                 \int_incr:N \l__nm_iddots_int
1468                 \int_compare:nNnTF \l__nm_iddots_int = \c_one_int
1469                 {
1470                     \dim_set:Nn \l__nm_delta_x_two_dim
1471                     { \g__nm_x_final_dim - \g__nm_x_initial_dim }
1472                     \dim_set:Nn \l__nm_delta_y_two_dim
1473                     { \g__nm_y_final_dim - \g__nm_y_initial_dim }
1474                 }
1475                 {
1476                     \dim_gset:Nn \g__nm_y_final_dim
1477                     {
1478                         \g__nm_y_initial_dim +
1479                         ( \g__nm_x_final_dim - \g__nm_x_initial_dim ) *
1480                         \dim_ratio:nn \l__nm_delta_y_two_dim \l__nm_delta_x_two_dim
1481                     }
1482                 }
1483             }
1484             \__nm_draw_tikz_line:
1485         }
1486     }
1487 }

```

16.10 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`.

```

1488 \cs_new_protected:Nn \__nm_draw_tikz_line:
1489 {

```

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.

```

1490 \dim_zero_new:N \l__nm_l_dim
1491 \dim_set:Nn \l__nm_l_dim
1492 {
1493   \fp_to_dim:n
1494   {
1495     sqrt
1496     (
1497       ( \dim_use:N \g__nm_x_final_dim
1498         - \dim_use:N \g__nm_x_initial_dim
1499       ) ^ 2
1500       +
1501       ( \dim_use:N \g__nm_y_final_dim
1502         - \dim_use:N \g__nm_y_initial_dim
1503       ) ^ 2
1504     )
1505   }
1506 }

```

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

```

1507 \dim_compare:nNf \l__nm_l_dim = \c_zero_dim

```

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

```

1508 {
1509   \bool_if:NTF \l__nm_initial_open_bool
1510   {
1511     \bool_if:NTF \l__nm_final_open_bool
1512     {
1513       \int_set:Nn \l_tmpa_int
1514       { \dim_ratio:nn \l__nm_l_dim \l__nm_inter_dots_dim }
1515     }
1516     {
1517       \int_set:Nn \l_tmpa_int
1518       { \dim_ratio:nn { \l__nm_l_dim - 0.3 em } \l__nm_inter_dots_dim }
1519     }
1520   }
1521   {
1522     \bool_if:NTF \l__nm_final_open_bool
1523     {
1524       \int_set:Nn \l_tmpa_int
1525       { \dim_ratio:nn { \l__nm_l_dim - 0.3 em } \l__nm_inter_dots_dim }
1526     }
1527     {
1528       \int_set:Nn \l_tmpa_int
1529       { \dim_ratio:nn { \l__nm_l_dim - 0.6 em } \l__nm_inter_dots_dim }
1530     }
1531   }

```

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

```

1532 \dim_set:Nn \l_tmpa_dim
1533 {
1534   ( \g__nm_x_final_dim - \g__nm_x_initial_dim ) *
1535   \dim_ratio:nn \l__nm_inter_dots_dim \l__nm_l_dim
1536 }
1537 \dim_set:Nn \l_tmpb_dim
1538 {
1539   ( \g__nm_y_final_dim - \g__nm_y_initial_dim ) *
1540   \dim_ratio:nn \l__nm_inter_dots_dim \l__nm_l_dim
1541 }

```

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`.

```

1542 \int_set:Nn \l_tmpb_int
1543 {
1544   \bool_if:NTF \l_nm_initial_open_bool
1545   { \bool_if:NTF \l_nm_final_open_bool 1 0 }
1546   { \bool_if:NTF \l_nm_final_open_bool 2 1 }
1547 }

```

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.

```

1548 \dim_gadd:Nn \g__nm_x_initial_dim
1549 {
1550   ( \g__nm_x_final_dim - \g__nm_x_initial_dim ) *
1551   \dim_ratio:nn
1552   { \l_nm_l_dim - \l_nm_inter_dots_dim * \l_tmpa_int }
1553   { \l_nm_l_dim * 2 }
1554   * \l_tmpb_int
1555 }

```

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

```

1556 \dim_gadd:Nn \g__nm_y_initial_dim
1557 {
1558   ( \g__nm_y_final_dim - \g__nm_y_initial_dim ) *
1559   \dim_ratio:nn
1560   { \l_nm_l_dim - \l_nm_inter_dots_dim * \l_tmpa_int }
1561   { \l_nm_l_dim * 2 } *
1562   \l_tmpb_int
1563 }
1564 \begin { tikzpicture } [ overlay ]
1565   \int_step_inline:nnnn 0 1 \l_tmpa_int
1566   {
1567     \pgfpathcircle
1568     { \pgfpoint { \g__nm_x_initial_dim } { \g__nm_y_initial_dim } }
1569     { \l_nm_radius_dim }
1570     \pgfusepath { fill }
1571     \dim_gadd:Nn \g__nm_x_initial_dim \l_tmpa_dim
1572     \dim_gadd:Nn \g__nm_y_initial_dim \l_tmpb_dim
1573   }
1574 \end { tikzpicture }
1575 }
1576 }

```

16.11 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).

```

1577 \cs_set_eq:NN \_nm_ldots \ldots
1578 \cs_set_eq:NN \_nm_cdots \cdots
1579 \cs_set_eq:NN \_nm_vdots \vdots
1580 \cs_set_eq:NN \_nm_ddots \ddots
1581 \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).

```

1582 \cs_new_protected:Nn \_nm_add_to_empty_cells:
1583 {
1584   \cs_gset:cpx
1585   { \_nm _ empty _ \int_use:N \g__nm_row_int - \int_use:N \g__nm_col_int }
1586   { \int_use:N \g__nm_env_int }
1587 }

```

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.

```

1588 \NewDocumentCommand \__nm_Ldots { s }
1589 {
1590   \bool_if:nF { #1 } { \__nm_instruction_of_type:n { Ldots } }
1591   \bool_if:NF \l__nm_nullify_dots_bool { \phantom \__nm_ldots }
1592   \__nm_add_to_empty_cells:
1593 }

1594 \NewDocumentCommand \__nm_Cdots { s }
1595 {
1596   \bool_if:nF { #1 } { \__nm_instruction_of_type:n { Cdots } }
1597   \bool_if:NF \l__nm_nullify_dots_bool { \phantom \__nm_cdots }
1598   \__nm_add_to_empty_cells:
1599 }

1600 \NewDocumentCommand \__nm_Vdots { s }
1601 {
1602   \bool_if:nF { #1 } { \__nm_instruction_of_type:n { Vdots } }
1603   \bool_if:NF \l__nm_nullify_dots_bool { \phantom \__nm_vdots }
1604   \__nm_add_to_empty_cells:
1605 }

1606 \NewDocumentCommand \__nm_Ddots { s }
1607 {
1608   \bool_if:nF { #1 } { \__nm_instruction_of_type:n { Ddots } }
1609   \bool_if:NF \l__nm_nullify_dots_bool { \phantom \__nm_ddots }
1610   \__nm_add_to_empty_cells:
1611 }

1612 \NewDocumentCommand \__nm_Iddots { s }
1613 {
1614   \bool_if:nF { #1 } { \__nm_instruction_of_type:n { Iddots } }
1615   \bool_if:NF \l__nm_nullify_dots_bool { \phantom \__nm_iddots }
1616   \__nm_add_to_empty_cells:
1617 }

```

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

```

1618 \cs_new_protected:Nn \__nm_Hspace:
1619 {
1620   \__nm_add_to_empty_cells:
1621   \hspace
1622 }

```

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

```

1623 \cs_set_eq:NN \__nm_old_multicolumn \multicolumn
1624 \cs_new:Npn \__nm_multicolumn:nnn #1 #2 #3
1625 {
1626   \__nm_old_multicolumn { #1 } { #2 } { #3 }
1627   \int_compare:nNnT #1 > 1
1628   {
1629     \seq_gput_left:Nx \g__nm_multicolumn_cells_seq
1630     { \int_eval:n \g__nm_row_int - \int_use:N \g__nm_col_int }
1631     \seq_gput_left:Nn \g__nm_multicolumn_sizes_seq { #1 }
1632   }
1633   \int_gadd:Nn \g__nm_col_int { #1 - 1 }
1634 }

```

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`.

```

1635 \cs_new:Npn \__nm_Hdotsfor:
1636 {
1637   \multicolumn { 1 } { C } { }
1638   \__nm_Hdotsfor_i
1639 }

```

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:`).

```

1640 \bool_if:NTF \c_nm_draft_bool
1641 {
1642   \NewDocumentCommand \__nm_Hdotsfor_i { 0 { } m }
1643   { \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { C } { } } }
1644 }
1645 {
1646   \NewDocumentCommand \__nm_Hdotsfor_i { 0 { } m }
1647   {
1648     \tl_gput_right:Nx \g__nm_Hdotsfor_lines_tl
1649     {
1650       \__nm_draw_Hdotsfor:nnn
1651       { \int_use:N \g__nm_row_int }
1652       { \int_use:N \g__nm_col_int }
1653       { #2 }
1654     }
1655     \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { C } { } }
1656   }
1657 }

```

```

1658 \cs_new_protected:Nn \__nm_draw_Hdotsfor:nnn
1659 {
1660   \bool_set_false:N \l_nm_initial_open_bool
1661   \bool_set_false:N \l_nm_final_open_bool

```

For the row, it's easy.

```

1662 \int_set:Nn \l_nm_initial_i_int { #1 }
1663 \int_set:Nn \l_nm_final_i_int { #1 }

```

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

```

1664 \int_compare:nNnTF #2 = 1
1665 {
1666   \int_set:Nn \l_nm_initial_j_int 1
1667   \bool_set_true:N \l_nm_initial_open_bool
1668 }
1669 {
1670   \int_set:Nn \l_tmpa_int { #2 - 1 }
1671   \__nm_if_not_empty_cell:nnTF \l_nm_initial_i_int \l_tmpa_int
1672   { \int_set:Nn \l_nm_initial_j_int { #2 - 1 } }
1673   {
1674     \int_set:Nn \l_nm_initial_j_int {#2}
1675     \bool_set_true:N \l_nm_initial_open_bool
1676   }
1677 }
1678 \int_compare:nNnTF { #2 + #3 - 1 } = \g__nm_col_int
1679 {
1680   \int_set:Nn \l_nm_final_j_int { #2 + #3 - 1 }
1681   \bool_set_true:N \l_nm_final_open_bool
1682 }
1683 {

```

```

1684 \int_set:Nn \l_tmpa_int { #2 + #3 }
1685 \__nm_if_not_empty_cell:nnTF \l__nm_final_i_int \l_tmpa_int
1686 { \int_set:Nn \l__nm_final_j_int { #2 + #3 } }
1687 {
1688   \int_set:Nn \l__nm_final_j_int { #2 + #3 - 1 }
1689   \bool_set_true:N \l__nm_final_open_bool
1690 }
1691 }
1692 \bool_if:nTF { \l__nm_initial_open_bool || \l__nm_final_open_bool }
1693   \__nm_create_extra_nodes:
1694   \__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).

```

1695 \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
1696 { \cs_set:cpn { __nm _ dotted _ #1 - ##1 } { } }
1697 }

```

16.12 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.

```

1698 \cs_new_protected:Nn \__nm_line:nn
1699 {
1700   \dim_zero_new:N \g__nm_x_initial_dim
1701   \dim_zero_new:N \g__nm_y_initial_dim
1702   \dim_zero_new:N \g__nm_x_final_dim
1703   \dim_zero_new:N \g__nm_y_final_dim
1704   \bool_set_false:N \l__nm_initial_open_bool
1705   \bool_set_false:N \l__nm_final_open_bool
1706   \bool_if:nTF
1707   {
1708     \cs_if_exist_p:c { pgf@sh@ns@nm - \int_use:N \g__nm_env_int - #1 }
1709     &&
1710     \cs_if_exist_p:c { pgf@sh@ns@nm - \int_use:N \g__nm_env_int - #2 }
1711   }
1712   {
1713     \begin { tikzpicture }
1714       \path~( #1 )~--~( #2 )~node[at~start]~( i )~{ }~node[at~end]~( f )~{ } ;
1715       \tikz@parse@node \pgfutil@firstofone ( i )
1716       \dim_gset:Nn \g__nm_x_initial_dim \pgf@x
1717       \dim_gset:Nn \g__nm_y_initial_dim \pgf@y
1718       \tikz@parse@node \pgfutil@firstofone ( f )
1719       \dim_gset:Nn \g__nm_x_final_dim \pgf@x
1720       \dim_gset:Nn \g__nm_y_final_dim \pgf@y
1721     \end { tikzpicture }
1722     \__nm_draw_tikz_line:
1723   }
1724   { \__nm_error:nnn { unknown-cell-for-line-in-code-after } { #1 } { #2 } }
1725 }

```

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).

16.13 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 position³¹ as the line created by `\hline` (or `\hdashline` of `arydshln`). That's why we use a `\noalign` to insert a box with a `\dotfill`.

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.

```
1726 \bool_if:NTF \c_nm_draft_bool
1727 { \cs_set_eq:NN \_nm_dotfill: \prg_do_nothing: }
1728 {
1729   \cs_set:Npn \_nm_dotfill:
1730   {
```

If the option `small` is used, we change the space between 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.

```
1731   \bool_if:NT \l_nm_small_bool
1732   { \dim_set:Nn \l_nm_inter_dots_dim { 0.25 em } }
1733   \cleaders
1734   \hbox_to_wd:nn
1735   { \l_nm_inter_dots_dim }
1736   {
1737     \c_math_toggle_token
1738     \bool_if:NT \l_nm_small_bool \scriptstyle
1739     \hss . \hss
1740     \c_math_toggle_token
1741   }
1742   \hfill
1743 }
1744 }
```

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

```
1745 \cs_new:Npn \_nm_hdottedline:
1746 {
1747   \noalign
1748   {
1749     \bool_gset_true:N \g_nm_extra_nodes_bool
1750     \cs_if_exist:cTF { __nm_width_ \int_use:N \g_nm_env_int }
1751     { \dim_set_eq:Nc \l_tmpa_dim { __nm_width_ \int_use:N \g_nm_env_int } }
1752     { \dim_set:Nn \l_tmpa_dim { 5 mm } }
1753     \hbox_overlap_right:n
1754     {
1755       \bool_if:nT
1756       {
1757         \l_nm_NiceArray_bool
1758         &&
1759         ! \l_nm_exterior_arraycolsep_bool
1760         &&
1761         \int_compare_p:nNn \l_nm_first_col_int > \c_zero_int
1762       }
1763       { \skip_horizontal:n { - \arraycolsep } }
1764       \hbox_to_wd:nn
1765       {
1766         \l_tmpa_dim + 2 \arraycolsep
1767         - \l_nm_left_margin_dim - \l_nm_right_margin_dim
1768       }
1769       \_nm_dotfill:
1770     }
1771   }
1772 }
```

³¹In 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`.


```

1773 \cs_new_protected:Nn \__nm_vdottedline:n
1774 {

```

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

```

1775 \int_compare:nNnTF #1 = \c_zero_int
1776 { \__nm_error:n { Use-of-~:~in-first-position } }
1777 {
1778   \__nm_create_extra_nodes:
1779   \bool_if:NF \c__nm_draft_bool
1780   {
1781     \dim_zero_new:N \g__nm_x_initial_dim
1782     \dim_zero_new:N \g__nm_y_initial_dim
1783     \dim_zero_new:N \g__nm_x_final_dim
1784     \dim_zero_new:N \g__nm_y_final_dim
1785     \bool_set_true:N \l__nm_initial_open_bool
1786     \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”.

```

1787 \begin { tikzpicture } [ remember-picture ]
1788   \tikz@parse@node\pgfutil@firstofone
1789   ( 1 - #1 - large .north-east )
1790   \dim_gset:Nn \g__nm_x_initial_dim \pgf@x
1791   \dim_gset:Nn \g__nm_y_initial_dim \pgf@y
1792   \tikz@parse@node\pgfutil@firstofone
1793   ( \int_use:N \g__nm_row_int - #1 - large .south-east )
1794   \dim_gset:Nn \g__nm_x_final_dim \pgf@x
1795   \dim_gset:Nn \g__nm_y_final_dim \pgf@y
1796 \end { tikzpicture }

```

However, if the previous column was constructed with a letter w, we use the w-nodes to change the x-value of the nodes.

```

1797 \cs_if_exist:cT
1798 { pgf@sh@ns@nm -\int_use:N \g__nm_env_int - 1 - #1 - w }
1799 {
1800   \begin { tikzpicture } [ remember-picture ]
1801     \tikz@parse@node\pgfutil@firstofone
1802     ( 1 - #1 - w .north-east )
1803     \dim_gset:Nn \g__nm_x_initial_dim \pgf@x
1804     \tikz@parse@node\pgfutil@firstofone
1805     ( \int_use:N \g__nm_row_int - #1 - w .south-east )
1806     \dim_gset:Nn \g__nm_x_final_dim \pgf@x
1807   \end { tikzpicture }
1808   \dim_gadd:Nn \g__nm_x_initial_dim \arraycolsep
1809   \dim_gadd:Nn \g__nm_x_final_dim \arraycolsep
1810 }

```

However, if a node of column exists (if the array has been constructed with a fixed width of column), we use it.

```

1811 \cs_if_exist:cT
1812 { pgf@sh@ns@nm -\int_use:N \g__nm_env_int - col - #1 }
1813 {
1814   \begin { tikzpicture } [ remember-picture ]
1815     \tikz@parse@node\pgfutil@firstofone
1816     ( col - #1 )
1817     \dim_gset:Nn \g__nm_x_initial_dim \pgf@x
1818     \dim_gset:Nn \g__nm_x_final_dim \pgf@x
1819   \end { tikzpicture }
1820 }
1821 \__nm_draw_tikz_line:
1822 }
1823 }
1824 }

```

16.14 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).

```
1825 \cs_new_protected:Npn \__nm_vline:
1826 {
1827   \int_compare:nNnTF \l__nm_first_col_int = \c_zero_int
1828   {
1829     \int_compare:nNnTF \g__nm_col_int = \c_zero_int
1830     {
1831       \int_compare:nNnTF \l__nm_first_row_int = \c_zero_int
1832       {
1833         \int_compare:nNnF \g__nm_row_int = \c_zero_int
1834         {
1835           \int_compare:nNnF \g__nm_row_int = \l__nm_last_row_int
1836           \__nm_vline_i:
1837         }
1838       }
1839       {
1840         \int_compare:nNnF \g__nm_row_int = \c_zero_int
1841         {
1842           \int_compare:nNnF \g__nm_row_int = \l__nm_last_row_int
1843           \__nm_vline_i:
1844         }
1845       }
1846     }
1847     {
1848       \int_compare:nNnF \g__nm_row_int = \c_zero_int
1849       {
1850         \int_compare:nNnF \g__nm_row_int = \l__nm_last_row_int
1851         \__nm_vline_i:
1852       }
1853     }
1854   }
1855   {
1856     \int_compare:nNnTF \g__nm_col_int = \c_zero_int
1857     {
1858       \int_compare:nNnF \g__nm_row_int = { -1 }
1859       {
1860         \int_compare:nNnF \g__nm_row_int = { \l__nm_last_row_int - 1 }
1861         \__nm_vline_i:
1862       }
1863     }
1864     {
1865       \int_compare:nNnF \g__nm_row_int = \c_zero_int
1866       {
1867         \int_compare:nNnF \g__nm_row_int = \l__nm_last_row_int
1868         \__nm_vline_i:
1869       }
1870     }
1871   }
1872 }
```

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

```
1873 \cs_set_eq:NN \_nm_vline_i: \vline
```

16.15 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.

```
1874 \bool_new:N \l__nm_block_auto_columns_width_bool
```

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

```
1875 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
1876 {
1877   auto-columns-width .code:n =
1878   {
1879     \bool_set_true:N \l__nm_block_auto_columns_width_bool
1880     \dim_gzero_new:N \g__nm_max_cell_width_dim
1881     \bool_set_true:N \l__nm_auto_columns_width_bool
1882   }
1883 }

1884 \NewDocumentEnvironment { NiceMatrixBlock } { ! 0 { } }
1885 {
1886   \int_gincr:N \g__nm_NiceMatrixBlock_int
1887   \dim_zero:N \l__nm_columns_width_dim
1888   \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
1889   \bool_if:NT \l__nm_block_auto_columns_width_bool
1890   {
1891     \cs_if_exist:cT { __nm_max_cell_width _ \int_use:N \g__nm_NiceMatrixBlock_int }
1892     {
1893       \dim_set:Nx \l__nm_columns_width_dim
1894       { \use:c { __nm_max_cell_width _ \int_use:N \g__nm_NiceMatrixBlock_int } }
1895     }
1896   }
1897 }
```

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`).

```
1898 {
1899   \bool_if:NT \l__nm_block_auto_columns_width_bool
1900   {
1901     \iow_now:Nn \@mainaux \ExplSyntaxOn
1902     \iow_now:Nx \@mainaux
1903     {
1904       \cs_gset:cpn
1905       { __nm _ max _ cell _ width _ \int_use:N \g__nm_NiceMatrixBlock_int }
1906       { \dim_use:N \g__nm_max_cell_width_dim }
1907     }
1908     \iow_now:Nn \@mainaux \ExplSyntaxOff
1909   }
1910 }
```

16.16 The extra nodes

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

```
1911 \cs_generate_variant:Nn \dim_min:nn { v n }
1912 \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`.

```
1913 \cs_new_protected:Nn \__nm_create_extra_nodes:
1914 {
1915   \begin { tikzpicture } [ remember-picture , overlay ]
1916     \int_step_variable:nnNn \l__nm_first_row_int \g__nm_row_total_int \__nm_i:
1917     {
1918       \dim_zero_new:c { l__nm_row\___nm_i: _min_dim }
1919       \dim_set_eq:cN { l__nm_row\___nm_i: _min_dim } \c_max_dim
1920       \dim_zero_new:c { l__nm_row\___nm_i: _max_dim }
1921       \dim_set:cn { l__nm_row\___nm_i: _max_dim } { - \c_max_dim }
1922     }
1923     \int_step_variable:nnNn \l__nm_first_col_int \g__nm_col_total_int \__nm_j:
1924     {
1925       \dim_zero_new:c { l__nm_column\___nm_j: _min_dim }
1926       \dim_set_eq:cN { l__nm_column\___nm_j: _min_dim } \c_max_dim
1927       \dim_zero_new:c { l__nm_column\___nm_j: _max_dim }
1928       \dim_set:cn { l__nm_column\___nm_j: _max_dim } { - \c_max_dim }
1929     }
1930 }
```

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

```
1930 \int_step_variable:nnNn \l__nm_first_row_int \g__nm_row_total_int \__nm_i:
1931 {
1932   \int_step_variable:nnNn
1933   \l__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.

```
1934 { \cs_if_exist:cT
1935   { 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`.

```
1936 {
1937   \tikz@parse@node \pgfutil@firstofone
1938   ( nm - \int_use:N \g__nm_env_int
1939     - \__nm_i: - \__nm_j: .south-west )
1940   \dim_set:cn { l__nm_row\___nm_i: _min_dim }
1941   { \dim_min:vn { l__nm_row\___nm_i: _min_dim } \pgf@y }
1942   \seq_if_in:NxF \g__nm_multicolumn_cells_seq { \__nm_i: - \__nm_j: }
1943   {
1944     \dim_set:cn { l__nm_column\___nm_j: _min_dim }
1945     { \dim_min:vn { l__nm_column\___nm_j: _min_dim } \pgf@x }
1946   }
```

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`.

```

1947         \tikz@parse@node \pgfutil@firstofone
1948         ( nm - \int_use:N \g__nm_env_int - \__nm_i: - \__nm_j: .north-east )
1949         \dim_set:cn { l__nm_row _ \__nm_i: _ max_dim }
1950         { \dim_max:vn { l__nm_row _ \__nm_i: _ max_dim } \pgf@y }
1951         \seq_if_in:NxF \g__nm_multicolumn_cells_seq { \__nm_i: - \__nm_j: }
1952         {
1953             \dim_set:cn { l__nm_column _ \__nm_j: _ max_dim }
1954             { \dim_max:vn { l__nm_column _ \__nm_j: _ max_dim } \pgf@x }
1955         }
1956     }
1957 }
1958 }

```

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`).

```

1959         \tikzset { name-suffix = -medium }
1960         \__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.

```

1961         \int_set:Nn \l__nm_first_row_int 1
1962         \int_set:Nn \l__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`.

```

1963         \int_step_variable:nNn { \g__nm_row_int - 1 } \__nm_i:
1964         {
1965             \dim_set:cn { l__nm_row _ \__nm_i: _ min _ dim }
1966             {
1967                 (
1968                     \dim_use:c { l__nm_row _ \__nm_i: _ min _ dim } +
1969                     \dim_use:c { l__nm_row _ \int_eval:n { \__nm_i: + 1 } _ max _ dim }
1970                 )
1971                 / 2
1972             }
1973             \dim_set_eq:cc { l__nm_row _ \int_eval:n { \__nm_i: + 1 } _ max _ dim }
1974             { l__nm_row _ \__nm_i: _ min_dim }
1975         }
1976         \int_step_variable:nNn { \g__nm_col_int - 1 } \__nm_j:
1977         {
1978             \dim_set:cn { l__nm_column _ \__nm_j: _ max _ dim }
1979             {
1980                 (
1981                     \dim_use:c
1982                     { l__nm_column _ \__nm_j: _ max _ dim } +
1983                     \dim_use:c
1984                     { l__nm_column _ \int_eval:n { \__nm_j: + 1 } _ min _ dim }
1985                 )
1986                 / 2
1987             }
1988             \dim_set_eq:cc { l__nm_column _ \int_eval:n { \__nm_j: + 1 } _ min _ dim }
1989             { l__nm_column _ \__nm_j: _ max _ dim }
1990         }
1991         \dim_sub:cn
1992         { l__nm_column _ 1 _ min _ dim }
1993         \l__nm_left_margin_dim
1994         \dim_add:cn
1995         { l__nm_column _ \int_use:N \g__nm_col_int _ max _ dim }
1996         \l__nm_right_margin_dim

```

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

```

1997      \tikzset { name~suffix = -large }
1998      \_nm_create_nodes:
1999      \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.

```

2000      \cs_set:Npn \_nm_create_extra_nodes: { }

```

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

```

2001      \begin { tikzpicture } [ remember~picture , overlay ]
2002      \tikz@parse@node \pgfutil@firstofone
2003      ( nm - \int_use:N \g__nm_env_int - 1 - 1 - large .north-west )
2004      \dim_gset:Nn \g_tmpa_dim \pgf@x
2005      \tikz@parse@node \pgfutil@firstofone
2006      ( nm - \int_use:N \g__nm_env_int - 1 -
2007        \int_use:N \g__nm_col_int - large .north-east )
2008      \dim_gset:Nn \g_tmpb_dim \pgf@x
2009      \end { tikzpicture }
2010      \iow_now:Nn \@mainaux \ExplSyntaxOn
2011      \iow_now:Nx \@mainaux
2012      {
2013        \cs_gset:cpn { __nm_width_ \int_use:N \g__nm_env_int }
2014        { \dim_eval:n { \g_tmpb_dim - \g_tmpa_dim } }
2015      }
2016      \iow_now:Nn \@mainaux \ExplSyntaxOff
2017  }

```

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.

```

2018 \cs_new_protected:Nn \_nm_create_nodes:
2019 {
2020   \int_step_variable:nnNn \l__nm_first_row_int \g__nm_row_total_int \_nm_i:
2021   {
2022     \int_step_variable:nnNn \l__nm_first_col_int \g__nm_col_total_int \_nm_j:

```

We create two ponctual 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.

```

2023   {
2024     \coordinate ( __nm~south~west )
2025     at ( \dim_use:c { l__nm_column_ \_nm_j: _min_dim } ,
2026         \dim_use:c { l__nm_row_ \_nm_i: _min_dim } ) ;
2027     \coordinate ( __nm~north~east )
2028     at ( \dim_use:c { l__nm_column_ \_nm_j: _max_dim } ,
2029         \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`.

```

2030      \node
2031      [
2032        node~contents = { } ,
2033        fit = ( __nm~south~west ) ( __nm~north~east ) ,
2034        inner~sep = \c_zero_dim ,
2035        name = nm - \int_use:N \g__nm_env_int - \_nm_i: - \_nm_j: ,
2036        alias =
2037        \str_if_empty:NF \l__nm_name_str

```

```

2038         { \l__nm_name_str - \__nm_i: - \__nm_j: }
2039     ]
2040     ;
2041 }
2042 }

```

Now, we create the nodes for the cells of the `\multicolumn`. We recall that we have stored in `\g_@@_multicolumn_cells_seq` the list of the cells where a `\multicolumn{n}{...}{...}` with $n > 1$ was issued and in `\g_@@_multicolumn_sizes_seq` the correspondent values of n .

```

2043     \seq_mapthread_function:NNN
2044     \g__nm_multicolumn_cells_seq
2045     \g__nm_multicolumn_sizes_seq
2046     \__nm_node_for_multicolumn:nn
2047 }
2048 \cs_new_protected:Npn \__nm_extract_coords: #1 - #2 \q_stop
2049 {
2050     \cs_set:Npn \__nm_i: { #1 }
2051     \cs_set:Npn \__nm_j: { #2 }
2052 }

```

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

```

2053 \cs_new_protected:Nn \__nm_node_for_multicolumn:nn
2054 {
2055     \__nm_extract_coords: #1 \q_stop
2056     \coordinate ( __nm~south~west ) at
2057     (
2058         \dim_use:c { l__nm_column _ \__nm_j: _ min _ dim } ,
2059         \dim_use:c { l__nm_row _ \__nm_i: _ min _ dim }
2060     ) ;
2061     \coordinate ( __nm~north~east ) at
2062     (
2063         \dim_use:c { l__nm_column _ \int_eval:n { \__nm_j: + #2 - 1 } _ max _ dim } ,
2064         \dim_use:c { l__nm_row _ \__nm_i: _ max _ dim }
2065     ) ;
2066     \node
2067     [
2068         node~contents = { } ,
2069         fit = ( __nm~south~west ) ( __nm~north~east ) ,
2070         inner~sep = \c_zero_dim ,
2071         name = nm - \int_use:N \g__nm_env_int - \__nm_i: - \__nm_j: ,
2072         alias =
2073             \str_if_empty:NF \l__nm_name_str { \l__nm_name_str - \__nm_i: - \__nm_j: }
2074     ]
2075     ;
2076 }

```

16.17 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 optionnal 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`.

```

2077 \NewExpandableDocumentCommand \__nm_Block: { m D < > { } m }
2078 {
2079     \__nm_Block_i #1 \q_stop { #2 } { #3 }
2080 }

```

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.

```

2081 \cs_new:Npn \_nm_Block_i #1-#2 \q_stop
2082 {
2083   \_nm_Block_ii:nnnn { #1 } { #2 }
2084 }

```

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).

```

2085 \cs_new:Npn \_nm_Block_ii:nnnn #1 #2 #3 #4
2086 {

```

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.

```

2087   \int_compare:nNnTF { #1 } = 1
2088   {
2089     \multicolumn { #2 } { C } { \hbox:n { #3 $#4$ } }
2090     \_nm_gobble_ampersands:n { #2 - 1 }
2091   }
2092   { \_nm_Block_iii:nnnn { #1 } { #2 } { #3 } { #4 } }
2093 }

```

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

```

2094 \cs_new_protected:Npn \_nm_Block_iii:nnnn #1 #2 #3 #4
2095 {
2096   \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).

```

2097   \tl_gput_left:Nx \g__nm_code_after_tl
2098   {
2099     \_nm_Block_iv:nnnnn
2100     { \int_use:N \g__nm_row_int }
2101     { \int_use:N \g__nm_col_int }
2102     { \int_eval:n { \g__nm_row_int + #1 - 1 } }
2103     { \int_eval:n { \g__nm_col_int + #2 - 1 } }
2104     \exp_not:n { { #3 $ #4 $ } }
2105   }
2106 }

```

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.

```

2107 \group_begin:
2108   \char_set_catcode_letter:N \&
2109   \cs_new:Npn \_nm_gobble_ampersands:n #1
2110   {
2111     \int_compare:nNnT { #1 } > 0
2112     {
2113       \peek_charcode_remove_ignore_spaces:NT &
2114       { \_nm_gobble_ampersands:n { #1 - 1 } }
2115     }
2116   }
2117 \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.

```

2118 \cs_new_protected:Npn \_nm_Block_iv:nnnnn #1 #2 #3 #4 #5
2119 {

```



```

2120 \bool_if:nTF
2121 {
2122     \int_compare_p:nNn { #3 } > \g__nm_row_int
2123     || \int_compare_p:nNn { #4 } > \g__nm_col_int
2124 }
2125 { \msg_error:nnnn { nicematrix } { Block~too~large } { #1 } { #2 } }
2126 {
2127     \begin{tikzpicture}
2128     \node
2129     [
2130         fit = ( #1 - #2 - medium . north-west )
2131             ( #3 - #4 - medium . south-east ) ,
2132         inner~sep = 0 pt ,
2133     ]

```

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

```

2134         (#1-#2) { } ;
2135     \node at (#1-#2.center) { #5 } ;
2136 \end{tikzpicture}
2137 }
2138 }

```

16.18 How to draw the dotted lines transparently

```

2139 \cs_set_protected:Npn \__nm_renew_matrix:
2140 {
2141     \RenewDocumentEnvironment { pmatrix } { } {
2142         { \pNiceMatrix }
2143         { \endpNiceMatrix }
2144     }
2145     \RenewDocumentEnvironment { vmatrix } { } {
2146         { \vNiceMatrix }
2147         { \endvNiceMatrix }
2148     }
2149     \RenewDocumentEnvironment { Vmatrix } { } {
2150         { \VNiceMatrix }
2151         { \endVNiceMatrix }
2152     }
2153     \RenewDocumentEnvironment { bmatrix } { } {
2154         { \bNiceMatrix }
2155         { \endbNiceMatrix }
2156     }
2157     \RenewDocumentEnvironment { Bmatrix } { } {
2158         { \BNiceMatrix }
2159         { \endBNiceMatrix }
2160     }
2161 }

```

16.19 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.

```

2157 \keys_define:nn { NiceMatrix / Package }
2158 {
2159     renew-dots .bool_set:N = \l__nm_renew_dots_bool ,
2160     renew-dots .value_forbidden:n = true ,
2161     renew-matrix .code:n = \__nm_renew_matrix: ,
2162     renew-matrix .value_forbidden:n = true ,
2163     transparent .meta:n = { renew-dots , renew-matrix } ,
2164     transparent .value_forbidden:n = true ,
2165 }
2166 \ProcessKeysOptions { NiceMatrix / Package }

```

16.20 Error messages of the package

```

2167 \_nm_msg_new:nn { unknown~cell~for~line~in~code~after }
2168 {
2169     Your~command~\token_to_str:N\line\{#1\}\{#2\}~in~the~'code~after'~
2170     can't~be~executed~because~a~Tikz~node~doesn't~exist.\\
2171     If~you~go~on~this~command~will~be~ignored.
2172 }
2173 \_nm_msg_new:nn { last~col~non~empty~for~NiceArray }
2174 {
2175     In~the~environment~\{g__nm_type_env_str\},~you~must~use~the~option~
2176     'last~col'~without~value~(the~number~of~columns~is~known~by~the~
2177     preamble~of~the~environment).\\
2178     However,~you~can~go~on~for~this~time~
2179     (the~value~'\l_keys_value_tl'~will~be~ignored).
2180 }
2181 \_nm_msg_new:nn { last~col~empty~for~NiceMatrix }
2182 {
2183     In~the~environment~\{g__nm_type_env_str\}~you~can't~use~the~option~
2184     'last~col'~without~value.~You~must~give~the~number~of~that~last~column.\\
2185     If~you~go~on~this~option~will~be~ignored.
2186 }
2187 \_nm_msg_new:nn { Block~too~large }
2188 {
2189     You~try~to~draw~a~block~in~the~cell~#1~#2~of~your~matrix~but~the~matrix~is~
2190     too~small~for~that~block.\\
2191     If~you~go~on,~this~command~line~will~be~ignored.
2192 }
2193 \_nm_msg_new:nn { Impossible~line }
2194 {
2195     A~dotted~line~can't~be~drawn~because~you~have~not~put~
2196     all~the~ampersands~required~on~the~row~#1.\\
2197     If~you~go~on,~this~dotted~line~will~be~ignored.
2198 }
2199 \_nm_msg_new:nn { Wrong~last~row }
2200 {
2201     You~have~used~'last~row=\int_use:N \l__nm_last_row_int'~but~your~environment~
2202     \{g__nm_type_env_str\}~seems~to~have~\int_use:N \g__nm_row_int\
2203     rows.~If~you~go~on,~the~value~of~\int_use:N \g__nm_row_int\
2204     will~be~used~for~last~row.~You~can~avoid~this~problem~by~using~'last~row'~
2205     without~value~(more~compilations~might~be~necessary).
2206 }
2207 \_nm_msg_new:nn { Draft~mode }
2208 { The~compilation~is~in~draft~mode:~the~dotted~lines~won't~be~drawn. }
2209 \_nm_msg_new:nn { Yet~in~env }
2210 {
2211     Environments~\{NiceArray\}~(or~\{NiceMatrix\},~etc.)~can't~be~nested.\\
2212     This~error~is~fatal.
2213 }
2214 \_nm_msg_new:nn { Outside~math~mode }
2215 {
2216     The~environment~\{g__nm_type_env_str\}~can~be~used~only~in~math~mode~
2217     (and~not~in~\token_to_str:N \vcenter).\\
2218     This~error~is~fatal.
2219 }
2220 \_nm_msg_new:nn { Option~Transparent~suppressed }
2221 {
2222     The~option~'Transparent'~has~been~renamed~'transparent'.\\
2223     However,~you~can~go~on~for~this~time.
2224 }

```

```

2225 \_nm_msg_new:nn { Option~RenewMatrix~suppressed }
2226 {
2227     The~option~'RenewMatrix'~has~been~renamed~'renew-matrix'.\\
2228     However,~you~can~go~on~for~this~time.
2229 }
2230 \_nm_msg_new:nn { Bad~value~for~letter~for~dotted~lines }
2231 {
2232     The~value~of~key~'\tl_use:N\l_keys_key_tl'~must~be~of~length~1.\\
2233     If~you~go~on,~it~will~be~ignored.
2234 }
2235 \_nm_msg_new:nnn { Unknown~key~for~NiceMatrixOptions }
2236 {
2237     The~key~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~command~
2238     \token_to_str:N \NiceMatrixOptions. \\
2239     If~you~go~on,~it~will~be~ignored. \\
2240     For~a~list~of~the~available~keys,~type~H~<return>.
2241 }
2242 {
2243     The~available~options~are~(in~alphabetic~order):~
2244     allow-duplicate-names,~
2245     code-for-first-col,~
2246     code-for-first-row,~
2247     code-for-last-col,~
2248     code-for-last-row,~
2249     exterior-arraycolsep,~
2250     hlines,~
2251     left-margin,~
2252     letter-for-dotted-lines,~
2253     nullify-dots,~
2254     parallelize-diags,~
2255     renew-dots,~
2256     renew-matrix,~
2257     right-margin,~
2258     small~
2259     and~transparent
2260 }
2261 \_nm_msg_new:nnn { Unknown~option~for~NiceArray }
2262 {
2263     The~option~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~environment~
2264     \{NiceArray\}. \\
2265     If~you~go~on,~it~will~be~ignored. \\
2266     For~a~list~of~the~available~options,~type~H~<return>.
2267 }
2268 {
2269     The~available~options~are~(in~alphabetic~order):~
2270     b,~
2271     c,~
2272     code-after,~
2273     code-for-first-col,~
2274     code-for-first-row,~
2275     code-for-last-col,~
2276     code-for-last-row,~
2277     columns-width,~
2278     create-extra-nodes,~
2279     extra-left-margin,~
2280     extra-right-margin,~
2281     first-col,~
2282     first-row,~
2283     hlines,~
2284     last-col,~
2285     last-row,~
2286     left-margin,~
2287     name,~

```

```

2288 nullify-dots,~
2289 parallelize-diags,~
2290 renew-dots,~
2291 right-margin,~
2292 small~
2293 and~t.
2294 }

```

This error message is used for the set of keys NiceMatrix/NiceMatrix and NiceMatrix/pNiceArray (but not by NiceMatrix/NiceArray (because, for this set of keys, there is also the options t, c and b).

```

2295 \_nm_msg_new:nnn { Unknown~option~for~NiceMatrix }
2296 {
2297   The~option~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~environment~
2298   \{g__nm_type_env_str\}. \\
2299   If~you~go~on,~it~will~be~ignored. \\
2300   For~a~list~of~the~available~options,~type~H~<return>.
2301 }
2302 {
2303   The~available~options~are~(in~alphabetic~order):~
2304   code~after,~
2305   code~for~first~col,~
2306   code~for~first~row,~
2307   code~for~last~col,~
2308   code~for~last~row,~
2309   columns~width,~
2310   create~extra~nodes,~
2311   extra~left~margin,~
2312   extra~right~margin,~
2313   first~col,~
2314   first~row,~
2315   hlines,~
2316   last~col,~
2317   last~row,~
2318   left~margin,~
2319   name,~
2320   nullify-dots,~
2321   parallelize-diags,~
2322   renew-dots,~
2323   right~margin~
2324   and~small.
2325 }
2326 \_nm_msg_new:nnn { Duplicate~name }
2327 {
2328   The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
2329   the~same~environment~name~twice.~You~can~go~on,~but,~
2330   maybe,~you~will~have~incorrect~results~especially~
2331   if~you~use~'columns-width=auto'.~If~you~use~nicematrix~inside~some~
2332   environments~of~amsmath,~this~error~may~be~an~artefact.~In~this~case,~
2333   use~the~option~'allow-duplicate-names'.\\
2334   For~a~list~of~the~names~already~used,~type~H~<return>. \\
2335 }
2336 {
2337   The~names~already~defined~in~this~document~are:~
2338   \seq_use:Nnnn \g__nm_names_seq { ,~ } { ,~ } { ~and~ }.
2339 }
2340 \_nm_msg_new:nn { Option~auto~for~columns~width }
2341 {
2342   You~can't~give~the~value~'auto'~to~the~option~'columns-width'~here.~
2343   If~you~go~on,~the~option~will~be~ignored.
2344 }
2345 \_nm_msg_new:nn { Zero~row }
2346 {

```

```

2347   There-is-a-problem.-Maybe-your-environment-\{g_nm_type_env_str\}-is-empty.-
2348   Maybe-you-have-used-l,~c~and~r~instead-of~L,~C~and~R~in~the-preamble~
2349   of~your~environment.  \\\
2350   If-you-go-on,~the~result~may~be~incorrect.
2351 }
2352 \_nm_msg_new:n { Use~of~::~in~first~position }
2353 {
2354   You-can't-use-the-column-specifier~'\l_nm_letter_for_dotted_lines_str'~in~the~
2355   first~position~of~the~preamble~of~the~environment~\{g_nm_type_env_str\}.  \\\
2356   If-you-go-on,~this~dotted~line~will~be~ignored.
2357 }
2358 % \end{macrocode}
2359 %
2360 %
2361 % \subsection{Obsolete environments}
2362 %
2363 % \begin{macrocode}
2364 \NewDocumentEnvironment { pNiceArrayC } { }
2365 {
2366   \int_set:Nn \l_nm_last_col_int \c_zero_dim
2367   \pNiceArray
2368 }
2369 { \endpNiceArray }
2370 \NewDocumentEnvironment { bNiceArrayC } { }
2371 {
2372   \int_set:Nn \l_nm_last_col_int \c_zero_dim
2373   \bNiceArray
2374 }
2375 { \endbNiceArray }
2376 \NewDocumentEnvironment { BNiceArrayC } { }
2377 {
2378   \int_set:Nn \l_nm_last_col_int \c_zero_dim
2379   \BNiceArray
2380 }
2381 { \endBNiceArray }
2382 \NewDocumentEnvironment { vNiceArrayC } { }
2383 {
2384   \int_set:Nn \l_nm_last_col_int \c_zero_dim
2385   \vNiceArray
2386 }
2387 { \endvNiceArray }
2388 \NewDocumentEnvironment { VNiceArrayC } { }
2389 {
2390   \int_set:Nn \l_nm_last_col_int \c_zero_dim
2391   \VNiceArray
2392 }
2393 { \endVNiceArray }
2394 \NewDocumentEnvironment { pNiceArrayRC } { }
2395 {
2396   \int_set:Nn \l_nm_last_col_int \c_zero_dim
2397   \int_set:Nn \l_nm_first_row_int \c_zero_int
2398   \pNiceArray
2399 }
2400 { \endpNiceArray }
2401 \NewDocumentEnvironment { bNiceArrayRC } { }
2402 {
2403   \int_set:Nn \l_nm_last_col_int \c_zero_dim
2404   \int_set:Nn \l_nm_first_row_int \c_zero_int
2405   \bNiceArray
2406 }
2407 { \endbNiceArray }

```

```

2408 \NewDocumentEnvironment { BNiceArrayRC } { }
2409 {
2410   \int_set:Nn \l__nm_last_col_int \c_zero_dim
2411   \int_set:Nn \l__nm_first_row_int \c_zero_int
2412   \BNiceArray
2413 }
2414 { \endBNiceArray }

2415 \NewDocumentEnvironment { vNiceArrayRC } { }
2416 {
2417   \bool_set_true:N \l__nm_last_col_bool
2418   \int_set:Nn \l__nm_first_row_int \c_zero_int
2419   \vNiceArray
2420 }
2421 { \endvNiceArray }

2422 \NewDocumentEnvironment { VNiceArrayRC } { }
2423 {
2424   \int_set:Nn \l__nm_last_col_int \c_zero_dim
2425   \int_set:Nn \l__nm_first_row_int \c_zero_int
2426   \VNiceArray
2427 }
2428 { \endVNiceArray }

2429 \NewDocumentEnvironment { NiceArrayCwithDelims } { }
2430 {
2431   \int_set:Nn \l__nm_last_col_int \c_zero_dim
2432   \NiceArrayWithDelims
2433 }
2434 { \endNiceArrayWithDelims }

2435 \NewDocumentEnvironment { NiceArrayRCwithDelims } { }
2436 {
2437   \int_set:Nn \l__nm_last_col_int \c_zero_dim
2438   \int_set:Nn \l__nm_first_row_int \c_zero_int
2439   \NiceArrayWithDelims
2440 }
2441 { \endNiceArrayWithDelims }

```

17 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 StackExchange³², Tikz externalization is now deactivated in the environments of the extension `nicematrix`.³³

Changes between version 2.1 and 2.1.2

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

Changes between version 2.1.2 and 2.1.3

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

$$\begin{pmatrix} & C_j & \\ 0 & \vdots & 0 \\ & a & \cdots \\ 0 & & 0 \end{pmatrix} L_i$$

Changes between version 2.1.3 and 2.1.4

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

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

Changes between version 2.1.4 and 2.1.5

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

Option `allow-duplicate-names`.

³²cf. tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package

³³Before 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 2.1.5 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).

Changes between version 3.1 and 3.2 (and 3.2a)

Option `small`.

Changes between version 3.2 and 3.3

The options `first-row`, `last-row`, `first-col` and `last-col` are now available in the environments `{NiceMatrix}`, `{pNiceMatrix}`, `{bNiceMatrix}`, etc.

The option `columns-width=auto` doesn’t need any more a second compilation.

The options `renew-dots`, `renew-matrix` and `transparent` are now available as package options (as said in the documentation).

The previous version of `nicematrix` was incompatible with a recent version of `expl3` (released 2019/09/30). This version is compatible.

Changes between version 3.3 and 3.4

Following a discussion on TeX StackExchange³⁴, optimization of Tikz externalization is disabled in the environments of `nicematrix` when the class `standalone` or the package `standalone` is used.

³⁴cf. tex.stackexchange.com/questions/510841/nicematrix-and-tikz-external-optimize

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>	2108
<code>\%</code>	2170, 2177, 2184, 2190, 2196, 2211, 2217, 2222, 2227, 2232, 2238, 2239, 2264, 2265, 2298, 2299, 2333, 2334, 2349, 2355
<code>\{</code>	937, 2169, 2175, 2183, 2202, 2211, 2216, 2264, 2298, 2347, 2355
<code>\}</code>	937, 2169, 2175, 2183, 2202, 2211, 2216, 2264, 2298, 2347, 2355
<code>\ </code>	953
<code>\</code>	2202, 2203
A	
<code>\array</code>	432
<code>\arraycolsep</code>	161, 163, 165, 457, 600, 601, 669, 673, 731, 769, 771, 785, 844, 872, 1763, 1766, 1808, 1809
<code>\arrayrulewidth</code>	446, 447
<code>\arraystretch</code>	456
<code>\AtBeginDocument</code>	62, 88
B	
<code>\begin</code>	962, 1010, 1244, 1564, 1713, 1787, 1800, 1814, 1915, 2001, 2127, 2363
<code>\bgroup</code>	348
<code>\Block</code>	504
<code>\BNiceArray</code>	2379, 2412
<code>\bNiceArray</code>	2373, 2405
<code>\BNiceMatrix</code>	2154
<code>\bNiceMatrix</code>	2151
bool commands:	
<code>\bool_do_until:Nn</code>	1115, 1175
<code>\bool_gset_eq:NN</code>	459
<code>\bool_gset_false:N</code>	546
<code>\bool_gset_true:N</code>	534, 851, 1749, 2096
<code>\bool_if:NTF</code>	34, 96, 286, 408, 424, 440, 454, 462, 505, 544, 560, 569, 572, 598, 629, 631, 638, 640, 700, 734, 782, 796, 857, 1007, 1047, 1065, 1074, 1081, 1139, 1167, 1199, 1228, 1236, 1266, 1273, 1288, 1302, 1304, 1316, 1320, 1335, 1349, 1351, 1363, 1367, 1372, 1380, 1385, 1391, 1395, 1410, 1414, 1419, 1423, 1452, 1456, 1461, 1465, 1509, 1511, 1522, 1544, 1545, 1546, 1591, 1597, 1603, 1609, 1615, 1640, 1726, 1731, 1738, 1779, 1889, 1899
<code>\bool_if:nTF</code>	651, 662, 677, 1043, 1235, 1590, 1596, 1602, 1608, 1614, 1692, 1706, 1755, 2120
<code>\bool_new:N</code>	11, 27, 52, 53, 60, 61, 83, 86, 87, 128, 129, 131, 132, 133, 135, 136, 1874
<code>\bool_set:Nn</code>	1376, 1389
<code>\bool_set_false:N</code>	983, 997, 1079, 1080, 1114, 1119, 1174, 1179, 1264, 1314, 1361, 1408, 1450, 1660, 1661, 1704, 1705
<code>\bool_set_true:N</code>	12, 29, 32, 66, 91, 130, 175, 224, 561, 578, 910, 1005, 1123, 1129, 1135, 1143, 1147, 1171, 1183, 1189, 1195, 1203, 1208, 1232, 1667, 1675, 1681, 1689, 1785, 1786, 1879, 1881, 2417
<code>\l_tmpa_bool</code>	983, 997, 1005, 1007, 1376, 1395
<code>\l_tmpb_bool</code>	1389, 1391
box commands:	
<code>\box_clear_new:N</code>	624
<code>\box_dp:N</code> ..	306, 318, 368, 473, 479, 483, 779
<code>\box_ht:N</code> ..	308, 313, 316, 475, 477, 481, 778
<code>\box_move_down:n</code>	369, 749
<code>\box_move_up:n</code>	383, 740
<code>\box_set_dp:Nn</code>	779
<code>\box_set_ht:Nn</code>	778
<code>\box_use:N</code>	349, 367, 383, 836, 904
<code>\box_use_drop:N</code>	757, 770, 780
<code>\box_wd:N</code>	325, 373, 613, 621, 809, 870
<code>\l_tmpa_box</code>	284, 306, 308, 313, 316, 318, 325, 349, 358, 367, 606, 613, 614, 621, 760, 778, 779, 780, 794, 809, 836, 855, 870, 904
<code>\l_tmpb_box</code>	366, 368, 373, 383
C	
<code>\Cdots</code>	496
<code>\cdots</code>	508, 1578
Cdots internal commands:	
<code>__nm_Cdots</code>	496, 508, 1594
cdots internal commands:	
<code>__nm_cdots</code>	1578, 1597
char commands:	
<code>\char_set_catcode_letter:N</code>	2108
<code>\cleaders</code>	1733
<code>\coordinate</code>	377, 382, 661, 686, 2024, 2027, 2056, 2061
<code>\crrcr</code>	657
cs commands:	
<code>\cs_generate_variant:Nn</code>	352, 1259, 1911, 1912
<code>\cs_gset:Npn</code>	1052, 1059, 1904, 2013
<code>\cs_gset:Npx</code>	1584
<code>\cs_gset_eq:NN</code>	109, 466
<code>\cs_if_exist:NTF</code>	562, 565, 581, 588, 984, 998, 1040, 1275, 1290, 1322, 1337, 1750, 1797, 1811, 1891, 1934
<code>\cs_if_exist_p:N</code>	1708, 1710
<code>\cs_if_free:NTF</code>	988, 1159, 1220, 1262, 1312, 1359, 1406, 1448
<code>\cs_new:Npn</code>	435, 1624, 1635, 1745, 2081, 2085, 2109
<code>\cs_new_protected:Nn</code>	274, 294, 320, 353, 1031, 1037, 1107, 1238, 1260, 1269, 1310, 1357, 1404, 1446, 1488, 1582, 1618, 1658, 1698, 1773, 1913, 2018, 2053
<code>\cs_new_protected:Npn</code>	18, 19, 20, 21, 22, 23, 24, 25, 54, 112, 301, 411, 422, 437, 452, 956, 1825, 2048, 2094, 2118
<code>\cs_set:Npn</code>	429, 456, 460, 484, 1109, 1149, 1210, 1696, 1729, 2000, 2050, 2051

<code>\cs_set_eq:NN</code>	100, 426, 427, 428, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 507, 508, 509, 510, 511, 512, 513, 522, 523, 524, 526, 1101, 1577, 1578, 1579, 1580, 1581, 1623, 1727, 1873
<code>\cs_set_protected:Npn</code>	67, 94, 409, 571, 2139
D	
<code>\Ddots</code>	498
<code>\ddots</code>	510, 1580
Ddots internal commands:	
<code>__nm_Ddots</code>	498, 510, 1606
ddots internal commands:	
<code>__nm_ddots</code>	1580, 1609
<code>\DeclareOption</code>	12, 13
dim commands:	
<code>\dim_abs:n</code>	1024
<code>\dim_add:Nn</code>	1994
<code>\dim_compare:nNnTF</code>	1023, 1507
<code>\dim_compare_p:nNn</code>	654, 1377, 1390
<code>\dim_eval:n</code>	2014
<code>\dim_gadd:Nn</code>	1306, 1307, 1548, 1556, 1571, 1572, 1808, 1809
<code>\dim_gset:Nn</code>	305, 307, 312, 315, 317, 324, 473, 475, 477, 479, 481, 483, 600, 601, 613, 621, 668, 672, 805, 866, 1019, 1021, 1249, 1250, 1255, 1256, 1393, 1434, 1476, 1716, 1717, 1719, 1720, 1790, 1791, 1794, 1795, 1803, 1806, 1817, 1818, 2004, 2008
<code>\dim_gset_eq:NN</code>	297, 1303, 1305, 1350, 1352, 1398
<code>\dim_gzero:N</code>	298, 299
<code>\dim_gzero_new:N</code>	472, 474, 476, 478, 480, 482, 570, 596, 597, 1240, 1241, 1242, 1243, 1880
<code>\dim_max:nn</code>	306, 308, 313, 316, 318, 325, 807, 868, 1395, 1912, 1950, 1954
<code>\dim_min:nn</code>	1395, 1911, 1941, 1945
<code>\dim_new:N</code>	50, 71, 73, 137, 138, 139, 140, 141, 142
<code>\dim_ratio:nn</code>	1438, 1480, 1514, 1518, 1525, 1529, 1535, 1540, 1551, 1559
<code>\dim_set:Nn</code>	72, 74, 176, 229, 352, 368, 457, 711, 721, 1083, 1084, 1428, 1430, 1470, 1472, 1491, 1532, 1537, 1732, 1752, 1893, 1921, 1928, 1940, 1944, 1949, 1953, 1965, 1978
<code>\dim_set_eq:NN</code>	605, 616, 1751, 1919, 1926, 1973, 1988
<code>\dim_sub:Nn</code>	1991
<code>\dim_use:N</code>	1497, 1498, 1501, 1502, 1906, 1968, 1969, 1981, 1983, 2025, 2026, 2028, 2029, 2058, 2059, 2063, 2064
<code>\dim_zero:N</code>	718, 728, 1887
<code>\dim_zero_new:N</code>	1069, 1070, 1071, 1072, 1490, 1700, 1701, 1702, 1703, 1781, 1782, 1783, 1784, 1918, 1920, 1925, 1927
<code>\c_max_dim</code>	1919, 1921, 1926, 1928
<code>\g_tmpa_dim</code>	668, 672, 675, 684, 1019, 1024, 2004, 2014
<code>\l_tmpa_dim</code>	368, 369, 383, 711, 718, 741, 766, 778, 1532, 1571, 1751, 1752, 1766
<code>\g_tmpb_dim</code>	1021, 1024, 2008, 2014
<code>\l_tmpb_dim</code>	721, 728, 751, 773, 779, 1537, 1572
<code>\c_zero_dim</code>	330, 331, 401, 605, 616, 654, 816, 817, 884, 885, 1507, 2034, 2070, 2366, 2372, 2378, 2384, 2390, 2396, 2403, 2410, 2424, 2431, 2437
<code>\dots</code>	512
E	
<code>\egroup</code>	350
else commands:	
<code>\else:</code>	56
<code>\end</code>	973, 1022, 1257, 1574, 1721, 1796, 1807, 1819, 1999, 2009, 2136, 2358
<code>\endarray</code>	693
<code>\endBNiceArray</code>	2381, 2414
<code>\endbNiceArray</code>	2375, 2407
<code>\endBNiceMatrix</code>	2155
<code>\endbNiceMatrix</code>	2152
<code>\endNiceArrayWithDelims</code>	915, 923, 931, 939, 947, 955, 2434, 2441
<code>\endpNiceArray</code>	2369, 2400
<code>\endpNiceMatrix</code>	2143
<code>\endVNiceArray</code>	2393, 2428
<code>\endvNiceArray</code>	2387, 2421
<code>\endVNiceMatrix</code>	2149
<code>\endvNiceMatrix</code>	2146
<code>\everycr</code>	470, 486
exp commands:	
<code>\exp_after:wN</code>	116
<code>\exp_args:NV</code>	529, 648
<code>\exp_not:n</code>	2104
<code>\ExplSyntaxOff</code>	1063, 1908, 2016
<code>\ExplSyntaxOn</code>	1049, 1901, 2010
F	
fi commands:	
<code>\fi:</code>	58
fp commands:	
<code>\fp_to_dim:n</code>	1493
G	
group commands:	
<code>\group_begin:</code>	98, 604, 1039, 2107
<code>\group_end:</code>	103, 622, 1104, 2117
H	
<code>\halign</code>	488, 490
hbox commands:	
<code>\hbox:n</code>	41, 43, 45, 379, 767, 2089
<code>\hbox_overlap_left:n</code>	811
<code>\hbox_overlap_right:n</code>	873, 1753
<code>\hbox_set:Nn</code>	366, 606, 614, 760
<code>\hbox_set:Nw</code>	284, 358, 644, 794, 855
<code>\hbox_set_end:</code>	323, 364, 697, 803, 864
<code>\hbox_to_wd:nn</code>	373, 1734, 1764
<code>\Hdotsfor</code>	502
<code>\hdotsfor</code>	513
<code>\hdottedline</code>	500
<code>\hfil</code>	375, 526
<code>\hfill</code>	1742
<code>\hrule</code>	446
<code>\Hspace</code>	501
<code>\hspace</code>	1621
<code>\hss</code>	526, 1739

I

`\ialign` 460, 484

`\iddots` 499

`\iddots` 36, 511, 1581

lddots internal commands:

`_nm_iddots` 499, 511, 1612

lddots internal commands:

`_nm_iddots` 1581, 1615

if commands:

`\if_mode_math:` 56

`\ifstandalone` 565

int commands:

`\int_add:Nn` 1117, 1118, 1201, 1202

`\int_compare:nNnTF` 277, 279, 287, 290, 303,
 310, 442, 444, 535, 576, 626, 635, 658, 698,
 702, 709, 719, 729, 736, 745, 966, 1001,
 1033, 1045, 1120, 1122, 1126, 1128, 1132,
 1134, 1180, 1182, 1186, 1188, 1192, 1194,
 1426, 1468, 1627, 1664, 1678, 1775, 1827,
 1829, 1831, 1833, 1835, 1840, 1842, 1848,
 1850, 1856, 1858, 1860, 1865, 1867, 2087, 2111

`\int_compare:nTF` 235

`\int_compare_p:nNn` 1761, 2122, 2123

`\int_eval:n` 968,
 1630, 1969, 1973, 1984, 1988, 2063, 2102, 2103

`\int_gadd:Nn` 1633

`\int_gdecr:N` 1043

`\int_gincr:N` ... 276, 296, 568, 683, 852, 1886

`\int_gset:Nn` 282, 518, 545, 676, 853

`\int_gset_eq:NN` 537, 705, 1042, 1044

`\int_gsub:Nn` 1046

`\int_gzero:N` 439

`\int_gzero_new:N` 517, 519, 520, 521, 543

`\int_incr:N` 1425, 1467

`\int_max:nn` 283, 854

`\int_new:N` 48, 49, 77, 79, 81, 84

`\int_set:Nn` 78,
 80, 82, 85, 250, 583, 590, 1110, 1111,
 1112, 1113, 1513, 1517, 1524, 1528, 1542,
 1662, 1663, 1666, 1670, 1672, 1674, 1680,
 1684, 1686, 1688, 1961, 1962, 2366, 2372,
 2378, 2384, 2390, 2396, 2397, 2403, 2404,
 2410, 2411, 2418, 2424, 2425, 2431, 2437, 2438

`\int_step_inline:nnn` 1695

`\int_step_inline:nnnn` 1565

`\int_step_variable:nNn` 1963, 1976

`\int_step_variable:nnNn`
 1916, 1923, 1930, 1932, 2020, 2022

`\int_sub:Nn` 1141, 1142, 1177, 1178

`\int_use:N` 337, 338,
 339, 344, 345, 391, 392, 393, 398, 399, 417,
 418, 539, 581, 584, 661, 688, 689, 825, 826,
 832, 893, 894, 895, 900, 901, 985, 991, 992,
 993, 999, 1002, 1014, 1015, 1016, 1052,
 1053, 1060, 1098, 1152, 1153, 1162, 1163,
 1164, 1170, 1213, 1214, 1223, 1224, 1225,
 1231, 1246, 1247, 1248, 1252, 1253, 1254,
 1278, 1279, 1280, 1293, 1294, 1295, 1325,
 1326, 1327, 1340, 1341, 1342, 1585, 1586,
 1630, 1651, 1652, 1708, 1710, 1750, 1751,
 1793, 1798, 1805, 1812, 1891, 1894, 1905,
 1935, 1938, 1948, 1995, 2003, 2006, 2007,
 2013, 2035, 2071, 2100, 2101, 2201, 2202, 2203

`\int_zero:N` 187, 188, 260, 265, 268, 269

`\int_zero_new:N`
 1067, 1068, 1075, 1076, 1077, 1078

`\c_one_int` 235, 277, 279,
 310, 1046, 1265, 1315, 1362, 1409, 1426, 1468

`\g_tmpa_int` 676, 683, 689

`\l_tmpa_int` 1513, 1517, 1524,
 1528, 1552, 1560, 1565, 1670, 1671, 1684, 1685

`\l_tmpb_int` 1542, 1554, 1562

`\c_zero_int` 287, 303, 626, 709,
 729, 736, 1033, 1265, 1315, 1362, 1761,
 1775, 1827, 1829, 1831, 1833, 1840, 1848,
 1856, 1865, 2397, 2404, 2411, 2418, 2425, 2438

iow commands:

`\iow_now:Nn` 1049, 1050,
 1057, 1063, 1901, 1902, 1908, 2010, 2011, 2016

K

`\kern` 45

keys commands:

`\keys_define:nn`
 143, 171, 192, 214, 246, 253, 263, 1875, 2157

`\l_keys_key_tl` 2232, 2237, 2263, 2297

`\keys_set:nn` 245, 573, 574, 961, 1888

`\l_keys_value_tl` 2179, 2328

L

`\Ldots` 495

`\ldots` 507, 1577

ldots internal commands:

`_nm_ldots` 495, 507, 512, 1588

ldots internal commands:

`_nm_ldots` 1577, 1591

`\left` 609, 618, 763

`\line` 1101, 2169

`\lineskip` 714, 724

M

`\makebox` 367

math commands:

`\c_math_toggle_token`
 285, 322, 608, 610, 617, 619, 647,
 694, 762, 776, 795, 802, 856, 863, 1737, 1740

`\mathinner` 38

`\mkern` 40, 42, 44, 45

msg commands:

`\msg_error:nn` 18

`\msg_error:nnn` 19, 1169, 1230

`\msg_error:nnnn` 20, 2125

`\msg_fatal:nn` 21, 22

`\msg_new:nnn` 23

`\msg_new:nnnn` 24

`\msg_redirect_name:nnn` 26

`\msg_warning:nn` 35

`\multicolumn` .. 503, 1623, 1637, 1643, 1655, 2089

`\myfiledate` 8

`\myfileversion` 9

N

`\newcolumntype` 355, 492, 493, 494, 529

`\NewDocumentCommand`
 244, 1588, 1594, 1600, 1606, 1612, 1642, 1646

`\NewDocumentEnvironment`
 554, 908, 916, 924, 932, 940,

948, 958, 1884, 2364, 2370, 2376, 2382, 2388, 2394, 2401, 2408, 2415, 2422, 2429, 2435	
\NewExpandableDocumentCommand	2077
\NiceArrayWithDelims	
... 913, 921, 929, 937, 945, 953, 2432, 2439	
\NiceMatrixOptions	244, 2238
nm internal commands:	
_nm_actualization_for_first_and_- last_row:	301, 326, 804, 865
_nm_actually_draw_Ldots:	1266, 1269, 1694
_nm_adapt_S_column:	94, 109, 558
_nm_add_to_empty_cells:	
... 1582, 1592, 1598, 1604, 1610, 1616, 1620	
_nm_after_array:	787, 1031
_nm_after_array_i:	1034, 1037
_nm_array:	422, 648
\l_nm_auto_columns_width_bool	
... 133, 175, 653, 664, 1881	
_nm_begin_of_row:	280, 294, 793
_nm_Block:	504, 2077
\l_nm_block_auto_columns_width_bool	
... 569, 653, 665, 1874, 1879, 1889, 1899	
_nm_Block_i	2079, 2081
_nm_Block_ii:nnnn	2083, 2085
_nm_Block_iii:nnnn	2092, 2094
_nm_Block_iv:nnnn	2099, 2118
\g_nm_Cdots_lines_tl	547, 1090
_nm_Cell:	119, 274, 359, 492, 493, 494
\g_nm_code_after_tl	
... 76, 185, 538, 1102, 1103, 2097	
\l_nm_code_for_first_col_tl	145, 797
\l_nm_code_for_first_row_tl	149, 288
\l_nm_code_for_last_col_tl	147, 858
\l_nm_code_for_last_row_tl	151, 291
\g_nm_col_int	276, 277, 283, 339, 345, 393, 399, 418, 439, 520, 535, 537, 539, 852, 854, 895, 901, 1042, 1043, 1132, 1192, 1585, 1630, 1633, 1652, 1678, 1829, 1856, 1976, 1995, 2007, 2101, 2103, 2123
\g_nm_col_total_int	282, 283, 521, 678, 679, 853, 854, 1042, 1923, 1933, 2022
\c_nm_colortbl_loaded_bool ...	61, 66, 462
\l_nm_columns_width_dim	
... 50, 176, 229, 654, 673, 1887, 1893	
_nm_create_extra_nodes:	
... 1074, 1235, 1236, 1693, 1778, 1913, 2000	
_nm_create_nodes:	1960, 1998, 2018
\l_nm_ddots_int	1067, 1425, 1426
\g_nm_Ddots_lines_tl	550, 1088
_nm_define_env:n	
... 956, 975, 976, 977, 978, 979, 980	
\l_nm_delta_x_one_dim ...	1069, 1428, 1438
\l_nm_delta_x_two_dim ...	1071, 1470, 1480
\l_nm_delta_y_one_dim ...	1070, 1430, 1438
\l_nm_delta_y_two_dim ...	1072, 1472, 1480
_nm_dotfill:	1727, 1729, 1769
\g_nm_dp_ante_last_row_dim	
... 297, 478, 479, 723, 725, 752	
\g_nm_dp_last_row_dim	
... 297, 298, 317, 318, 482, 483, 725, 752	
\g_nm_dp_row_zero_dim	305, 306, 472, 473, 713
\c_nm_draft_bool	
... 11, 12, 34, 408, 1640, 1726, 1779	
_nm_draw_Cdots:nn	1310
_nm_draw_Ddots:nn	1404
_nm_draw_Hdotsfor:nnn	1650, 1658
_nm_draw_Iddots:nn	1446
_nm_draw_Ldots:nn	1260
_nm_draw_tikz_line:	
... 1308, 1353, 1400, 1442, 1484, 1488, 1722, 1821	
_nm_draw_Vdots:nn	1357
_nm_end_Cell: .	121, 320, 363, 492, 493, 494
\g_nm_env_int	48, 337, 391, 568, 581, 584, 661, 688, 825, 893, 991, 1004, 1014, 1052, 1098, 1162, 1223, 1246, 1252, 1278, 1293, 1325, 1340, 1586, 1708, 1710, 1750, 1751, 1798, 1812, 1935, 1938, 1948, 2003, 2006, 2013, 2035, 2071
_nm_error:n .	18, 218, 222, 228, 237, 240, 249, 251, 259, 261, 267, 272, 704, 1035, 1776
_nm_error:nn	19, 181
_nm_error:nnn	20, 1724
_nm_everycr:	435, 467, 470
_nm_everycr_i:	436, 437
\l_nm_exterior_arraycolsep_bool	
... 128, 225, 631, 640, 1759	
\l_nm_extra_left_margin_dim	
... 141, 166, 646, 841	
\g_nm_extra_nodes_bool	
... 136, 459, 534, 1074, 1749, 2096	
\l_nm_extra_nodes_bool	135, 159, 459
\l_nm_extra_right_margin_dim	
... 142, 167, 696, 879	
_nm_extract_coords:	2048, 2055
_nm_fatal:n	21, 57, 560
_nm_fatal:nn	22
\l_nm_final_i_int	
... 1077, 1112, 1117, 1120, 1141, 1146, 1152, 1163, 1170, 1253, 1294, 1341, 1663, 1685	
\l_nm_final_j_int	
... 1078, 1113, 1118, 1126, 1132, 1142, 1146, 1153, 1164, 1254, 1295, 1342, 1680, 1686, 1688	
\l_nm_final_open_bool	
... 1080, 1119, 1123, 1129, 1135, 1139, 1167, 1236, 1288, 1304, 1335, 1351, 1372, 1385, 1419, 1461, 1511, 1522, 1545, 1546, 1661, 1681, 1689, 1692, 1705, 1786	
_nm_find_extremities_of_line:nnnn ..	
... 1107, 1265, 1315, 1362, 1409, 1451	
\l_nm_first_col_int	
... 79, 80, 187, 265, 279, 626, 658, 729, 1761, 1827, 1923, 1933, 1962, 2022	
\l_nm_first_row_int	77, 78, 188, 269, 518, 709, 736, 1831, 1916, 1930, 1961, 2020, 2397, 2404, 2411, 2418, 2425, 2438
_nm_gobble_ampersands:n	2090, 2109, 2114
_nm_Hdotsfor:	502, 513, 1635
_nm_Hdotsfor_i	1638, 1642, 1646
\g_nm_Hdotsfor_lines_tl ..	552, 1086, 1648
_nm_hdottedline:	500, 1745
\l_nm_hlines_bool	131, 154, 440
_nm_Hspace:	501, 1618
\g_nm_ht_last_row_dim	
... 299, 315, 316, 480, 481, 723	
\g_nm_ht_row_one_dim	
... 312, 313, 476, 477, 713, 715, 741	

<code>\g_nm_ht_row_zero_dim</code>	<code>_nm_msg_new:nnn</code> 24, 2235, 2261, 2295, 2326
..... 307, 308, 474, 475, 715, 741	<code>_nm_msg_redirect_name:nn</code> 25, 231
<code>_nm_i:</code> 1916, 1918,	<code>_nm_multicolumn:nnn</code> 503, 1624
1919, 1920, 1921, 1930, 1935, 1939, 1940,	<code>\g_nm_multicolumn_cells_seq</code>
1941, 1942, 1948, 1949, 1950, 1951, 1963, 515, 1629, 1942, 1951, 2044
1965, 1968, 1969, 1973, 1974, 2020, 2026,	<code>\g_nm_multicolumn_sizes_seq</code> 516, 1631, 2045
2029, 2035, 2038, 2050, 2059, 2064, 2071, 2073	<code>\l_nm_name_str</code> 134,
<code>\l_nm_iddots_int</code> 1068, 1467, 1468	183, 341, 343, 395, 397, 579, 588, 591, 829,
<code>\g_nm_iddots_lines_tl</code> 551, 1089	831, 897, 899, 1055, 1059, 2037, 2038, 2073
<code>_nm_if_not_empty_cell:nn</code> 981	<code>\g_nm_names_seq</code> 51, 180, 182, 2338
<code>_nm_if_not_empty_cell:nnTF</code>	<code>\l_nm_NiceArray_bool</code>
..... 1146, 1206, 1671, 1685 53, 572, 598, 629, 638, 734, 910, 1757
<code>\l_nm_impossible_line_bool</code>	<code>\g_nm_NiceMatrixBlock_int</code>
..... 60, 1171, 1232, 1264, 1266, 49, 1886, 1891, 1894, 1905
1314, 1316, 1361, 1363, 1408, 1410, 1450, 1452	<code>_nm_node_for_multicolumn:nn</code> .. 2046, 2053
<code>\l_nm_in_env_bool</code> 52, 560, 561	<code>\l_nm_nullify_dots_bool</code>
<code>\l_nm_initial_i_int</code> 132, 158, 1591, 1597, 1603, 1609, 1615
..... 1075, 1110, 1177, 1180, 1201, 1207,	<code>_nm_old_multicolumn</code> 1623, 1626
1213, 1224, 1231, 1247, 1279, 1326, 1662, 1671	<code>\l_nm_parallelize_diags_bool</code>
<code>\l_nm_initial_j_int</code> 129, 130, 155, 1065, 1423, 1465
1076, 1111, 1178, 1186, 1192, 1202, 1207,	<code>\l_nm_pos_env_str</code>
1214, 1225, 1248, 1280, 1327, 1666, 1672, 1674 126, 127, 255, 256, 257, 433, 738, 747
<code>\l_nm_initial_open_bool</code> .. 1079, 1179,	<code>_nm_pre_array:</code> 452, 595
1183, 1189, 1195, 1199, 1228, 1235, 1273,	<code>\c_nm_preamble_first_col_tl</code> 627, 789
1302, 1320, 1349, 1367, 1380, 1414, 1456,	<code>\c_nm_preamble_last_col_tl</code> 636, 847
1509, 1544, 1660, 1667, 1675, 1692, 1704, 1785	<code>\l_nm_radius_dim</code> 73, 74, 1083, 1569
<code>_nm_instruction_of_type:n</code>	<code>\l_nm_renew_dots_bool</code> . 156, 224, 505, 2159
..... 409, 411, 1590, 1596, 1602, 1608, 1614	<code>_nm_renew_matrix:</code> 216, 219, 223, 2139, 2161
<code>\l_nm_inter_dots_dim</code>	<code>_nm_renew_NC@rewrite@S:</code> 112, 544
..... 71, 72, 1084, 1514, 1518,	<code>_nm_renewcolumn:nn</code> 353, 525, 526
1525, 1529, 1535, 1540, 1552, 1560, 1732, 1735	<code>_nm_retrieve_coords:nn</code>
<code>_nm_j:</code> 1923, 1925,	1238, 1259, 1271, 1318, 1365, 1378, 1412, 1454
1926, 1927, 1928, 1933, 1935, 1939, 1942,	<code>\c_nm_revtext_bool</code> 27, 29, 32, 424
1944, 1945, 1948, 1951, 1953, 1954, 1976,	<code>\g_nm_right_delim_dim</code> .. 597, 601, 621, 877
1978, 1982, 1984, 1988, 1989, 2022, 2025,	<code>\l_nm_right_margin_dim</code>
2028, 2035, 2038, 2051, 2058, 2063, 2071, 2073 138, 162, 695, 878, 1767, 1996
<code>\l_nm_l_dim</code> 1490, 1491, 1507, 1514, 1518,	<code>\g_nm_row_int</code> 287, 290, 296, 303, 310, 338,
1525, 1529, 1535, 1540, 1552, 1553, 1560, 1561	344, 392, 398, 417, 442, 444, 517, 518, 702,
<code>\l_nm_last_col_bool</code> 2417	705, 826, 832, 894, 900, 1033, 1044, 1046,
<code>\g_nm_last_col_found_bool</code>	1120, 1585, 1630, 1651, 1793, 1805, 1833,
..... 86, 546, 677, 782, 851, 1043	1835, 1840, 1842, 1848, 1850, 1858, 1860,
<code>\l_nm_last_col_int</code> 84, 85, 250,	1865, 1867, 1963, 2100, 2102, 2122, 2202, 2203
260, 268, 635, 966, 968, 2366, 2372, 2378,	<code>\g_nm_row_total_int</code>
2384, 2390, 2396, 2403, 2410, 2424, 2431, 2437 519, 1044, 1053, 1060, 1916, 1930, 2020
<code>\l_nm_last_row_int</code> 81, 82, 189, 270,	<code>\c_nm_siunitx_loaded_bool</code> . 87, 91, 96, 544
290, 444, 576, 583, 590, 698, 702, 705, 719,	<code>\l_nm_small_bool</code>
745, 1045, 1835, 1842, 1850, 1860, 1867, 2201	... 153, 286, 454, 796, 857, 1081, 1731, 1738
<code>\l_nm_last_row_without_value_bool</code> ..	<code>\l_nm_stop_loop_bool</code>
..... 83, 578, 700, 1047	1114, 1115, 1143, 1147, 1174, 1175, 1203, 1208
<code>\g_nm_last_vdotted_col_int</code>	<code>\c_nm_table_collect_begin_tl</code> 104, 106, 119
..... 535, 537, 543, 545	<code>\c_nm_table_print_tl</code> 107, 108, 121
<code>\g_nm_ldots_lines_tl</code> 548, 1091	<code>_nm_test_if_math_mode:</code>
<code>\g_nm_left_delim_dim</code> ... 596, 600, 613, 839 54, 559, 920, 928, 936, 944, 952
<code>\l_nm_left_margin_dim</code>	<code>\l_nm_the_array_box</code> 624, 644, 757, 770
..... 137, 160, 645, 840, 1767, 1993	<code>\g_nm_type_env_str</code>
<code>\l_nm_letter_for_dotted_lines_str</code> 75, 556, 557, 911, 912, 918, 919,
..... 236, 242, 243, 528, 529, 2354	926, 927, 934, 935, 942, 943, 950, 951, 960,
<code>_nm_line:nn</code> 1101, 1698	1105, 2175, 2183, 2202, 2216, 2298, 2347, 2355
<code>\g_nm_max_cell_width_dim</code>	<code>\g_nm_Vdots_lines_tl</code> 549, 1087
..... 324, 325, 570, 669, 1880, 1906	<code>_nm_vdottedline:n</code> 539, 1773
<code>_nm_msg_new:nn</code> 23,	<code>_nm_vline:</code> 571, 1825
2167, 2173, 2181, 2187, 2193, 2199, 2207,	<code>_nm_vline_i:</code>
2209, 2214, 2220, 2225, 2230, 2340, 2345, 2352 67, 1836, 1843, 1851, 1861, 1868, 1873

<code>\g_nm_width_first_col_dim</code>	140, 732, 805, 808
<code>\g_nm_width_last_col_dim</code>	139, 784, 866, 869
<code>\g_nm_x_final_dim</code> 1242, 1255, 1377, 1390, 1396, 1398, 1429, 1437, 1471, 1479, 1497, 1534, 1550, 1702, 1719, 1783, 1794, 1806, 1809, 1818
<code>\g_nm_x_initial_dim</code> 1240, 1249, 1377, 1390, 1393, 1396, 1398, 1429, 1437, 1471, 1479, 1498, 1534, 1548, 1550, 1568, 1571, 1700, 1716, 1781, 1790, 1803, 1808, 1817
<code>\g_nm_y_final_dim</code>	... 1243, 1256, 1303, 1305, 1307, 1350, 1352, 1431, 1434, 1473, 1476, 1501, 1539, 1558, 1703, 1720, 1784, 1795
<code>\g_nm_y_initial_dim</code> 1241, 1250, 1303, 1305, 1306, 1350, 1352, 1431, 1436, 1473, 1478, 1502, 1539, 1556, 1558, 1568, 1572, 1701, 1717, 1782, 1791
<code>\noalign</code> 436, 466, 1747
<code>\node</code>	.. 334, 388, 820, 888, 2030, 2066, 2128, 2135
<code>\nulldelimiterspace</code> 605, 616
O	
<code>\omit</code> 658, 659, 682
P	
<code>\path</code> 1714
peek commands:	
<code>\peek_charcode_remove_ignore_spaces:N</code>	NTF 2113
<code>\pgfpathcircle</code> 1567
<code>\pgfpoint</code> 1568
<code>\pgfpointanchor</code> 1018, 1020
<code>\pgfusepath</code> 1570
<code>\phantom</code> 1591, 1597, 1603, 1609, 1615
<code>\pNiceArray</code> 2367, 2398
<code>\pNiceMatrix</code> 2142
prg commands:	
<code>\prg_do_nothing:</code> 100, 109, 466, 1727
<code>\prg_replicate:nn</code> 678, 679, 1643, 1655
<code>\prg_return_false:</code> 995, 1008, 1025
<code>\prg_return_true:</code> 986, 1026
<code>\prg_set_conditional:Npnn</code> 981
<code>\ProcessKeysOptions</code> 2166
<code>\ProcessOptions</code> 14
<code>\ProvideDocumentCommand</code> 36
<code>\ProvidesExplPackage</code> 6
Q	
quark commands:	
<code>\q_stop</code> 2048, 2055, 2079, 2081
R	
<code>\raise</code> 41, 43, 45
<code>\relax</code> 14, 523, 524
<code>\renewcommand</code> 114
<code>\RenewDocumentEnvironment</code> 2141, 2144, 2147, 2150, 2153
<code>\RequirePackage</code> 2, 4, 5, 15, 16, 17
<code>\right</code> 609, 618, 775
S	
<code>\scriptstyle</code> 286, 796, 857, 1738
seq commands:	
<code>\seq_gclear_new:N</code> 515, 516
<code>\seq_gput_left:Nn</code> 182, 1629, 1631
<code>\seq_if_in:NnTF</code> 180, 1942, 1951
<code>\seq_mapthread_function:NNN</code> 2043
<code>\seq_new:N</code> 51
<code>\seq_use:Nnnn</code> 2338
skip commands:	
<code>\skip_horizontal:N</code> 675, 684
<code>\skip_horizontal:n</code> 533, 645, 646, 695, 696, 731, 732, 769, 771, 784, 785, 837, 844, 872, 875, 1763
<code>\skip_vertical:n</code> 447, 766, 773
<code>\c_zero_skip</code> 471, 487
str commands:	
<code>\c_colon_str</code> 243
<code>\str_gclear:N</code> 1105
<code>\str_gset:Nn</code> 557, 912, 919, 927, 935, 943, 951, 960
<code>\str_if_empty:N</code>	NTF 341, 395, 556, 579, 829, 897, 911, 918, 926, 934, 942, 950, 1055, 2037, 2073
<code>\str_if_eq:nnTF</code> 174, 227, 738, 747
<code>\str_new:N</code> 75, 126, 134, 242
<code>\str_set:Nn</code> 127, 179, 236, 255, 256, 257
<code>\str_set_eq:NN</code> 183, 243
<code>\l_tmpa_str</code> 179, 180, 182, 183
<code>\subsection</code> 2361
T	
<code>\tabskip</code> 471, 487
TeX and L ^A T _E X commands:	
<code>\@acol</code> 428
<code>\@acoll</code> 426
<code>\@acolr</code> 427
<code>\@addtopreamble</code> 571
<code>\@array@array</code> 430
<code>\@arrayacol</code> 426, 427, 428
<code>\@arrayrule</code> 571
<code>\@arstrutbox</code> 473, 475, 477, 479, 481, 483
<code>\@halignto</code> 429
<code>\@height</code> 446
<code>\@ifclassloaded</code> 28, 31
<code>\@ifnextchar</code> 522
<code>\@ifpackageloaded</code> 64, 90
<code>\@mainaux</code> 1049, 1050, 1057, 1063, 1901, 1902, 1908, 2010, 2011, 2016
<code>\@temptokena</code> 99, 102, 116, 118
<code>\c@MaxMatrixCols</code> 967
<code>\CT@arc@</code> 67
<code>\CT@everycr</code> 464
<code>\CT@row@color</code> 466
<code>\NC@find</code> 100, 123
<code>\NC@find@W</code> 524
<code>\NC@find@w</code> 523
<code>\NC@rewrite@S</code> 101, 114
<code>\new@ifnextchar</code> 522
<code>\p@</code> 41, 43, 45
<code>\pgf@x</code> 1019, 1021, 1249, 1255, 1716, 1719, 1790, 1794, 1803, 1806, 1817, 1818, 1945, 1954, 2004, 2008
<code>\pgf@y</code> 1250, 1256, 1717, 1720, 1791, 1795, 1941, 1950
<code>\pgfutil@firstofone</code> 1245, 1251, 1715, 1718, 1788, 1792, 1801, 1804, 1815, 1937, 1947, 2002, 2005

<code>\tikz@library@external@loaded</code> ...	562, 1040	<code>\l_tmpa_tl</code>	105, 106, 625, 627, 632, 636, 641, 648, 1011, 1018, 1020
<code>\tikz@parse@node</code>	1245, 1251, 1715, 1718, 1788, 1792, 1801, 1804, 1815, 1937, 1947, 2002, 2005	token commands:	
tex commands:		<code>\token_to_str:N</code>	2169, 2217, 2238
<code>\tex_the:D</code>	118		U
<code>\tikz</code>	327, 376, 381, 387, 660, 685, 813, 881	use commands:	
<code>\tikzset</code>	564, 566, 1041, 1092, 1959, 1997	<code>\use:N</code>	416, 584, 591, 1002, 1894
tl commands:		<code>\usetikzlibrary</code>	3
<code>\tl_const:Nn</code>	789, 847		V
<code>\tl_count:n</code>	235	<code>\vbox</code>	45
<code>\tl_gclear:N</code>	1103	vbox commands:	
<code>\tl_gclear_new:N</code>	547, 548, 549, 550, 551, 552	<code>\vbox:n</code>	371
<code>\tl_gput_left:Nn</code>	2097	<code>\vcenter</code>	609, 618, 764, 2217
<code>\tl_gput_right:Nn</code>	413, 538, 1648	<code>\Vdots</code>	497
<code>\tl_gset:Nn</code>	102, 106, 108	<code>\vdots</code>	509, 1579
<code>\tl_if_empty:nTF</code>	248, 258, 266	Vdots internal commands:	
<code>\tl_item:Nn</code>	105, 106, 108	<code>_nm_Vdots</code>	497, 509, 1600
<code>\tl_new:N</code>	76, 104, 107	vdots internal commands:	
<code>\tl_put_left:Nn</code>	627, 632	<code>_nm_vdots</code>	1579, 1603
<code>\tl_put_right:Nn</code>	636, 641	<code>\vline</code>	67, 1873
<code>\tl_set:Nn</code>	105, 625, 1011	<code>\VNiceArray</code>	2391, 2426
<code>\tl_set_rescan:Nnn</code>	527	<code>\vNiceArray</code>	2385, 2419
<code>\tl_use:N</code>	2232, 2237, 2263, 2297	<code>\VNiceMatrix</code>	2148
<code>\g_tmpa_tl</code>	102, 105, 108	<code>\vNiceMatrix</code>	2145

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 exterior row and columns	7
8	The dotted lines to separate rows or columns	9
9	The width of the columns	10
10	Block matrices	11
11	The option <code>small</code>	11
12	The option <code>hlines</code>	12
13	Utilisation of the column type <code>S</code> of <code>siunitx</code>	12

14	Technical remarks	12
14.1	Intersections of dotted lines	12
14.2	Diagonal lines	13
14.3	The “empty” cells	13
14.4	The option exterior-arraycolsep	14
14.5	The class option draft	14
14.6	A technical problem with the argument of <code>\</code>	14
14.7	Obsolete environments	15
15	Examples	15
15.1	Dotted lines	15
15.2	Width of the columns	17
15.3	How to highlight cells of the matrix	18
15.4	Direct utilisation of the Tikz nodes	20
16	Implementation	22
16.1	Declaration of the package and extensions loaded	22
16.2	Technical definitions	23
16.2.1	Variables for the exterior rows and columns	25
16.2.2	The column S of siunitx	26
16.3	The options	27
16.4	Important code used by <code>{NiceArrayWithDelims}</code>	31
16.5	The environment <code>{NiceArrayWithDelims}</code>	38
16.6	The environment <code>{NiceMatrix}</code> and its variants	46
16.7	Automatic width of the cells	47
16.8	How to know whether a cell is “empty”	47
16.9	After the construction of the array	48
16.10	The actual instructions for drawing the dotted line with Tikz	58
16.11	User commands available in the new environments	60
16.12	The command <code>\line</code> accessible in code-after	63
16.13	The commands to draw dotted lines to separate columns and rows	63
16.14	The vertical rules	66
16.15	The environment <code>{NiceMatrixBlock}</code>	67
16.16	The extra nodes	68
16.17	Block matrices	71
16.18	How to draw the dotted lines transparently	73
16.19	We process the options	73
16.20	Error messages of the package	74
17	History	78
	Index	81