

The package `nicematrix`*

F. Pantigny
 fpantigny@wanadoo.fr

January 10, 2020

Abstract

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

1 Presentation

This package can be used with `xelatex`, `lualatex`, `pdflatex` but also by the classical workflow `latex-dvips-ps2pdf` (or Adobe Distiller). Two or three compilations may be necessary. This package requires and **loads** the packages `expl3`, `l3keys2e`, `xparse`, `array`, `amsmath` and `tikz`. It also loads the Tikz library `fit`. The final user only has to load the extension with `\usepackage{nicematrix}`.

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

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

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

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 3.9 of `nicematrix`, at the date of 2020/01/10.

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 & \Cdots & & & a_1 \\</code>	$\left[\begin{array}{cccc} a_1 & \cdots & \cdots & a_1 \\ \vdots & & & \\ \vdots & a_2 & \cdots & a_2 \\ \vdots & \vdots & \ddots & \\ a_1 & a_2 & & a_n \end{array} \right]$
<code>\Vdots & & \Cdots & & \\</code>	
<code>& \Vdots & \Ddots \\</code>	
<code>\\</code>	
<code>a_1 & a_2 & & & 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 & \Cdots & 0 \\</code>	$\left[\begin{array}{ccc} 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{array} \right]$
<code>\Vdots & & \Vdots \\</code>	
<code>0 & \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 & \Cdots & \Cdots & 0 \\</code>	$\left[\begin{array}{cccc} 0 & \cdots & \cdots & 0 \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{array} \right]$
<code>\Vdots & & & \Vdots \\</code>	
<code>\Vdots & & & \Vdots \\</code>	
<code>0 & \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. 15).

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.

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

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

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

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”: the “medium nodes” and the “large nodes”. The first ones are created with the option `create-medium-nodes` and the second ones with the option `create-large-nodes`.⁸

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.)

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

The names of the “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.⁹

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

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`.¹⁰

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

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. 19).

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}
```

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

⁹There is no “large nodes” created in the exterior rows and columns (for these rows and columns, cf. p. 7).

¹⁰The 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 R¹¹ 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.¹²

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.

```


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


```

7 The exterior rows and columns

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

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

¹¹The column types L, C and R are defined locally inside `{NiceArray}` with `\newcolumntype` of `array`. This definition overrides an eventual previous definition. In fact, the column types `w` and `W` are also redefined.

¹²In 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 rows of the array and writes that information in the `.aux` file for the second run. In the following example, the option `last-row` will be used without value.

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

```

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

```

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

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.¹³

If one wishes to define new specifiers for columns in order to draw vertical rules (for example thicker than the standard rules), he should consider the command `\OnlyMainNiceMatrix` described on page 13.

- 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)
```

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

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}
```

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

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

¹³The latter is not true when the extension `arydshln` is loaded besides `nicematrix`. In fact, `nicematrix` and `arydshln` are not totally compatible because `arydshln` redefines many internals of `array`. On another hand, if one really wants a vertical rule running in the first and in the last row, he should use `!\vline` instead of `|` in the preamble of the array.

```

\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)

```

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

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`.

```

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

```

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

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

It's possible to give the 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.¹⁴

```

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

```

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

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

```

$$\left(\begin{array}{cc} a & b \\ c & d \end{array} \right) = \left(\begin{array}{cc} 1 & 1245 \\ 345 & 2 \end{array} \right)$$

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`.¹⁵

¹⁴The 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).

¹⁵At this time, this is the only usage of the environment `{NiceMatrixBlock}` but it may have other usages in the future.

```

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

```

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

Several compilations may be necessary to achieve the job.

10 Block matrices

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

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

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

```

\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}

```

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

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}

```

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

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`).

```

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

```

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

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

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

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

12 The counters `iRow` and `jCol`

In the cells of the array, it's possible to use the LaTeX counters `iRow` and `jCol` which represent the number of the current row and the number of the current col¹⁶. Of course, the user must not change the value of these counters which are used internally by `nicematrix`.

```

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

```

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

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

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

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

```

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

```

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

¹⁶We recall that the first row (if it exists) has the number 0 and that the first col (if it exists) has also the number 0.

13 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>

14 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`.

```
\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}
```

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

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

15 Technical remarks

15.1 Definition of new column types

The extension `nicematrix` provides the command `\OnlyMainNiceMatrix` which is meant to be used in definitions of new column types. Its argument is evaluated if and only if we are in the main part of the array, that is to say not in an eventual exterior row.

For example, one may wish to define a new column type `?` in order to draw a thick rule of width 1 pt. The following definition will do the job:

```
\newcolumnstype{?}{!\OnlyMainNiceMatrix{\vrule width 1 pt}}
```

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

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

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

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

15.2 Intersections of dotted lines

Since the version 3.1 of `nicematrix`, the dotted lines created by `\Cdots`, `\Ldots`, `\Vdots`, etc. can't intersect.¹⁷

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

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

```


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


```

15.3 The names of the Tikz nodes created by nicematrix

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

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

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

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

15.4 Diagonal lines

By default, all the diagonal lines¹⁸ 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):

```


$$A = \begin{pmatrix} 1 & \cdots & \cdots & 1 \\ a+b & \textcolor{blue}{\Ddots} & & \Vdots \\ \Vdots & \Ddots & & \\ a+b & \cdots & a+b & 1 \end{pmatrix}$$


```

```


$$A = \begin{pmatrix} 1 & \cdots & \cdots & 1 \\ a+b & & & \Vdots \\ \Vdots & \textcolor{blue}{\Ddots} & \Ddots & \\ a+b & \cdots & a+b & 1 \end{pmatrix}$$


```

¹⁷On 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.

¹⁸We speak of the lines created by `\Ddots` and not the lines created by a command `\line` in `code-after`.

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

The same example without parallelization:

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

15.5 The “empty” cells

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

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

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

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

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

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

15.6 The option `exterior-arraycolsep`

The environment `{array}` inserts an horizontal space equal to `\arraycolsep` before and after each column. In particular, there is a space equal to `\arraycolsep` before and after the array. This feature of the environment `{array}` was probably not a good idea¹⁹. The environment `{matrix}` of `amsmath` and its variants (`{pmatrix}`, `{vmatrix}`, etc.) of `amsmath` prefer to delete these spaces with explicit instructions `\hskip -\arraycolsep`. The extension `nicematrix` does the same in all its environments, `{NiceArray}` included. However, if the user wants the environment `{NiceArray}` behaving by default like the environment `{array}` of `array` (for example, when adapting an existing document) it's possible to control this behaviour with the option `exterior-arraycolsep`, set by the command `\NiceMatrixOptions`. With this option, exterior spaces of length `\arraycolsep` will be inserted in the environments `{NiceArray}` (the other environments of `nicematrix` are not affected).

¹⁹In 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@{}}\dots\end{array}`.

15.7 The class option draft

The package `nicematrix` is rather slow when drawing the dotted lines (generated by `\Cdots`, `\Ldots`, `\Ddots`, etc. but also by `\hdottedline` or the specifier `:`).²⁰ That’s why, when the class option `draft` is used, the dotted lines are not drawn, for a faster compilation.

15.8 A technical problem with the argument of `\`

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

<pre>\begin{pNiceMatrix} a & \frac{AB}{2mm} 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{AB}{2mm} \align{\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.

<pre>\begin{pNiceMatrix} a & \multicolumn{1}{\frac{AB}{2mm}} b & c \end{pNiceMatrix}</pre>	$\begin{pmatrix} a & \frac{A}{B} \\ b & c \end{pmatrix}$
--	--

15.9 Obsolete environments

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

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

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

Since the version 3.8, an error is raised when one of these environments is used. It’s still possible to use these environments by loading `nicematrix` with the option `obsolete-environments`.

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

²⁰The 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.

16 Examples

16.1 Dotted lines

A tridiagonal matrix:

```

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

```

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

A permutation matrix:

```

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

```

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

An example with `\Iddots`:

```

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

```

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

An example with `\multicolumn`:

```

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

```

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

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}
```

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

An example for the resultant of two polynomials:

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

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

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

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

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

16.2 Width of the columns

In the following example, we use `{NiceMatrixBlock}` with the option `auto-columns-width` because we want the same automatic width for all the columns of the matrices.

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

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

16.3 How to highlight cells of the matrix

In order to highlight a cell of a matrix, it's possible to “draw” one of the correspondent nodes (the “normal node”, the “medium node” or the “large node”). In the following example, we use the “large nodes” of the diagonal of the matrix (with the Tikz key “`name suffix`”, it's easy to use the “large nodes”).

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

```
$\begin{pNiceArray}{>{\strut}CCCC}%
[create-large-nodes,margin,extra-margin = 2pt ,
code-after = {\begin{tikzpicture}
[name suffix = -large,
every node/.style = {draw,
```

```

inner sep = -\pgflinewidth/2]
\begin{tikzpicture}
\draw[fit = (1-1)] (0,0) rectangle (1,1);
\draw[fit = (2-2)] (1,0) rectangle (2,1);
\draw[fit = (3-3)] (2,0) rectangle (3,1);
\draw[fit = (4-4)] (3,0) rectangle (4,1);
\end{tikzpicture}
\begin{array}{cccc}
a_{11} & a_{12} & a_{13} & a_{14} \\
a_{21} & a_{22} & a_{23} & a_{24} \\
a_{31} & a_{32} & a_{33} & a_{34} \\
a_{41} & a_{42} & a_{43} & a_{44}
\end{array}

```

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

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

```

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

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

```

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

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

```

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

```

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

```

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

```

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

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

```

\tikzset{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}CCC}%
[create-large-nodes,margin,extra-margin=2pt,
code-after = {\tikz \path [name suffix = -large,
                           fill = red!15,
                           blend mode = multiply]
(1-1.north west)
|- (2-2.north west)
|- (3-3.north west)

```

```

|-(4-4.north west)
|-(4-4.south east)
|-(1-1.north west) ; } ]
A_{11} & A_{12} & A_{13} & A_{14} \\
A_{21} & A_{22} & A_{23} & A_{24} \\
A_{31} & A_{32} & A_{33} & A_{34} \\
A_{41} & A_{42} & A_{43} & A_{44} \\
\end{pNiceArray}

```

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

16.4 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}

```

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

17 Implementation

By default, the package `nicematrix` doesn't patch any existing code.

However, when the option `renew-dots` is used, the commands `\cdots`, `\ldots`, `\dots`, `\vdots`, `\ddots` and `\iddots` are redefined in the environments provided by `nicematrix` as explained previously. In the same way, if the option `renew-matrix` is used, the environment `{matrix}` of `amsmath` is redefined.

On the other hand, the environment `{array}` is never redefined.

Of course, the package `nicematrix` uses the features of the package `array`. It tries to be independent of its implementation. Unfortunately, it was not possible to be strictly independent: the package `nicematrix` relies upon the fact that the package `{array}` uses `\ialign` to begin the `\halign`.

17.1 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.²¹

```
<@@=nicematrix>
```

```

1 \RequirePackage{tikz}
2 \usetikzlibrary{fit}
3 \RequirePackage{expl3}[2019/07/01]

```

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

```

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

```

²¹cf. tex.stackexchange.com/questions/57424/using-of-usetikzlibrary-in-an-expl3-package-fails

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

```

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

```

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

We load `array` and `amsmath`.

```

14 \RequirePackage { array }
15 \RequirePackage { amsmath }
16 \RequirePackage { xparse } [ 2018-07-01 ]

17 \cs_new_protected:Npn \@@_error:n { \msg_error:nn { nicematrix } }
18 \cs_new_protected:Npn \@@_error:nn { \msg_error:nnn { nicematrix } }
19 \cs_new_protected:Npn \@@_error:nnn { \msg_error:nnnn { nicematrix } }
20 \cs_new_protected:Npn \@@_fatal:n { \msg_fatal:nn { nicematrix } }
21 \cs_new_protected:Npn \@@_fatal:nn { \msg_fatal:nn { nicematrix } }
22 \cs_new_protected:Npn \@@_msg_new:nn { \msg_new:nnn { nicematrix } }
23 \cs_new_protected:Npn \@@_msg_new:nnn { \msg_new:nnnn { nicematrix } }

24 \cs_new_protected:Npn \@@_msg_redirect_name:nn
25   { \msg_redirect_name:nnn { nicematrix } }

```

17.2 Technical definitions

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

```

26 \bool_new:N \c_@@_revtex_bool
27 \@ifclassloaded { revtex4-1 }
28   { \bool_set_true:N \c_@@_revtex_bool }
29   { }
30 \@ifclassloaded { revtex4-2 }
31   { \bool_set_true:N \c_@@_revtex_bool }
32   { }

```

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

```

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

35 \bool_if:NT \c_@@_draft_bool
36   { \msg_warning:nn { nicematrix } { Draft-mode } }

```

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

```

37 \ProvideDocumentCommand \iddots { }
38   {
39     \mathinner
40       {
41         \mkern 1 mu
42         \raise \p@ \hbox:n { . }
43         \mkern 2 mu
44         \raise 4 \p@ \hbox:n { . }
45         \mkern 2 mu
46         \raise 7 \p@ \vbox { \kern 7 pt \hbox:n { . } } \mkern 1 mu
47       }
48   }

```


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.

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

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

```
50 \int_new:N \g_@@_NiceMatrixBlock_int
```

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

```
51 \dim_new:N \l_@@_columns_width_dim
```

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

```
52 \seq_new:N \g_@@_names_seq
```

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

```
53 \bool_new:N \l_@@_in_env_bool
```

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

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

```
55 \cs_new_protected:Npn \@@_test_if_math_mode:
56 {
57   \if_mode_math: \else:
58     \@@_fatal:n { Outside~math~mode }
59   \fi:
60 }
```

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.²²

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

²²Of course, the user should solve the problem by adding the lacking ampersands.

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

```

62 \bool_new:N \c_@@_colortbl_loaded_bool
63 \AtBeginDocument
64 {
65   \ifpackageloaded { colortbl }
66   {
67     \bool_set_true:N \c_@@_colortbl_loaded_bool
68     \cs_set_protected:Npn \@@_vline_i: { { \CT@arc@ \vline } }
69   }
70   { }
71 }

```

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

```

72 \dim_new:N \l_@@_inter_dots_dim
73 \dim_set:Nn \l_@@_inter_dots_dim { 0.45 em }

```

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

```

74 \dim_new:N \l_@@_radius_dim
75 \dim_set:Nn \l_@@_radius_dim { 0.53 pt }

```

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

```

76 \str_new:N \g_@@_type_env_str
77 \tl_new:N \g_@@_code_after_tl

```

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

```

78 \int_new:N \l_@@_save_iRow_int
79 \int_new:N \l_@@_save_jCol_int

```

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

17.2.1 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`.

```

80 \int_new:N \l_@@_first_row_int
81 \int_set:Nn \l_@@_first_row_int 1

```

- **First column**

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

```

82 \int_new:N \l_@@_first_col_int
83 \int_set:Nn \l_@@_first_col_int 1

```

- **Last row**

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

```
84 \int_new:N \l_@@_last_row_int
85 \int_set:Nn \l_@@_last_row_int { -2 }
```

If, in an environment like `{pNiceArray}`, the option `last-row` is used without value, we will globally raise the following flag. It will be used to know if we have, after the construction of the array, to write in the `aux` file the number of the “last row”.²³

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

- **Last column**

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

```
87 \int_new:N \l_@@_last_col_int
88 \int_set:Nn \l_@@_last_col_int { -1 }
```

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

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

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

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

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

17.2.2 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`.

```
90 \bool_new:N \c_@@_siunitx_loaded_bool
91 \AtBeginDocument
92 {
93   \ifpackageloaded { siunitx }
94     { \bool_set_true:N \c_@@_siunitx_loaded_bool }
95     { }
96 }
```

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:

²³We can’t use `\l_@@_last_row_int` for this usage because, if `nicematrix` has read its value from the `aux` file, the value of the counter won’t be `-1` any longer.

```

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

```

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

```

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

```

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

```

97 \cs_set_protected:Npn \@@adapt_S_column:
98 {

```

In the preamble of the LaTeX document, the boolean `\c_@@_siunitx_loaded_bool` won't be known. That's why we test the existence of `\c_@@_siunitx_loaded_bool` and not its value.²⁴

```

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

```

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

```

103     \cs_set_eq:NN \NC@find \prg_do_nothing:
104     \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`).

```

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

```

²⁴Indeed, `nicematrix` may be used in the preamble of the LaTeX document. For example, in this document, we compose a matrix in the box `\ExampleOne` before loading `arydshln` (because `arydshln` is not totally compatible with `nicematrix`).

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

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

```

112     \cs_gset_eq:NN \@@_adapt_S_column: \prg_do_nothing:
113   }
114 }

```

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).

```

115 \cs_new_protected:Npn \@@_renew_NC@rewrite@S:
116 {
117   \renewcommand*{\NC@rewrite@S}[1] []
118   {
119     \@temptokena \exp_after:wN
120     {
121       \tex_the:D \@temptokena
122       > { \@@_Cell: \c_@@_table_collect_begin_tl S {##1} }
123       c
124       < { \c_@@_table_print_tl \@@_end_Cell: }
125     }
126     \NC@find
127   }
128 }

```

17.3 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.

```

129 \str_new:N \l_@@_pos_env_str
130 \str_set:Nn \l_@@_pos_env_str c

```

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

```

131 \bool_new:N \l_@@_exterior_arraycolsep_bool

```

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

```

132 \bool_new:N \l_@@_parallelize_diags_bool
133 \bool_set_true:N \l_@@_parallelize_diags_bool

```

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

```

134 \bool_new:N \l_@@_hlines_bool

```

The flag `\l_@@_nullify_dots_bool` corresponds to the option `nullify-dots`. When the flag is down, the instructions like `\vdots` are inserted within a `\hphantom` (and so the constructed matrix has exactly the same size as a matrix constructed with the classical `{matrix}` and `\ldots`, `\vdots`, etc.).

```

135 \bool_new:N \l_@@_nullify_dots_bool

```

The following flag will be used when the current options specify that all the columns of the array must have the same width equal to the largest width of a cell of the array (except the cells of the potential exterior columns).

```

136 \bool_new:N \l_@@_auto_columns_width_bool

```

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

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

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

```
138 \bool_new:N \l_@@_medium_nodes_bool
139 \bool_new:N \g_@@_medium_nodes_bool
140 \bool_new:N \l_@@_large_nodes_bool
141 \bool_new:N \g_@@_large_nodes_bool
```

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

```
142 \dim_new:N \l_@@_left_margin_dim
143 \dim_new:N \l_@@_right_margin_dim
144 \dim_new:N \g_@@_width_last_col_dim
145 \dim_new:N \g_@@_width_first_col_dim
```

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

```
146 \dim_new:N \l_@@_extra_left_margin_dim
147 \dim_new:N \l_@@_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.

```
148 \keys_define:nn { NiceMatrix / Global }
149 {
150   code-for-first-col .tl_set:N = \l_@@_code_for_first_col_tl ,
151   code-for-last-col .tl_set:N = \l_@@_code_for_last_col_tl ,
152   code-for-first-row .tl_set:N = \l_@@_code_for_first_row_tl ,
153   code-for-last-row .tl_set:N = \l_@@_code_for_last_row_tl ,
154   small .bool_set:N = \l_@@_small_bool ,
155   hlines .bool_set:N = \l_@@_hlines_bool ,
156   parallelize-diags .bool_set:N = \l_@@_parallelize_diags_bool ,
```

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

```
157   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
158   nullify-dots .bool_set:N = \l_@@_nullify_dots_bool ,
```

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

```
159   create-medium-nodes .bool_set:N = \l_@@_medium_nodes_bool ,
160   create-large-nodes .bool_set:N = \l_@@_large_nodes_bool ,
161   create-extra-nodes .meta:n =
162     { create-medium-nodes , create-large-nodes } ,
163   left-margin .dim_set:N = \l_@@_left_margin_dim ,
164   left-margin .default:n = \arraycolsep ,
165   right-margin .dim_set:N = \l_@@_right_margin_dim ,
166   right-margin .default:n = \arraycolsep ,
167   margin .meta:n = { left-margin = #1 , right-margin = #1 } ,
168   margin .default:n = \arraycolsep ,
169   extra-left-margin .dim_set:N = \l_@@_extra_left_margin_dim ,
170   extra-right-margin .dim_set:N = \l_@@_extra_right_margin_dim ,
171   extra-margin .meta:n =
172     { extra-left-margin = #1 , extra-right-margin = #1 } ,
```

The following lines are the properties `.value_required:n` and `.value_forbidden:n` or the keys. These properties have no effect because they are not transmitted by inheritance (unfortunately). We maintain these lines in the DTX only for the case of a modification of `l3keys`.

```

173 {*comment}
174   code-for-first-col .value_required:n = true ,
175   code-for-last-col .value_required:n = true ,
176   code-for-first-row .value_required:n = true ,
177   code-for-last-row .value_required:n = true ,
178   renew-dots .value_forbidden:n = true ,
179 {/comment}
180 }
```

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

```

181 \keys_define:nn { NiceMatrix / Env }
182 {
183   columns-width .code:n =
184     \str_if_eq:nnTF { #1 } { auto }
185     { \bool_set_true:N \l_@@_auto_columns_width_bool }
186     { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,
187   name .code:n =
188     \unless \ifmeasuring@
189       \str_set:Nn \l_tmpa_str { #1 }
190       \seq_if_in:NVTF \g_@@_names_seq \l_tmpa_str
191       { \@@_error:nn { Duplicate-name } { #1 } }
192       { \seq_gput_left:NV \g_@@_names_seq \l_tmpa_str }
193       \str_set_eq:NN \l_@@_name_str \l_tmpa_str
194     \fi ,
195   code-after .tl_gset:N = \g_@@_code_after_tl ,
196   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
197   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
198   last-row .int_set:N = \l_@@_last_row_int ,
199   last-row .default:n = -1 ,
```

The following lines are the properties `.value_required:n` and `.value_forbidden:n` or the keys. These properties have no effect because they are not transmitted by inheritance (unfortunately). We maintain these lines in the DTX only for the case of a modification of `l3keys`.

```

200 {*comment}
201   columns-width .value_required:n = true ,
202   name .value_required:n = true ,
203   code-after .value_required:n = true ,
204 {/comment}
205 }
```

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

```

206 \keys_define:nn { NiceMatrix }
207 {
208   NiceMatrixOptions .inherit:n =
209     {
210       NiceMatrix / Global ,
211     } ,
212   NiceMatrix .inherit:n =
213     {
214       NiceMatrix / Global ,
215       NiceMatrix / Env
216     } ,
217   NiceArray .inherit:n =
218     {
219       NiceMatrix / Global ,
220       NiceMatrix / Env ,
```

```

221     } ,
222     pNiceArray .inherit:n =
223     {
224         NiceMatrix / Global ,
225         NiceMatrix / Env ,
226     }
227 }

```

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

```

228 \keys_define:nn { NiceMatrix / NiceMatrixOptions }
229 {

```

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

```

230     renew-matrix .code:n = \@@_renew_matrix: ,
231     renew-matrix .value_forbidden:n = true ,
232     RenewMatrix .code:n = \@@_error:n { Option~RenewMatrix~suppressed }
233         \@@_renew_matrix: ,
234     transparent .meta:n = { renew-dots , renew-matrix } ,
235     transparent .value_forbidden:n = true,
236     Transparent .code:n = \@@_error:n { Option~Transparent~suppressed }
237         \@@_renew_matrix:
238         \bool_set_true:N \l_@@_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}`.

```

239     exterior-arraycolsep .bool_set:N = \l_@@_exterior_arraycolsep_bool ,

```

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

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

```

240     columns-width .code:n =
241         \str_if_eq:nnTF { #1 } { auto }
242         { \@@_error:n { Option~auto~for~columns-width } }
243         { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,

```

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

```

244     allow-duplicate-names .code:n =
245         \@@_msg_redirect_name:nn { Duplicate~name } { none } ,
246     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`).

```

247     letter-for-dotted-lines .code:n =
248     {
249         \int_compare:nTF { \tl_count:n { #1 } = \c_one_int }
250         { \str_set:Nx \l_@@_letter_for_dotted_lines_str { #1 } }
251         { \@@_error:n { Bad~value~for~letter~for~dotted~lines } }
252     } ,
253     letter-for-dotted-lines .value_required:n = true ,

254     unknown .code:n = \@@_error:n { Unknown~key~for~NiceMatrixOptions }
255 }

256 \str_new:N \l_@@_letter_for_dotted_lines_str
257 \str_set_eq:NN \l_@@_letter_for_dotted_lines_str \c_colon_str

```


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

```
258 \NewDocumentCommand \NiceMatrixOptions { m }
259 { \keys_set:nn { NiceMatrix / NiceMatrixOptions } { #1 } }
```

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

```
260 \keys_define:nn { NiceMatrix / NiceMatrix }
261 {
262   last-col .code:n = \tl_if_empty:nTF {#1}
263     { \@@_error:n { last-col~empty~for~NiceMatrix } }
264     { \int_set:Nn \l_@@_last_col_int { #1 } } ,
265   unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix }
266 }
```

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

```
267 \keys_define:nn { NiceMatrix / NiceArray }
268 {
```

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

```
269   c .code:n = \str_set:Nn \l_@@_pos_env_str c ,
270   t .code:n = \str_set:Nn \l_@@_pos_env_str t ,
271   b .code:n = \str_set:Nn \l_@@_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.

```
272   last-col .code:n = \tl_if_empty:nF {#1}
273     { \@@_error:n { last-col~non~empty~for~NiceArray } }
274     { \int_zero:N \l_@@_last_col_int ,
275   unknown .code:n = \@@_error:n { Unknown~option~for~NiceArray }
276 }

277 \keys_define:nn { NiceMatrix / pNiceArray }
278 {
279   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
280   last-col .code:n = \tl_if_empty:nF {#1}
281     { \@@_error:n { last-col~non~empty~for~NiceArray } }
282     { \int_zero:N \l_@@_last_col_int ,
283   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
284   last-row .int_set:N = \l_@@_last_row_int ,
285   last-row .default:n = -1 ,
286   unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix }
287 }
```

17.4 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}`).

```
288 \cs_new_protected:Nn \@@_Cell:
289 {
```

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

```
290   \int_gincr:N \c@jCol
```

Now, we increment the counter of the rows. We don't do this incrementation in the `\everycr` because some packages, like `arydshln`, create special rows in the `\halign` that we don't want to take into account.

```

291 \int_compare:nNnT \c@jCol = \c_one_int
292 {
293   \int_compare:nNnT \l_@@_first_col_int = \c_one_int
294   \@@_begin_of_row:
295 }
296 \int_gset:Nn \g_@@_col_total_int
297 { \int_max:nn \g_@@_col_total_int \c@jCol }

```

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).

```

298 \hbox_set:Nw \l_tmpa_box
299 \c_math_toggle_token
300 \bool_if:NT \l_@@_small_bool \scriptstyle

```

We will call *corners* of the matrix the cases which are at the intersection of the exterior rows and exterior columns (of course, the four corners doesn't always exist simultaneously). In a corner of the matrix, it would be logical to use none of the codes `\l_@@_code_for_first_row_tl` and *al*. As for now, this result is achieved only for the north-west corner (this allows an automatic numerotation of the rows and the columns with `iRow` and `jCol` with the first col and the first row — probably the preferential choice for such a numerotation).

```

301 \int_compare:nNnTF \c@iRow = \c_zero_int
302 { \int_compare:nNnT \c@jCol > \c_zero_int \l_@@_code_for_first_row_tl }
303 {
304   \int_compare:nNnT \c@iRow = \l_@@_last_row_int
305   \l_@@_code_for_last_row_tl
306 }
307 }

```

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.

```

308 \cs_new_protected:Nn \@@_begin_of_row:
309 {
310   \int_gincr:N \c@iRow
311   \dim_gset_eq:NN \g_@@_dp_ante_last_row_dim \g_@@_dp_last_row_dim
312   \dim_gzero:N \g_@@_dp_last_row_dim
313   \dim_gzero:N \g_@@_ht_last_row_dim
314 }

```

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.

```

315 \cs_new_protected:Npn \@@_actualization_for_first_and_last_row:
316 {
317   \int_compare:nNnT \c@iRow = \c_zero_int
318   {
319     \dim_gset:Nn \g_@@_dp_row_zero_dim
320     { \dim_max:nn \g_@@_dp_row_zero_dim { \box_dp:N \l_tmpa_box } }
321     \dim_gset:Nn \g_@@_ht_row_zero_dim
322     { \dim_max:nn \g_@@_ht_row_zero_dim { \box_ht:N \l_tmpa_box } }
323   }
324   \int_compare:nNnT \c@iRow = \c_one_int
325   {
326     \dim_gset:Nn \g_@@_ht_row_one_dim
327     { \dim_max:nn \g_@@_ht_row_one_dim { \box_ht:N \l_tmpa_box } }
328   }
329   \dim_gset:Nn \g_@@_ht_last_row_dim
330   { \dim_max:nn \g_@@_ht_last_row_dim { \box_ht:N \l_tmpa_box } }

```

```

331 \dim_gset:Nn \g_@@_dp_last_row_dim
332 { \dim_max:nn \g_@@_dp_last_row_dim { \box_dp:N \l_tmpa_box } }
333 }
334 \cs_new_protected:Nn \@@_end_Cell:
335 {
336 \c_math_toggle_token
337 \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”).

```

338 \dim_gset:Nn \g_@@_max_cell_width_dim
339 { \dim_max:nn \g_@@_max_cell_width_dim { \box_wd:N \l_tmpa_box } }

```

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

```

340 \@@_actualization_for_first_and_last_row:

```

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

```

341 \tikz
342 [
343 remember~picture ,
344 inner~sep = \c_zero_dim ,
345 minimum~width = \c_zero_dim ,
346 baseline
347 ]
348 \node
349 [
350 anchor = base ,
351 name = nm - \int_use:N \g_@@_env_int -
352 \int_use:N \c@iRow -
353 \int_use:N \c@jCol ,
354 alias =
355 \str_if_empty:NF \l_@@_name_str
356 {
357 \l_@@_name_str -
358 \int_use:N \c@iRow -
359 \int_use:N \c@jCol
360 }
361 ]
362 \bgroup
363 \box_use:N \l_tmpa_box
364 \egroup ;
365 }
366 \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.

```

367 \cs_new_protected:Nn \@@_renewcolumnntype:nn
368 {
369 \newcolumnntype #1 [ 2 ]
370 {
371 > {
372 \hbox_set:Nw \l_tmpa_box
373 \@@_Cell:
374 }
375 c
376 < {
377 \@@_end_Cell:
378 \hbox_set_end:
379 #2

```

```

380 \hbox_set:Nn \l_tmpb_box
381 { \makebox [ ##2 ] [ ##1 ] { \box_use:N \l_tmpa_box } }
382 \dim_set:Nn \l_tmpa_dim { \box_dp:N \l_tmpb_box }
383 \box_move_down:nn \l_tmpa_dim
384 {
385   \vbox:n
386   {
387     \hbox_to_wd:nn { \box_wd:N \l_tmpb_box }
388     {
389       \hfil
390       \tikz [ remember~picture , overlay ]
391       \coordinate (@@~north~east) ;
392     }
393     \hbox:n
394     {
395       \tikz [ remember~picture , overlay ]
396       \coordinate (@@~south~west) ;
397       \box_move_up:nn \l_tmpa_dim { \box_use:N \l_tmpb_box }
398     }
399   }
400 }

```

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?).

```

401 \tikz [ remember~picture , overlay ]
402 \node
403 [
404   node~contents = { } ,
405   name = nm - \int_use:N \g_@@_env_int -
406           \int_use:N \c@iRow -
407           \int_use:N \c@jCol - w,
408   alias =
409     \str_if_empty:NF \l_@@_name_str
410     {
411       \l_@@_name_str -
412       \int_use:N \c@iRow -
413       \int_use:N \c@jCol - w
414     } ,
415   inner~sep = \c_zero_dim ,
416   fit = (@@~south~west) (@@~north~east)
417 ]
418 ;
419 }
420 }
421 }

```

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.

```

422 \bool_if:NTF \c_@@_draft_bool
423 { \cs_set_protected:Npn \@@_instruction_of_type:n #1 { } }
424 {
425   \cs_new_protected:Npn \@@_instruction_of_type:n #1
426   {

```

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).

```

427     \tl_gput_right:cx
428     { g_@@_ #1 _ lines _ tl }
429     {
430       \use:c { @@ _ draw _ #1 : nn }
431       { \int_use:N \c@iRow }
432       { \int_use:N \c@jCol }
433     }
434   }
435 }

```

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`.

```

436 \cs_new_protected:Npn \@@_array:
437 {
438   \bool_if:NTF \c_@@_revtex_bool
439   {
440     \cs_set_eq:NN \@acoll \@arrayacol
441     \cs_set_eq:NN \@acolr \@arrayacol
442     \cs_set_eq:NN \@acol \@arrayacol
443     \cs_set:Npn \@halignto { }
444     \@array@array
445   }
446   \array

```

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

```

447   [ \l_@@_pos_env_str ]
448 }

```

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

```

449 \cs_new:Npn \@@_everycr:
450 { \noalign { \@@_everycr_i: } }
451 \cs_new_protected:Npn \@@_everycr_i:
452 {
453   \int_gzero:N \c@jCol
454   \bool_if:NT \l_@@_hlines_bool
455   {

```

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

```

456     \int_compare:nNnT \c@iRow > { -1 }
457     {
458       \int_compare:nNnF \c@iRow = \l_@@_last_row_int
459       {
460         \hrule \@height \arrayrulewidth
461         \skip_vertical:n { - \arrayrulewidth }
462       }
463     }
464   }
465 }

```

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

```

466 \cs_new_protected:Npn \@@_pre_array:
467 {
468   \cs_if_exist:NT \theiRow
469   { \int_set_eq:NN \l_@@_save_iRow_int \c@iRow }
470   \int_gzero_new:N \c@iRow
471   \cs_if_exist:NT \thejCol
472   { \int_set_eq:NN \l_@@_save_jCol_int \c@jCol }
473   \int_gzero_new:N \c@jCol
474   \normalbaselines

```

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

```

475   \bool_if:NT \l_@@_small_bool
476   {
477     \cs_set:Npn \arraystretch { 0.47 }
478     \dim_set:Nn \arraycolsep { 1.45 pt }
479   }

```

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

```

480   \bool_gset_eq:NN \g_@@_medium_nodes_bool \l_@@_medium_nodes_bool
481   \bool_gset_eq:NN \g_@@_large_nodes_bool \l_@@_large_nodes_bool

```

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

```

482   \cs_set:Npn \ialign
483   {
484     \bool_if:NTF \c_@@_colortbl_loaded_bool
485     {
486       \CT@everycr
487       {
488         \noalign { \cs_gset_eq:NN \CT@row@color \prg_do_nothing: }
489         \@@_everycr:
490       }
491     }
492     { \everycr { \@@_everycr: } }
493     \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`²⁵ 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`.

```

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

```

²⁵The option `small` of `nicematrix` changes (among other) the value of `\arraystretch`. This is done, of course, before the call of `{array}`.

After its first utilisation, the definition of `\ialign` will revert automatically to its default definition. With this programming, we will have, in the cells of the array, a clean version of `\ialign`.²⁶

```

506     \cs_set:Npn \ialign
507     {
508         \everycr { }
509         \tabskip = \c_zero_skip
510         \halign
511     }
512     \halign
513 }

```

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}`.

```

514     \newcolumntype L { > \@@_Cell: l < \@@_end_Cell: }
515     \newcolumntype C { > \@@_Cell: c < \@@_end_Cell: }
516     \newcolumntype R { > \@@_Cell: r < \@@_end_Cell: }

517     \cs_set_eq:NN \Ldots \@@_Ldots
518     \cs_set_eq:NN \Cdots \@@_Cdots
519     \cs_set_eq:NN \Vdots \@@_Vdots
520     \cs_set_eq:NN \Ddots \@@_Ddots
521     \cs_set_eq:NN \Iddots \@@_Iddots
522     \cs_set_eq:NN \hdottedline \@@_hdottedline:
523     \cs_set_eq:NN \Hspace \@@_Hspace:
524     \cs_set_eq:NN \hdotsfor \@@_Hdotsfor:
525     \cs_set_eq:NN \multicolumn \@@_multicolumn:nnn
526     \cs_set_eq:NN \Block \@@_Block:
527     \cs_set_eq:NN \OnlyMainNiceMatrix \@@_OnlyMainNiceMatrix:n
528     \bool_if:NT \l_@@_renew_dots_bool
529     {
530         \cs_set_eq:NN \ldots \@@_Ldots
531         \cs_set_eq:NN \cdots \@@_Cdots
532         \cs_set_eq:NN \vdots \@@_Vdots
533         \cs_set_eq:NN \ddots \@@_Ddots
534         \cs_set_eq:NN \iddots \@@_Iddots
535         \cs_set_eq:NN \dots \@@_Ldots
536         \cs_set_eq:NN \hdotsfor \@@_Hdotsfor:
537     }

```

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

```

538     \seq_gclear_new:N \g_@@_multicolumn_cells_seq
539     \seq_gclear_new:N \g_@@_multicolumn_sizes_seq

```

The counter `\c@iRow` will be used to count the rows of the array (its incrementation will be in the first cell of the row).

```

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

```

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

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

```

541     \int_gzero_new:N \g_@@_row_total_int

```

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

```

542     \int_gzero_new:N \g_@@_col_total_int
543     \cs_set_eq:NN \@ifnextchar \new@ifnextchar

```

²⁶The user will probably not employ directly `\ialign` in the array... but more likely environments that utilize `\ialign` internally (e.g.: `{substack}`).

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:`.

```

544 \cs_set_eq:NN \NC@find@w \relax
545 \cs_set_eq:NN \NC@find@W \relax
546 \@@_renewcolumnntype:nn w { }
547 \@@_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).

```

548 \tl_set_rescan:Nno
549 \l_@@_letter_for_dotted_lines_str { } \l_@@_letter_for_dotted_lines_str
550 \exp_args:NV \newcolumnntype \l_@@_letter_for_dotted_lines_str
551 {
552   !
553   {
554     \skip_horizontal:n { 0.53 pt }

```

If the array is an array with all the columns of the same width, we don’t ask for the creation of the extra nodes because we will use the “`col`” nodes for the vertical dotted line.

```

555 \bool_if:nF
556 {
557   \l_@@_auto_columns_width_bool
558   || \dim_compare_p:nNn \l_@@_columns_width_dim > \c_zero_dim
559 }
560 { \bool_gset_true:N \g_@@_large_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`.

```

561 \int_compare:nNnT \c@jCol > \g_@@_last_vdotted_col_int
562 {
563   \int_gset_eq:NN \g_@@_last_vdotted_col_int \c@jCol
564   \tl_gput_right:Nx \g_@@_code_after_tl

```

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

```

565   { \@@_vdottedline:n { \int_use:N \c@jCol } }
566 }
567 }
568 }
569 \int_gzero_new:N \g_@@_last_vdotted_col_int
570 \bool_if:NT \c_@@_siunitx_loaded_bool \@@_renew_NC@rewrite@S:
571 \int_gset:Nn \g_@@_last_vdotted_col_int { -1 }
572 \bool_gset_false:N \g_@@_last_col_found_bool

```


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

```

573 \tl_gclear_new:N \g_@@_Cdots_lines_tl
574 \tl_gclear_new:N \g_@@_Ldots_lines_tl
575 \tl_gclear_new:N \g_@@_Vdots_lines_tl
576 \tl_gclear_new:N \g_@@_Ddots_lines_tl
577 \tl_gclear_new:N \g_@@_Iddots_lines_tl
578 \tl_gclear_new:N \g_@@_Hdotsfor_lines_tl
579 }

```

17.5 The environment `{NiceArrayWithDelims}`

```

580 \NewDocumentEnvironment { NiceArrayWithDelims } { m m O { } m ! O { } }
581 {
582   \str_if_empty:NT \g_@@_type_env_str
583   {
584     \str_gset:Nn \g_@@_type_env_str
585       { environment ~ { NiceArrayWithDelims } }
586   }
587   \@@_adapt_S_column:
588   \@@_test_if_math_mode:
589   \bool_if:NT \l_@@_in_env_bool { \@@_fatal:n { Yet-in-env } }
590   \bool_set_true:N \l_@@_in_env_bool

```

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

```

591 \cs_if_exist:NT \tikz@library@external@loaded
592 {
593   \tikzset { external / export = false }
594   \cs_if_exist:NT \ifstandalone
595     { \tikzset { external / optimize = false } }
596 }

```

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

```

597 \int_gincr:N \g_@@_env_int
598 \bool_if:NF \l_@@_block_auto_columns_width_bool
599 { \dim_gzero_new:N \g_@@_max_cell_width_dim }

```

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

```

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

```

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

```

601 \bool_if:NTF \l_@@_NiceArray_bool
602 { \keys_set:nn { NiceMatrix / NiceArray } }
603 { \keys_set:nn { NiceMatrix / pNiceArray } }
604 { #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).

```

605 \int_compare:nNnT \l_@@_last_row_int = { -1 }
606 {
607   \bool_set_true:N \l_@@_last_row_without_value_bool

```

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

```

608 \str_if_empty:NTF \l_@@_name_str
609 {
610   \cs_if_exist:cT { @@_last_row_ \int_use:N \g_@@_env_int }
611   {
612     \int_set:Nn \l_@@_last_row_int
613       { \use:c { @@_last_row_ \int_use:N \g_@@_env_int } }
614   }

```

```

615     }
616     {
617         \cs_if_exist:cT { @@_last_row_ \l_@@_name_str }
618         {
619             \int_set:Nn \l_@@_last_row_int
620             { \use:c { @@_last_row_ \l_@@_name_str } }
621         }
622     }
623 }

```

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

```

624 \@@_pre_array:

```

We compute the width of the two delimiters.

```

625 \dim_gzero_new:N \g_@@_left_delim_dim
626 \dim_gzero_new:N \g_@@_right_delim_dim
627 \bool_if:NTF \l_@@_NiceArray_bool
628 {
629     \dim_gset:Nn \g_@@_left_delim_dim { 2 \arraycolsep }
630     \dim_gset:Nn \g_@@_right_delim_dim { 2 \arraycolsep }
631 }
632 {
633     \group_begin:
634     \dim_set_eq:NN \nulldelimiterspace \c_zero_dim
635     \hbox_set:Nn \l_tmpa_box
636     {
637         \c_math_toggle_token
638         \left #1 \vcenter to 3 cm { } \right.
639         \c_math_toggle_token
640     }
641     \dim_gset:Nn \g_@@_left_delim_dim { \box_wd:N \l_tmpa_box }
642     \hbox_set:Nn \l_tmpa_box
643     {
644         \dim_set_eq:NN \nulldelimiterspace \c_zero_dim
645         \c_math_toggle_token
646         \left. \vcenter to 3 cm { } \right #2
647         \c_math_toggle_token
648     }
649     \dim_gset:Nn \g_@@_right_delim_dim { \box_wd:N \l_tmpa_box }
650     \group_end:
651 }
652 }

```

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

```

653 \box_clear_new:N \l_@@_the_array_box

```

We construct the preamble of the array in \l_tmpa_tl.

```

654 \tl_set:Nn \l_tmpa_tl { #4 }
655 \int_compare:nNnTF \l_@@_first_col_int = \c_zero_int
656 { \tl_put_left:NV \l_tmpa_tl \c_@@_preamble_first_col_tl }
657 {
658     \bool_if:NT \l_@@_NiceArray_bool
659     {
660         \bool_if:NF \l_@@_exterior_arraycolsep_bool
661         { \tl_put_left:Nn \l_tmpa_tl { @ { } } }
662     }
663 }
664 \int_compare:nNnTF \l_@@_last_col_int > { -1 }
665 { \tl_put_right:NV \l_tmpa_tl \c_@@_preamble_last_col_tl }
666 {
667     \bool_if:NT \l_@@_NiceArray_bool
668     {
669         \bool_if:NF \l_@@_exterior_arraycolsep_bool
670         { \tl_put_right:Nn \l_tmpa_tl { @ { } } }
671     }
672 }

```

```

671     }
672 }

```

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).

```

673 \hbox_set:Nw \l_@@_the_array_box
674 \skip_horizontal:n \l_@@_left_margin_dim
675 \skip_horizontal:n \l_@@_extra_left_margin_dim
676 \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`).

```

677 \exp_args:NV \@@_array: \l_tmpa_tl
678 }

```

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 commands `\@@_vdottedline:n`).

```

679 {
680   \bool_if:nT
681   {
682     \l_@@_auto_columns_width_bool
683     || \dim_compare_p:nNn \l_@@_columns_width_dim > \c_zero_dim
684   }
685   {
686     \crrc
687     \int_compare:nNnT \l_@@_first_col_int = 0 { \omit & }
688     \omit

```

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

```

689     \skip_horizontal:N \arraycolsep
690     \tikz [ remember~picture , overlay ]
691       \coordinate [ name = nm - \int_use:N \g_@@_env_int - col - 0 ] ;
692     \skip_horizontal:n { - \arraycolsep }

```

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 because 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.

```

693     \bool_if:nTF
694     {
695       \l_@@_auto_columns_width_bool
696       && ! \l_@@_block_auto_columns_width_bool
697     }
698     {
699       \dim_gset:Nn \g_tmpa_dim
700       { \g_@@_max_cell_width_dim + 2 \arraycolsep }
701     }
702     {
703       \dim_gset:Nn \g_tmpa_dim
704       { \l_@@_columns_width_dim + 2 \arraycolsep }
705     }
706     \skip_horizontal:N \g_tmpa_dim
707     \tikz [ remember~picture , overlay ]
708       \coordinate [ name = nm - \int_use:N \g_@@_env_int - col - 1 ] ;

```

We begin a loop over the columns. The integer `\g_tmpa_int` will be the number of the current column. 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.

```

709     \int_gset:Nn \g_tmpa_int 1
710     \bool_if:nTF \g_@@_last_col_found_bool

```

```

711     { \prg_replicate:nn { \g_@@_col_total_int - 3 } }
712     { \prg_replicate:nn { \g_@@_col_total_int - 2 } }
713     {
714         &
715         \omit

```

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

```

716         \int_gincr:N \g_tmpa_int
717         \skip_horizontal:N \g_tmpa_dim

```

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

```

718         \tikz [ remember~picture , overlay ]
719         \coordinate
720         [
721             name = nm - \int_use:N \g_@@_env_int -
722                 col - \int_use:N \g_tmpa_int
723         ] ;
724     }

```

For the last column, we want a special treatment because of the final `\arraycolsep`.

```

725     &
726     \omit
727     \int_gincr:N \g_tmpa_int
728     \skip_horizontal:N \g_tmpa_dim
729     \skip_horizontal:n { - \arraycolsep }
730     \tikz [ remember~picture , overlay ]
731     \coordinate
732     [
733         name = nm - \int_use:N \g_@@_env_int -
734             col - \int_use:N \g_tmpa_int
735     ] ;
736     \skip_horizontal:N \arraycolsep
737 }
738 \endarray
739 \c_math_toggle_token
740 \skip_horizontal:n \l_@@_right_margin_dim
741 \skip_horizontal:n \l_@@_extra_right_margin_dim
742 \hbox_set_end:

```

```

743 \int_compare:nNnT \l_@@_last_row_int > { -2 }
744 {
745     \bool_if:NF \l_@@_last_row_without_value_bool
746     {
747         \int_compare:nNnF \l_@@_last_row_int = \c_iRow
748         {
749             \@@_error:n { Wrong~last~row }
750             \int_gset_eq:NN \l_@@_last_row_int \c_iRow
751         }
752     }
753 }

```

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).

```

754 \int_compare:nNnTF \l_@@_first_row_int = \c_zero_int
755 {
756     \dim_set:Nn \l_tmpa_dim
757     { \g_@@_dp_row_zero_dim + \lineskip + \g_@@_ht_row_zero_dim }
758 }
759 { \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”.²⁷

²⁷A 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`).

```

760 \int_compare:nNnTF \l_@@_last_row_int > { -2 }
761 {
762   \dim_set:Nn \l_tmpb_dim
763     { \g_@@_ht_last_row_dim + \lineskip + \g_@@_dp_last_row_dim }
764 }
765 { \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. 46).

```

766 \int_compare:nNnT \l_@@_first_col_int = \c_zero_int
767 {
768   \skip_horizontal:n \arraycolsep
769   \skip_horizontal:n \g_@@_width_first_col_dim
770 }

```

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}`.

```

771 \bool_if:NTF \l_@@_NiceArray_bool
772 {
773   \int_compare:nNnT \l_@@_first_row_int = \c_zero_int
774   {
775     \str_if_eq:VnTF \l_@@_pos_env_str { t }
776     {
777       \box_move_up:nn
778         { \l_tmpa_dim - \g_@@_ht_row_zero_dim + \g_@@_ht_row_one_dim }
779     }
780   }
781   {
782     \int_compare:nNnT \l_@@_last_row_int > 0
783     {
784       \str_if_eq:VnT \l_@@_pos_env_str { b }
785       {
786         \box_move_down:nn
787           {
788             \l_tmpb_dim
789             - \g_@@_dp_last_row_dim + \g_@@_dp_ante_last_row_dim
790           }
791       }
792     }
793   }
794   { \box_use_drop:N \l_@@_the_array_box }
795 }

```

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

```

796 {
797   \hbox_set:Nn \l_tmpa_box
798   {
799     \c_math_toggle_token
800     \left #1
801     \vcenter
802     {

```

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

```

803     \skip_vertical:n { - \l_tmpa_dim }
804     \hbox:n
805     {
806       \skip_horizontal:n { - \arraycolsep }
807       \box_use_drop:N \l_@@_the_array_box
808       \skip_horizontal:n { - \arraycolsep }
809     }

```

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

```

810     \skip_vertical:n { - \l_tmpb_dim }

```

```

811         }
812         \right #2
813         \c_math_toggle_token
814     }
815     \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }
816     \box_set_dp:Nn \l_tmpa_box { \box_dp:N \l_tmpa_box + \l_tmpb_dim }
817     \box_use_drop:N \l_tmpa_box
818 }

```

We take into account a potential “last column” (this “last column” has been constructed in an overlapping position and we have computed its width in `\g_@@_width_last_col_dim`: see p. 48).

```

819     \bool_if:NT \g_@@_last_col_found_bool
820     {
821         \skip_horizontal:n \g_@@_width_last_col_dim
822         \skip_horizontal:n \arraycolsep
823     }
824     \@@_after_array:
825 }

```

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

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

```

826 \tl_const:Nn \c_@@_preamble_first_col_tl
827 {
828     >
829     {
830         \@@_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.

```

831         \hbox_set:Nw \l_tmpa_box
832         \c_math_toggle_token
833         \bool_if:NT \l_@@_small_bool \scriptstyle

```

We insert `\l_@@_code_for_first_col_tl...` but we don’t insert it in the potential “first row” and in the potential “last row”.

```

834         \bool_if:nT
835         {
836             \int_compare_p:nNn \c@iRow > \c_zero_int
837             &&
838             (
839                 \int_compare_p:nNn \l_@@_last_row_int < 0
840                 ||
841                 \int_compare_p:nNn \c@iRow < \l_@@_last_row_int
842             )
843         }
844         { \l_@@_code_for_first_col_tl }
845     }
846     1
847     <
848     {
849         \c_math_toggle_token
850         \hbox_set_end:
851         \@@_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.

```

852         \dim_gset:Nn \g_@@_width_first_col_dim
853         {
854             \dim_max:nn
855             \g_@@_width_first_col_dim
856             { \box_wd:N \l_tmpa_box }
857         }

```

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

```

858         \hbox_overlap_left:n

```

```

859     {
860         \tikz
861         [
862             remember~picture ,
863             inner~sep = \c_zero_dim ,
864             minimum~width = \c_zero_dim ,
865             baseline
866         ]
867         \node
868         [
869             anchor = base ,
870             name =
871             nm -
872             \int_use:N \g_@@_env_int -
873             \int_use:N \c@iRow -
874             0 ,
875             alias =
876             \str_if_empty:NF \l_@@_name_str
877             {
878                 \l_@@_name_str -
879                 \int_use:N \c@iRow -
880                 0
881             }
882         ]
883         { \box_use:N \l_tmpa_box } ;
884         \skip_horizontal:n
885         {
886             \g_@@_left_delim_dim +
887             \l_@@_left_margin_dim +
888             \l_@@_extra_left_margin_dim
889         }
890     }
891     \skip_horizontal:n { - 2 \arraycolsep }
892 }
893 }

```

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

```

894 \tl_const:Nn \c_@@_preamble_last_col_tl
895 {
896     >
897     {

```

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

```

898     \bool_gset_true:N \g_@@_last_col_found_bool
899     \int_gincr:N \c@jCol
900     \int_gset:Nn \g_@@_col_total_int
901     { \int_max:nn \g_@@_col_total_int \c@jCol }

```

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

```

902     \hbox_set:Nw \l_tmpa_box
903     \c_math_toggle_token
904     \bool_if:NT \l_@@_small_bool \scriptstyle

```

We insert `\l_@@_code_for_last_col_tl...` but we don’t insert it in the potential “first row” and in the potential “last row”.

```

905     \bool_if:nT
906     {
907         \int_compare_p:nNn \c@iRow > \c_zero_int
908         &&
909         (
910             \int_compare_p:nNn \l_@@_last_row_int < 0
911             ||
912             \int_compare_p:nNn \c@iRow < \l_@@_last_row_int
913         )
914     }

```

```

915         { \l_@@_code_for_last_col_tl }
916     }
917     1
918     <
919     {
920         \c_math_toggle_token
921         \hbox_set_end:
922         \@@_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.

```

923         \dim_gset:Nn \g_@@_width_last_col_dim
924         {
925             \dim_max:nn
926             \g_@@_width_last_col_dim
927             { \box_wd:N \l_tmpa_box }
928         }
929         \skip_horizontal:n { - 2 \arraycolsep }

```

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

```

930         \hbox_overlap_right:n
931         {
932             \skip_horizontal:n
933             {
934                 \g_@@_right_delim_dim +
935                 \l_@@_right_margin_dim +
936                 \l_@@_extra_right_margin_dim
937             }
938             \tikz
939             [
940                 remember~picture ,
941                 inner~sep = \c_zero_dim ,
942                 minimum~width = \c_zero_dim ,
943                 baseline
944             ]
945             \node
946             [
947                 anchor = base ,
948                 name =
949                     nm -
950                     \int_use:N \g_@@_env_int -
951                     \int_use:N \c@iRow -
952                     \int_use:N \c@jCol ,
953                 alias =
954                     \str_if_empty:NF \l_@@_name_str
955                     {
956                         \l_@@_name_str -
957                         \int_use:N \c@iRow -
958                         \int_use:N \c@jCol
959                     }
960             ]
961             { \box_use:N \l_tmpa_box } ;
962         }
963     }
964 }

```

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.

```

965 \NewDocumentEnvironment { NiceArray } { }
966 {
967     \bool_set_true:N \l_@@_NiceArray_bool
968     \str_if_empty:NT \g_@@_type_env_str
969     { \str_gset:Nn \g_@@_type_env_str { environment ~ { 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).

```

970   \NiceArrayWithDelims . .
971 }
972 { \endNiceArrayWithDelims }

```

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

```

973 \NewDocumentEnvironment { pNiceArray } { }
974 {
975   \str_if_empty:NT \g_@@_type_env_str
976   { \str_gset:Nn \g_@@_type_env_str { environment ~ { pNiceArray } } }
977   \@@_test_if_math_mode:
978   \NiceArrayWithDelims ( )
979 }
980 { \endNiceArrayWithDelims }

981 \NewDocumentEnvironment { bNiceArray } { }
982 {
983   \str_if_empty:NT \g_@@_type_env_str
984   { \str_gset:Nn \g_@@_type_env_str { environment ~ { bNiceArray } } }
985   \@@_test_if_math_mode:
986   \NiceArrayWithDelims [ ]
987 }
988 { \endNiceArrayWithDelims }

989 \NewDocumentEnvironment { BNiceArray } { }
990 {
991   \str_if_empty:NT \g_@@_type_env_str
992   { \str_gset:Nn \g_@@_type_env_str { environment ~ { BNiceArray } } }
993   \@@_test_if_math_mode:
994   \NiceArrayWithDelims \{ \}
995 }
996 { \endNiceArrayWithDelims }

997 \NewDocumentEnvironment { vNiceArray } { }
998 {
999   \str_if_empty:NT \g_@@_type_env_str
1000   { \str_gset:Nn \g_@@_type_env_str { environment ~ { vNiceArray } } }
1001   \@@_test_if_math_mode:
1002   \NiceArrayWithDelims | |
1003 }
1004 { \endNiceArrayWithDelims }

1005 \NewDocumentEnvironment { VNiceArray } { }
1006 {
1007   \str_if_empty:NT \g_@@_type_env_str
1008   { \str_gset:Nn \g_@@_type_env_str { environment ~ { VNiceArray } } }
1009   \@@_test_if_math_mode:
1010   \NiceArrayWithDelims \| \|
1011 }
1012 { \endNiceArrayWithDelims }

```

17.6 The environment {NiceMatrix} and its variants

```

1013 \cs_new_protected:Npn \@@_define_env:n #1
1014 {
1015   \NewDocumentEnvironment { #1 NiceMatrix } { ! 0 { } }
1016   {
1017     \str_gset:Nn \g_@@_type_env_str { environment ~ { #1 NiceMatrix } }
1018     \keys_set:nn { NiceMatrix / NiceMatrix } { ##1 }
1019     \begin { #1 NiceArray }
1020     {
1021       *

```

```

1022         {
1023             \int_compare:nNnTF \l_@@_last_col_int = { -1 }
1024                 \c@MaxMatrixCols
1025                 { \int_eval:n { \l_@@_last_col_int - 1 } }
1026         }
1027         C
1028     }
1029 }
1030 { \end { #1 NiceArray } }
1031 }
1032 \@@_define_env:n { }
1033 \@@_define_env:n p
1034 \@@_define_env:n b
1035 \@@_define_env:n B
1036 \@@_define_env:n v
1037 \@@_define_env:n V

```

17.7 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.

```

1038 \prg_set_conditional:Npnn \@@_if_not_empty_cell:nn #1 #2 { T , TF }
1039 {

```

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.

```

1040     \bool_set_false:N \l_tmpa_bool
1041     \cs_if_exist:cTF
1042     { @@ _ dotted _ \int_use:N #1 - \int_use:N #2 }
1043     \prg_return_true:
1044     {

```

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.

```

1045     \cs_if_free:cTF
1046     {
1047         pgf@sh@ns@nm -
1048         \int_use:N \g_@@_env_int -
1049         \int_use:N #1 -
1050         \int_use:N #2
1051     }
1052     { \prg_return_false: }
1053     {

```

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.

```

1054 \bool_set_false:N \l_tmpa_bool
1055 \cs_if_exist:cT
1056 { @@ _ empty _ \int_use:N #1 - \int_use:N #2 }
1057 {
1058   \int_compare:nNnT
1059   { \use:c { @@ _ empty _ \int_use:N #1 - \int_use:N #2 } }
1060   =
1061   \g_@@_env_int
1062   { \bool_set_true:N \l_tmpa_bool }
1063 }
1064 \bool_if:NTF \l_tmpa_bool
1065 \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.

```

1066 {
1067   \begin { pgfpicture }

```

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

```

1068   \tl_set:Nx \l_tmpa_tl
1069   {
1070     nm -
1071     \int_use:N \g_@@_env_int -
1072     \int_use:N #1 -
1073     \int_use:N #2
1074   }
1075   \pgfpointanchor \l_tmpa_tl { east }
1076   \dim_gset:Nn \g_tmpa_dim \pgf@x
1077   \pgfpointanchor \l_tmpa_tl { west }
1078   \dim_gset:Nn \g_tmpb_dim \pgf@x
1079   \end { pgfpicture }
1080   \dim_compare:nNnTF
1081   { \dim_abs:n { \g_tmpb_dim - \g_tmpa_dim } } < { 0.5 pt }
1082   \prg_return_false:
1083   \prg_return_true:
1084 }
1085 }
1086 }
1087 }

```

17.8 After the construction of the array

```

1088 \cs_new_protected:Nn \@@_after_array:
1089 {
1090   \int_compare:nNnTF \c_iRow > \c_zero_int
1091   \@@_after_array_i:
1092   {
1093     \@@_error:n { Zero~row }
1094     \@@_restore_iRow_jCol:
1095   }
1096 }

```

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

```

1097 \cs_new_protected:Nn \@@_after_array_i:
1098 {
1099   \group_begin:
1100   \cs_if_exist:NT \tikz@library@external@loaded
1101   { \tikzset { external / export = false } }

```

Now, the definition of `\c@jCol` and `\g_@@_col_total_int` change: `\c@jCol` will be the number of columns without the “last column”; `\g_@@_col_total_int` will be the number of columns with this “last column”.²⁸

```
1102 \int_gset_eq:NN \c@jCol \g_@@_col_total_int
1103 \bool_if:nT \g_@@_last_col_found_bool { \int_gdecr:N \c@jCol }
```

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

```
1104 \int_gset_eq:NN \g_@@_row_total_int \c@iRow
1105 \int_compare:nNnT \l_@@_last_row_int > { -1 }
1106 { \int_gsub:Nn \c@iRow \c_one_int }
```

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

```
1107 \bool_if:NT \l_@@_last_row_without_value_bool
1108 {
1109   \iow_now:Nn \@mainaux \ExplSyntaxOn
1110   \iow_now:Nx \@mainaux
1111   {
1112     \cs_gset:cpn { @@_last_row_ \int_use:N \g_@@_env_int }
1113     { \int_use:N \g_@@_row_total_int }
1114   }
}
```

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.

```
1115 \str_if_empty:NF \l_@@_name_str
1116 {
1117   \iow_now:Nx \@mainaux
1118   {
1119     \cs_gset:cpn { @@_last_row_ \l_@@_name_str }
1120     { \int_use:N \g_@@_row_total_int }
1121   }
1122 }
1123 \iow_now:Nn \@mainaux \ExplSyntaxOff
1124 }
```

By default, the diagonal lines will be parallelized²⁹. 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.

```
1125 \bool_if:NT \l_@@_parallelize_diags_bool
1126 {
1127   \int_zero_new:N \l_@@_ddots_int
1128   \int_zero_new:N \l_@@_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.

```
1129 \dim_zero_new:N \l_@@_delta_x_one_dim
1130 \dim_zero_new:N \l_@@_delta_y_one_dim
1131 \dim_zero_new:N \l_@@_delta_x_two_dim
1132 \dim_zero_new:N \l_@@_delta_y_two_dim
1133 }
```

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

```
1134 \bool_if:nTF \g_@@_medium_nodes_bool
1135 {
1136   \bool_if:NTF \g_@@_large_nodes_bool
1137   @@_create_medium_and_large_nodes:
1138   @@_create_medium_nodes:
```

²⁸We remind that the potential “first column” has the number 0.

²⁹It’s possible to use the option `parallelize-diags` to disable this parallelization.

```

1139     }
1140     { \bool_if:NT \g_@@_large_nodes_bool \@@_create_large_nodes: }
1141     \int_zero_new:N \l_@@_initial_i_int
1142     \int_zero_new:N \l_@@_initial_j_int
1143     \int_zero_new:N \l_@@_final_i_int
1144     \int_zero_new:N \l_@@_final_j_int
1145     \bool_set_false:N \l_@@_initial_open_bool
1146     \bool_set_false:N \l_@@_final_open_bool

```

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

```

1147     \bool_if:NT \l_@@_small_bool
1148     {
1149         \dim_set:Nn \l_@@_radius_dim { 0.37 pt }
1150         \dim_set:Nn \l_@@_inter_dots_dim { 0.25 em }
1151     }

```

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

```

1152     \g_@@_Hdotsfor_lines_tl
1153     \g_@@_Vdots_lines_tl
1154     \g_@@_Ddots_lines_tl
1155     \g_@@_Iddots_lines_tl
1156     \g_@@_Cdots_lines_tl
1157     \g_@@_Ldots_lines_tl

```

Now, the code-after.

```

1158     \tikzset
1159     {
1160         every~picture / .style =
1161         {
1162             overlay ,
1163             remember~picture ,
1164             name~prefix = nm - \int_use:N \g_@@_env_int -
1165         }
1166     }
1167     \cs_set_eq:NN \line \@@_line:nn
1168     \g_@@_code_after_tl
1169     \tl_gclear:N \g_@@_code_after_tl
1170     \group_end:
1171     \str_gclear:N \g_@@_type_env_str
1172     \@@_restore_iRow_jCol:
1173 }
1174 \cs_new_protected:Nn \@@_restore_iRow_jCol:
1175 {
1176     \cs_if_exist:NT \theiRow { \int_gset_eq:NN \c@iRow \l_@@_save_iRow_int }
1177     \cs_if_exist:NT \thejCol { \int_gset_eq:NN \c@jCol \l_@@_save_jCol_int }
1178 }

```

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

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

For a closed extremity, we use the normal node and for a open one, we use the “medium node” or, if it exists, the `w` node.

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

The command `\l_@@_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.

```
1179 \cs_new_protected:Nn \l_@@_find_extremities_of_line:nnnn
1180 {
```

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

```
1181   \cs_set:cpn { @@ _ dotted _ #1 - #2 } { }
```

Initialization of variables.

```
1182   \int_set:Nn \l_@@_initial_i_int { #1 }
1183   \int_set:Nn \l_@@_initial_j_int { #2 }
1184   \int_set:Nn \l_@@_final_i_int { #1 }
1185   \int_set:Nn \l_@@_final_j_int { #2 }
```

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

```
1186   \bool_set_false:N \l_@@_stop_loop_bool
1187   \bool_do_until:Nn \l_@@_stop_loop_bool
1188   {
1189     \int_add:Nn \l_@@_final_i_int { #3 }
1190     \int_add:Nn \l_@@_final_j_int { #4 }
```

We test if we are still in the matrix.

```
1191     \bool_set_false:N \l_@@_final_open_bool
1192     \int_compare:nNnTF \l_@@_final_i_int > \c@iRow
1193     {
1194       \int_compare:nNnT { #3 } = 1
1195       { \bool_set_true:N \l_@@_final_open_bool }
1196     }
1197     {
1198       \int_compare:nNnTF \l_@@_final_j_int < 1
1199       {
1200         \int_compare:nNnT { #4 } = { -1 }
1201         { \bool_set_true:N \l_@@_final_open_bool }
1202       }
1203       {
1204         \int_compare:nNnT \l_@@_final_j_int > \c@jCol
1205         {
1206           \int_compare:nNnT { #4 } = 1
1207           { \bool_set_true:N \l_@@_final_open_bool }
1208         }
1209       }
1210     }
1211     \bool_if:NTF \l_@@_final_open_bool
```

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

```
1212   {
```

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).

```

1213         \int_sub:Nn \l_@@_final_i_int { #3 }
1214         \int_sub:Nn \l_@@_final_j_int { #4 }
1215         \bool_set_true:N \l_@@_stop_loop_bool
1216     }

```

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`.

```

1217     {
1218         \@@_if_not_empty_cell:nnTF \l_@@_final_i_int \l_@@_final_j_int
1219         { \bool_set_true:N \l_@@_stop_loop_bool }

```

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

```

1220         {
1221             \cs_set:cpn
1222             {
1223                 @@ _ dotted _
1224                 \int_use:N \l_@@_final_i_int -
1225                 \int_use:N \l_@@_final_j_int
1226             }
1227         { }
1228     }
1229 }
1230 }

```

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).

```

1231     \cs_if_free:cT
1232     {
1233         pgf@sh@ns@nm -
1234         \int_use:N \g_@@_env_int -
1235         \int_use:N \l_@@_final_i_int -
1236         \int_use:N \l_@@_final_j_int
1237     }
1238     {
1239         \bool_if:NF \l_@@_final_open_bool
1240         {
1241             \msg_error:nnx { nicematrix } { Impossible~line }
1242             { \int_use:N \l_@@_final_i_int }
1243             \bool_set_true:N \l_@@_impossible_line_bool
1244         }
1245     }

```

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

```

1246     \bool_set_false:N \l_@@_stop_loop_bool
1247     \bool_do_until:Nn \l_@@_stop_loop_bool
1248     {
1249         \int_sub:Nn \l_@@_initial_i_int { #3 }
1250         \int_sub:Nn \l_@@_initial_j_int { #4 }
1251         \bool_set_false:N \l_@@_initial_open_bool
1252         \int_compare:nNnTF \l_@@_initial_i_int < 1
1253         {
1254             \int_compare:nNnT { #3 } = 1
1255             { \bool_set_true:N \l_@@_initial_open_bool }
1256         }
1257     }

```

```

1258     \int_compare:nNnTF \l_@@_initial_j_int < 1
1259     {
1260         \int_compare:nNnT { #4 } = 1
1261         { \bool_set_true:N \l_@@_initial_open_bool }
1262     }
1263     {
1264         \int_compare:nNnT \l_@@_initial_j_int > \c@jCol
1265         {
1266             \int_compare:nNnT { #4 } = { -1 }
1267             { \bool_set_true:N \l_@@_initial_open_bool }
1268         }
1269     }
1270 }
1271 \bool_if:NTF \l_@@_initial_open_bool
1272 {
1273     \int_add:Nn \l_@@_initial_i_int { #3 }
1274     \int_add:Nn \l_@@_initial_j_int { #4 }
1275     \bool_set_true:N \l_@@_stop_loop_bool
1276 }
1277 {
1278     \@@_if_not_empty_cell:nnTF
1279     \l_@@_initial_i_int \l_@@_initial_j_int
1280     { \bool_set_true:N \l_@@_stop_loop_bool }
1281     {
1282         \cs_set:cpn
1283         {
1284             @@ _ dotted _
1285             \int_use:N \l_@@_initial_i_int -
1286             \int_use:N \l_@@_initial_j_int
1287         }
1288         { }
1289     }
1290 }
1291 }

```

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).

```

1292 \cs_if_free:cT
1293 {
1294     pgf@sh@ns@nm -
1295     \int_use:N \g_@@_env_int -
1296     \int_use:N \l_@@_initial_i_int -
1297     \int_use:N \l_@@_initial_j_int
1298 }
1299 {
1300     \bool_if:NF \l_@@_initial_open_bool
1301     {
1302         \msg_error:nnx { nicematrix } { Impossible~line }
1303         { \int_use:N \l_@@_initial_i_int }
1304         \bool_set_true:N \l_@@_impossible_line_bool
1305     }
1306 }

```

If we have at least one open extremity, we create the “medium nodes” in the matrix³⁰. We remind that, when used once, the command `\@@_create_medium_nodes:` becomes no-op in the current TeX group.

```

1307     \bool_if:nT \l_@@_initial_open_bool \@@_create_medium_nodes:
1308     \bool_if:NT \l_@@_final_open_bool \@@_create_medium_nodes:
1309 }

```

³⁰We 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 draw ³¹. 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}`.

```

1310 \cs_new_protected:Nn \@@_retrieve_coords:nn
1311 {
1312   \dim_gzero_new:N \g_@@_x_initial_dim
1313   \dim_gzero_new:N \g_@@_y_initial_dim
1314   \dim_gzero_new:N \g_@@_x_final_dim
1315   \dim_gzero_new:N \g_@@_y_final_dim
1316   \begin { tikzpicture } [ remember-picture ]
1317     \tikz@parse@node \pgfutil@firstofone
1318       ( nm - \int_use:N \g_@@_env_int -
1319         \int_use:N \l_@@_initial_i_int -
1320         \int_use:N \l_@@_initial_j_int #1 )
1321     \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1322     \dim_gset:Nn \g_@@_y_initial_dim \pgf@y
1323     \tikz@parse@node \pgfutil@firstofone
1324       ( nm - \int_use:N \g_@@_env_int -
1325         \int_use:N \l_@@_final_i_int -
1326         \int_use:N \l_@@_final_j_int #2 )
1327     \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1328     \dim_gset:Nn \g_@@_y_final_dim \pgf@y
1329   \end { tikzpicture }
1330 }
1331 \cs_generate_variant:Nn \@@_retrieve_coords:nn { x x }

```

For the horizontal lines with open extremities, we must take into account the “col” nodes created in the environments which have a fixed width of the columns. The following command will recompute the x -value of the extremities in this case (erasing the value computed in `\@@_retrieve_coords:nn`).

```

1332 \cs_new_protected:Nn \@@_adjust_with_col_nodes:
1333 {
1334   \bool_if:NT \l_@@_initial_open_bool
1335   {
1336     \cs_if_exist:cT
1337       { pgf@sh@ns@nm - \int_use:N \g_@@_env_int - col - 0 }
1338     {
1339       \begin { tikzpicture } [ remember-picture ]
1340         \tikz@parse@node \pgfutil@firstofone
1341           ( nm - \int_use:N \g_@@_env_int - col - 0 )
1342         \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1343         \end { tikzpicture }
1344       }
1345     }
1346   \bool_if:NT \l_@@_final_open_bool
1347   {
1348     \cs_if_exist:cT
1349     {
1350       pgf@sh@ns@nm - \int_use:N \g_@@_env_int - col -
1351       \int_use:N \c@jCol
1352     }
1353     {
1354       \begin { tikzpicture } [ remember-picture ]
1355         \tikz@parse@node \pgfutil@firstofone
1356           ( nm - \int_use:N \g_@@_env_int - col - \int_use:N \c@jCol )

```

³¹In fact, with diagonal lines, or vertical lines in columns of type L or R, an adjustment of one of the coordinates may be done.

```

1357         \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1358     \end { tikzpicture }
1359 }
1360 }
1361 }

```

```

1362 \cs_new_protected:Nn \@@_draw_Ldots:nn
1363 {
1364     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1365     {
1366         \bool_set_false:N \l_@@_impossible_line_bool
1367         \@@_find_extremities_of_line:nnnn { #1 } { #2 } \c_zero_int \c_one_int
1368         \bool_if:NF \l_@@_impossible_line_bool \@@_actually_draw_Ldots:
1369     }
1370 }

```

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`.

```

1371 \cs_new_protected:Nn \@@_actually_draw_Ldots:
1372 {
1373     \@@_retrieve_coords:xx
1374     {
1375         \bool_if:NTF \l_@@_initial_open_bool
1376         {

```

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

```

1377         \cs_if_exist:cTF
1378         {
1379             pgf@sh@ns@nm
1380             - \int_use:N \g_@@_env_int
1381             - \int_use:N \l_@@_initial_i_int
1382             - \int_use:N \l_@@_initial_j_int - w
1383         }
1384         { - w.base~west }
1385         { - medium.base~west }
1386     }
1387     { .base~east }
1388 }
1389 {
1390     \bool_if:NTF \l_@@_final_open_bool
1391     {
1392         \cs_if_exist:cTF
1393         {
1394             pgf@sh@ns@nm
1395             - \int_use:N \g_@@_env_int
1396             - \int_use:N \l_@@_final_i_int
1397             - \int_use:N \l_@@_final_j_int - w
1398         }
1399         { - w.base~east }
1400         { - medium.base~east }
1401     }
1402     { .base~west }
1403 }
1404 \@@_adjust_with_col_nodes:
1405 \bool_if:NT \l_@@_initial_open_bool
1406 { \dim_gset_eq:NN \g_@@_y_initial_dim \g_@@_y_final_dim }
1407 \bool_if:NT \l_@@_final_open_bool
1408 { \dim_gset_eq:NN \g_@@_y_final_dim \g_@@_y_initial_dim }

```

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

```

1409 \dim_gadd:Nn \g_@@_y_initial_dim { 0.53 pt }
1410 \dim_gadd:Nn \g_@@_y_final_dim { 0.53 pt }
1411 \@@_draw_tikz_line:
1412 }

1413 \cs_new_protected:Nn \@@_draw_Cdots:nn
1414 {
1415   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1416   {
1417     \bool_set_false:N \l_@@_impossible_line_bool
1418     \@@_find_extremities_of_line:nnnn { #1 } { #2 } \c_zero_int \c_one_int
1419     \bool_if:NF \l_@@_impossible_line_bool
1420     {
1421       \@@_retrieve_coords:xx
1422       {
1423         \bool_if:NTF \l_@@_initial_open_bool
1424         {
1425           \cs_if_exist:cTF
1426           {
1427             pgf@sh@ns@nm
1428             - \int_use:N \g_@@_env_int
1429             - \int_use:N \l_@@_initial_i_int
1430             - \int_use:N \l_@@_initial_j_int - w
1431           }
1432           { - w.mid~west }
1433           { - medium.mid~west }
1434         }
1435         { .mid~east }
1436       }
1437       {
1438         \bool_if:NTF \l_@@_final_open_bool
1439         {
1440           \cs_if_exist:cTF
1441           {
1442             pgf@sh@ns@nm
1443             - \int_use:N \g_@@_env_int
1444             - \int_use:N \l_@@_final_i_int
1445             - \int_use:N \l_@@_final_j_int - w
1446           }
1447           { - w.mid~east }
1448           { - medium.mid~east }
1449         }
1450         { .mid~west }
1451       }
1452       \@@_adjust_with_col_nodes:
1453       \bool_if:NT \l_@@_initial_open_bool
1454       { \dim_gset_eq:NN \g_@@_y_initial_dim \g_@@_y_final_dim }
1455       \bool_if:NT \l_@@_final_open_bool
1456       { \dim_gset_eq:NN \g_@@_y_final_dim \g_@@_y_initial_dim }
1457       \@@_draw_tikz_line:
1458     }
1459   }
1460 }

```

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?).

```

1461 \cs_new_protected:Nn \@@_draw_Vdots:nn
1462 {
1463   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1464   {
1465     \bool_set_false:N \l_@@_impossible_line_bool

```

```

1466 \@@_find_extremities_of_line:nnnn { #1 } { #2 } \c_one_int \c_zero_int
1467 \bool_if:NF \l_@@_impossible_line_bool
1468 {
1469   \@@_retrieve_coords:xx
1470   {
1471     \bool_if:NTF \l_@@_initial_open_bool
1472     { - medium.north-west }
1473     { .south~west }
1474   }
1475   {
1476     \bool_if:NTF \l_@@_final_open_bool
1477     { - medium.south-west }
1478     { .north~west }
1479   }

```

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

```

1480 \bool_set:Nn \l_tmpa_bool
1481 { \dim_compare_p:nNn \g_@@_x_initial_dim = \g_@@_x_final_dim }
1482 \@@_retrieve_coords:xx
1483 {
1484   \bool_if:NTF \l_@@_initial_open_bool
1485   { - medium.north }
1486   { .south }
1487 }
1488 {
1489   \bool_if:NTF \l_@@_final_open_bool
1490   { - medium.south }
1491   { .north }
1492 }

```

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

```

1493 \bool_set:Nn \l_tmpb_bool
1494 { \dim_compare_p:nNn \g_@@_x_initial_dim = \g_@@_x_final_dim }
1495 \bool_if:NF \l_tmpb_bool
1496 {
1497   \dim_gset:Nn \g_@@_x_initial_dim
1498   {
1499     \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
1500     \g_@@_x_initial_dim \g_@@_x_final_dim
1501   }
1502   \dim_gset_eq:NN \g_@@_x_final_dim \g_@@_x_initial_dim
1503 }
1504 \@@_draw_tikz_line:
1505 }
1506 }
1507 }

```

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.

```

1508 \cs_new_protected:Nn \@@_draw_Ddots:nn
1509 {
1510   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1511   {
1512     \bool_set_false:N \l_@@_impossible_line_bool
1513     \@@_find_extremities_of_line:nnnn { #1 } { #2 } \c_one_int \c_one_int
1514     \bool_if:NF \l_@@_impossible_line_bool
1515     {
1516       \@@_retrieve_coords:xx
1517       {
1518         \bool_if:NTF \l_@@_initial_open_bool

```

```

1519         { - medium.north~west }
1520         { .south~east }
1521     }
1522     {
1523         \bool_if:NTF \l_@@_final_open_bool
1524         { - medium.south~east }
1525         { .north~west }
1526     }

```

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.

```

1527     \bool_if:NT \l_@@_parallelize_diags_bool
1528     {
1529         \int_incr:N \l_@@_ddots_int

```

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

```

1530         \int_compare:nNnTF \l_@@_ddots_int = \c_one_int

```

If the diagonal line is the first one, we have no adjustment of the line to do but we store the 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.

```

1531         {
1532             \dim_set:Nn \l_@@_delta_x_one_dim
1533             { \g_@@_x_final_dim - \g_@@_x_initial_dim }
1534             \dim_set:Nn \l_@@_delta_y_one_dim
1535             { \g_@@_y_final_dim - \g_@@_y_initial_dim }
1536         }

```

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`.

```

1537         {
1538             \dim_gset:Nn \g_@@_y_final_dim
1539             {
1540                 \g_@@_y_initial_dim +
1541                 ( \g_@@_x_final_dim - \g_@@_x_initial_dim ) *
1542                 \dim_ratio:nn \l_@@_delta_y_one_dim \l_@@_delta_x_one_dim
1543             }
1544         }
1545     }

```

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

```

1546     \@@_draw_tikz_line:
1547 }
1548 }
1549 }

```

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

```

1550 \cs_new_protected:Nn \@@_draw_Iddots:nn
1551 {
1552     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1553     {
1554         \bool_set_false:N \l_@@_impossible_line_bool
1555         \@@_find_extremities_of_line:nnnn { #1 } { #2 } 1 { -1 }
1556         \bool_if:NF \l_@@_impossible_line_bool
1557         {
1558             \@@_retrieve_coords:xx
1559             {
1560                 \bool_if:NTF \l_@@_initial_open_bool
1561                 { - medium.north~east }
1562                 { .south~west }
1563             }
1564             {
1565                 \bool_if:NTF \l_@@_final_open_bool
1566                 { - medium.south~west }

```

```

1567         { .north-east }
1568     }
1569     \bool_if:NT \l_@@_parallelize_diags_bool
1570     {
1571         \int_incr:N \l_@@_iddots_int
1572         \int_compare:nNnTF \l_@@_iddots_int = \c_one_int
1573         {
1574             \dim_set:Nn \l_@@_delta_x_two_dim
1575             { \g_@@_x_final_dim - \g_@@_x_initial_dim }
1576             \dim_set:Nn \l_@@_delta_y_two_dim
1577             { \g_@@_y_final_dim - \g_@@_y_initial_dim }
1578         }
1579         {
1580             \dim_gset:Nn \g_@@_y_final_dim
1581             {
1582                 \g_@@_y_initial_dim +
1583                 ( \g_@@_x_final_dim - \g_@@_x_initial_dim ) *
1584                 \dim_ratio:nn \l_@@_delta_y_two_dim \l_@@_delta_x_two_dim
1585             }
1586         }
1587     }
1588     \@@_draw_tikz_line:
1589 }
1590 }
1591 }

```

The command `\NiceMatrixLastEnv` is not used by the package `nicematrix`. It's only a facility given to the final user. It gives the number of the last environment (in fact the number of the current environment but it's meant to be used after the environment in order to refer to that environment — and its nodes — without having to give it a name).

```

1592 \NewExpandableDocumentCommand \NiceMatrixLastEnv { }
1593 { \int_use:N \g_@@_env_int }

```

17.9 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`.

```

1594 \cs_new_protected:Nn \@@_draw_tikz_line:
1595 {

```

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.

```

1596     \dim_zero_new:N \l_@@_l_dim
1597     \dim_set:Nn \l_@@_l_dim
1598     {
1599         \fp_to_dim:n
1600         {
1601             sqrt
1602             (
1603                 ( \dim_use:N \g_@@_x_final_dim
1604                   - \dim_use:N \g_@@_x_initial_dim
1605                 ) ^ 2
1606                 +
1607                 ( \dim_use:N \g_@@_y_final_dim
1608                   - \dim_use:N \g_@@_y_initial_dim
1609                 ) ^ 2
1610             )
1611         }
1612     }

```

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

```
1613 \dim_compare:nNf \l_@@_l_dim = \c_zero_dim
```

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

```
1614 {
1615   \bool_if:NTF \l_@@_initial_open_bool
1616   {
1617     \bool_if:NTF \l_@@_final_open_bool
1618     {
1619       \int_set:Nn \l_tmpa_int
1620       { \dim_ratio:nn \l_@@_l_dim \l_@@_inter_dots_dim }
1621     }
1622     {
1623       \int_set:Nn \l_tmpa_int
1624       { \dim_ratio:nn { \l_@@_l_dim - 0.3 em } \l_@@_inter_dots_dim }
1625     }
1626   }
1627   {
1628     \bool_if:NTF \l_@@_final_open_bool
1629     {
1630       \int_set:Nn \l_tmpa_int
1631       { \dim_ratio:nn { \l_@@_l_dim - 0.3 em } \l_@@_inter_dots_dim }
1632     }
1633     {
1634       \int_set:Nn \l_tmpa_int
1635       { \dim_ratio:nn { \l_@@_l_dim - 0.6 em } \l_@@_inter_dots_dim }
1636     }
1637   }
}
```

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

```
1638 \dim_set:Nn \l_tmpa_dim
1639 {
1640   ( \g_@@_x_final_dim - \g_@@_x_initial_dim ) *
1641   \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
1642 }
1643 \dim_set:Nn \l_tmpb_dim
1644 {
1645   ( \g_@@_y_final_dim - \g_@@_y_initial_dim ) *
1646   \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
1647 }
```

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`.

```
1648 \int_set:Nn \l_tmpb_int
1649 {
1650   \bool_if:NTF \l_@@_initial_open_bool
1651   { \bool_if:NTF \l_@@_final_open_bool 1 0 }
1652   { \bool_if:NTF \l_@@_final_open_bool 2 1 }
1653 }
```

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.

```
1654 \dim_gadd:Nn \g_@@_x_initial_dim
1655 {
1656   ( \g_@@_x_final_dim - \g_@@_x_initial_dim ) *
1657   \dim_ratio:nn
1658   { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
1659   { \l_@@_l_dim * 2 }
1660   * \l_tmpb_int
1661 }
```

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

```

1662 \dim_gadd:Nn \g_@@_y_initial_dim
1663 {
1664   ( \g_@@_y_final_dim - \g_@@_y_initial_dim ) *
1665   \dim_ratio:nn
1666   { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
1667   { \l_@@_l_dim * 2 } *
1668   \l_tmpb_int
1669 }
1670 \begin { tikzpicture } [ overlay ]
1671 \int_step_inline:nnn 0 \l_tmpa_int
1672 {
1673   \pgfpathcircle
1674   { \pgfpoint { \g_@@_x_initial_dim } { \g_@@_y_initial_dim } }
1675   { \l_@@_radius_dim }
1676   \pgfusepath { fill }
1677   \dim_gadd:Nn \g_@@_x_initial_dim \l_tmpa_dim
1678   \dim_gadd:Nn \g_@@_y_initial_dim \l_tmpb_dim
1679 }
1680 \end { tikzpicture }
1681 }
1682 }

```

17.10 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).

```

1683 \cs_set_eq:NN \@@_ldots \ldots
1684 \cs_set_eq:NN \@@_cdots \cdots
1685 \cs_set_eq:NN \@@_vdots \vdots
1686 \cs_set_eq:NN \@@_ddots \ddots
1687 \cs_set_eq:NN \@@_iddots \iddots

```

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

```

1688 \cs_new_protected:Nn \@@_add_to_empty_cells:
1689 {
1690   \cs_gset:cpx
1691   { @@ _ empty _ \int_use:N \c@iRow - \int_use:N \c@jCol }
1692   { \int_use:N \g_@@_env_int }
1693 }

```

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.

```

1694 \NewDocumentCommand \@@_Ldots { s }
1695 {
1696   \bool_if:nF { #1 } { \@@_instruction_of_type:n { Ldots } }
1697   \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_ldots }
1698   \@@_add_to_empty_cells:
1699 }

1700 \NewDocumentCommand \@@_Cdots { s }
1701 {
1702   \bool_if:nF { #1 } { \@@_instruction_of_type:n { Cdots } }
1703   \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_cdots }
1704   \@@_add_to_empty_cells:
1705 }

```



```

1706 \NewDocumentCommand \@@_Vdots { s }
1707 {
1708   \bool_if:nF { #1 } { \@@_instruction_of_type:n { Vdots } }
1709   \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_vdots }
1710   \@@_add_to_empty_cells:
1711 }

1712 \NewDocumentCommand \@@_Ddots { s }
1713 {
1714   \bool_if:nF { #1 } { \@@_instruction_of_type:n { Ddots } }
1715   \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_ddots }
1716   \@@_add_to_empty_cells:
1717 }

1718 \NewDocumentCommand \@@_Iddots { s }
1719 {
1720   \bool_if:nF { #1 } { \@@_instruction_of_type:n { Iddots } }
1721   \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_iddots }
1722   \@@_add_to_empty_cells:
1723 }

```

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

```

1724 \cs_new_protected:Nn \@@_Hspace:
1725 {
1726   \@@_add_to_empty_cells:
1727   \hspace
1728 }

```

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

```

1729 \cs_set_eq:NN \@@_old_multicolumn \multicolumn
1730 \cs_new:Npn \@@_multicolumn:nnn #1 #2 #3
1731 {
1732   \@@_old_multicolumn { #1 } { #2 } { #3 }
1733   \int_compare:nNnT #1 > 1
1734   {
1735     \seq_gput_left:Nx \g_@@_multicolumn_cells_seq
1736     { \int_eval:n \c@iRow - \int_use:N \c@jCol }
1737     \seq_gput_left:Nn \g_@@_multicolumn_sizes_seq { #1 }
1738   }
1739   \int_gadd:Nn \c@jCol { #1 - 1 }
1740 }

```

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`.

```

1741 \cs_new:Npn \@@_Hdotsfor:
1742 {
1743   \multicolumn { 1 } { C } { }
1744   \@@_Hdotsfor_i
1745 }

```

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:`).

```

1746 \bool_if:NTF \c_@@_draft_bool
1747 {

```

```

1748 \NewDocumentCommand \@@_Hdotsfor_i { 0 { } m }
1749 { \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { C } { } } }
1750 }
1751 {
1752 \NewDocumentCommand \@@_Hdotsfor_i { 0 { } m }
1753 {
1754 \tl_gput_right:Nx \g_@@_Hdotsfor_lines_tl
1755 {
1756 \@@_draw_Hdotsfor:nnn
1757 { \int_use:N \c@iRow }
1758 { \int_use:N \c@jCol }
1759 { #2 }
1760 }
1761 \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { C } { } }
1762 }
1763 }

1764 \cs_new_protected:Nn \@@_draw_Hdotsfor:nnn
1765 {
1766 \bool_set_false:N \l_@@_initial_open_bool
1767 \bool_set_false:N \l_@@_final_open_bool

```

For the row, it's easy.

```

1768 \int_set:Nn \l_@@_initial_i_int { #1 }
1769 \int_set:Nn \l_@@_final_i_int { #1 }

```

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

```

1770 \int_compare:nNnTF #2 = 1
1771 {
1772 \int_set:Nn \l_@@_initial_j_int 1
1773 \bool_set_true:N \l_@@_initial_open_bool
1774 }
1775 {
1776 \int_set:Nn \l_tmpa_int { #2 - 1 }
1777 \@@_if_not_empty_cell:nnTF \l_@@_initial_i_int \l_tmpa_int
1778 { \int_set:Nn \l_@@_initial_j_int { #2 - 1 } }
1779 {
1780 \int_set:Nn \l_@@_initial_j_int {#2}
1781 \bool_set_true:N \l_@@_initial_open_bool
1782 }
1783 }
1784 \int_compare:nNnTF { #2 + #3 - 1 } = \c@jCol
1785 {
1786 \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
1787 \bool_set_true:N \l_@@_final_open_bool
1788 }
1789 {
1790 \int_set:Nn \l_tmpa_int { #2 + #3 }
1791 \@@_if_not_empty_cell:nnTF \l_@@_final_i_int \l_tmpa_int
1792 { \int_set:Nn \l_@@_final_j_int { #2 + #3 } }
1793 {
1794 \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
1795 \bool_set_true:N \l_@@_final_open_bool
1796 }
1797 }
1798 \bool_if:nT { \l_@@_initial_open_bool || \l_@@_final_open_bool }
1799 \@@_create_medium_nodes:
1800 \@@_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).

```

1801 \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
1802 { \cs_set:cpn { @@ _ dotted _ #1 - ##1 } { } }
1803 }

```

17.11 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.

First, we write a command with an argument of the format i - j and applies the command `\int_eval:n` to i and j ; this must *not* be protected (and is, of course fully expandable).³²

```
1804 \cs_new:Npn \@@_double_int_eval:n #1-#2 \q_stop
1805   { \int_eval:n { #1 } - \int_eval:n { #2 } }
```

With the following construction, the command `\@@_double_int_eval:n` is applied to both arguments before the application of `\@@_line_i:nn` (the construction uses the fact the `\@@_line_i:nn` is protected and that `\@@_double_int_eval:n` is fully expandable).

```
1806 \cs_new_protected:Npn \@@_line:nn #1 #2
1807   {
1808     \use:x
1809     {
1810       \@@_line_i:nn
1811       { \@@_double_int_eval:n #1 \q_stop }
1812       { \@@_double_int_eval:n #2 \q_stop }
1813     }
1814   }

1815 \cs_new_protected:Nn \@@_line_i:nn
1816   {
1817     \bool_if:NF \c_@@_draft_bool
1818     {
1819       \dim_zero_new:N \g_@@_x_initial_dim
1820       \dim_zero_new:N \g_@@_y_initial_dim
1821       \dim_zero_new:N \g_@@_x_final_dim
1822       \dim_zero_new:N \g_@@_y_final_dim
1823       \bool_set_false:N \l_@@_initial_open_bool
1824       \bool_set_false:N \l_@@_final_open_bool
1825       \bool_if:nTF
1826       {
1827         \cs_if_exist_p:c { pgf@sh@ns@nm - \int_use:N \g_@@_env_int - #1 }
1828         &&
1829         \cs_if_exist_p:c { pgf@sh@ns@nm - \int_use:N \g_@@_env_int - #2 }
1830       }
1831       {
1832         \begin { tikzpicture }
1833           \path~( #1 )---( #2 )~node[at~start]~( i )~{}~node[at~end]~( f )~{} ;
1834           \tikz@parse@node \pgfutil@firstofone ( i )
1835           \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1836           \dim_gset:Nn \g_@@_y_initial_dim \pgf@y
1837           \tikz@parse@node \pgfutil@firstofone ( f )
1838           \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1839           \dim_gset:Nn \g_@@_y_final_dim \pgf@y
1840         \end { tikzpicture }
1841         \@@_draw_tikz_line:
1842       }
1843       {
1844         \@@_error:nnn { unknown~cell~for~line~in~code~after }
1845         { #1 } { #2 }
1846       }
1847     }
1848   }
```

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).

³²Indeed, we want that the user may use the command `\line` in `code-after` with LaTeX counters in the arguments — with the command `\value`.

17.12 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 position³³ as the line created by `\hline` (or `\hdashline` or `\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.

```

1849 \bool_if:NTF \c_@@_draft_bool
1850 { \cs_set_eq:NN \@@_dotfill: \prg_do_nothing: }
1851 {
1852   \cs_set:Npn \@@_dotfill:
1853     {

```

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.

```

1854   \bool_if:NT \l_@@_small_bool
1855   { \dim_set:Nn \l_@@_inter_dots_dim { 0.25 em } }
1856   \cleaders
1857   \hbox_to_wd:nn
1858   { \l_@@_inter_dots_dim }
1859   {
1860     \c_math_toggle_token
1861     \bool_if:NT \l_@@_small_bool \scriptstyle
1862     \hss . \hss
1863     \c_math_toggle_token
1864   }
1865   \hfill
1866 }
1867 }

```

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

```

1868 \cs_new:Npn \@@_hdottedline:
1869 {
1870   \noalign
1871   {
1872     \bool_gset_true:N \g_@@_large_nodes_bool
1873     \cs_if_exist:cTF { @@_width_ \int_use:N \g_@@_env_int }
1874     { \dim_set_eq:Nc \l_tmpa_dim { @@_width_ \int_use:N \g_@@_env_int } }
1875     { \dim_set:Nn \l_tmpa_dim { 5 mm } }
1876     \hbox_overlap_right:n
1877     {
1878       \bool_if:nT
1879       {
1880         \l_@@_NiceArray_bool
1881         &&
1882         ! \l_@@_exterior_arraycolsep_bool
1883         &&
1884         \int_compare_p:nNn \l_@@_first_col_int > \c_zero_int
1885       }
1886       { \skip_horizontal:n { - \arraycolsep } }
1887     }
1888     \hbox_to_wd:nn
1889     {
1890       \l_tmpa_dim + 2 \arraycolsep
1891       - \l_@@_left_margin_dim - \l_@@_right_margin_dim

```

³³In 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`.

```

1891     }
1892     \@@_dotfill:
1893   }
1894 }
1895 }

```

```

1896 \cs_new_protected:Nn \@@_vdottedline:n
1897 {

```

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

```

1898   \int_compare:nNnTF #1 = \c_zero_int
1899   { \@@_error:n { Use-of~::~in-first~position } }
1900   {
1901     \bool_if:NF \c_@@_draft_bool
1902     {
1903       \dim_zero_new:N \g_@@_x_initial_dim
1904       \dim_zero_new:N \g_@@_y_initial_dim
1905       \dim_zero_new:N \g_@@_x_final_dim
1906       \dim_zero_new:N \g_@@_y_final_dim
1907       \bool_set_true:N \l_@@_initial_open_bool
1908       \bool_set_true:N \l_@@_final_open_bool

```

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

```

1909   \cs_if_exist:cTF
1910   { pgf@sh@ns@nm -\int_use:N \g_@@_env_int - col - #1 }
1911   {
1912     \begin { tikzpicture } [ remember~picture ]
1913       \tikz@parse@node\pgfutil@firstofone
1914       ( col - #1 )
1915       \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1916       \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1917       \dim_gset:Nn \g_@@_y_final_dim \pgf@y
1918     \end { tikzpicture }
1919     \dim_gset:Nn \g_@@_y_initial_dim { - \c_max_dim }
1920     \int_step_inline:nn \c@jCol
1921     {
1922       \begin { tikzpicture } [ remember~picture ]
1923         \tikz@parse@node\pgfutil@firstofone
1924         ( 1 - #1 . north~east )
1925         \dim_gset:Nn \g_@@_y_initial_dim
1926         { \dim_max:nn \g_@@_y_initial_dim \pgf@y }
1927       \end { tikzpicture }
1928     }
1929   }

```

If not, we use the “large node”.

```

1930   {
1931     \begin { tikzpicture } [ remember~picture ]
1932       \tikz@parse@node\pgfutil@firstofone
1933       ( 1 - #1 - large .north~east )
1934       \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1935       \dim_gset:Nn \g_@@_y_initial_dim \pgf@y
1936       \tikz@parse@node\pgfutil@firstofone
1937       ( \int_use:N \c@iRow - #1 - large .south~east )
1938       \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1939       \dim_gset:Nn \g_@@_y_final_dim \pgf@y
1940     \end { tikzpicture }

```

However, if the previous column was constructed with a letter w, we use the w-nodes (and we erase the previous computation of the x -value of the vertical dotted line).

```

1941   \cs_if_exist:cT
1942   { pgf@sh@ns@nm -\int_use:N \g_@@_env_int - 1 - #1 - w }
1943   {

```

```

1944         \begin { tikzpicture } [ remember~picture ]
1945             \tikz@parse@node\pgfutil@firstofone
1946             ( 1 - #1 - w .north-east )
1947             \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1948             \tikz@parse@node\pgfutil@firstofone
1949             ( \int_use:N \c@iRow - #1 - w .south-east )
1950             \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1951         \end { tikzpicture }
1952         \dim_gadd:Nn \g_@@_x_initial_dim \arraycolsep
1953         \dim_gadd:Nn \g_@@_x_final_dim \arraycolsep
1954     }
1955 }
1956 \@@_draw_tikz_line:
1957 }
1958 }
1959 }

```

17.13 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:nNf \c@iRow = \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).

```

1960 \cs_new_protected:Npn \@@_vline:
1961 {
1962     \int_compare:nNfTF \l_@@_first_col_int = \c_zero_int
1963     {
1964         \int_compare:nNfTF \c@jCol = \c_zero_int
1965         {
1966             \int_compare:nNfTF \l_@@_first_row_int = \c_zero_int
1967             {
1968                 \int_compare:nNf \c@iRow = \c_zero_int
1969                 {
1970                     \int_compare:nNf \c@iRow = \l_@@_last_row_int
1971                     \@@_vline_i:
1972                 }
1973             }
1974         }
1975         \int_compare:nNfTF \c@iRow = \c_zero_int
1976         {
1977             \int_compare:nNf \c@iRow = \l_@@_last_row_int
1978             \@@_vline_i:
1979         }
1980     }
1981 }
1982 {
1983     \int_compare:nNf \c@iRow = \c_zero_int
1984     {
1985         \int_compare:nNf \c@iRow = \l_@@_last_row_int
1986         \@@_vline_i:
1987     }

```

```

1988     }
1989   }
1990   {
1991     \int_compare:nNnTF \c@jCol = \c_zero_int
1992     {
1993       \int_compare:nNnF \c@iRow = { -1 }
1994       {
1995         \int_compare:nNnF \c@iRow = { \l_@@_last_row_int - 1 }
1996         \@@_vline_i:
1997       }
1998     }
1999   {
2000     \int_compare:nNnF \c@iRow = \c_zero_int
2001     {
2002       \int_compare:nNnF \c@iRow = \l_@@_last_row_int
2003       \@@_vline_i:
2004     }
2005   }
2006 }
2007 }

```

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

```

2008 \cs_set_eq:NN \@@_vline_i: \vline

```

We give now the definition of `\OnlyMainNiceMatrix`. Internally, it is not used by `nicematrix`. It's only a facility given to the final user, which may be useful in the definitions of new columns types (with `\newcolumnstype`).

First, we give the definition of `\OnlyMainNiceMatrix` in the general case: it's no-op (thus, a definition of column type may be used outside the environments of `nicematrix`, in `{array}`, etc.).

```

2009 \cs_set_eq:NN \OnlyMainNiceMatrix \use:n

```

Now, we give the definition of `\OnlyMainNiceMatrix` which will be used in the environments of `nicematrix`. This command `\@@_OnlyMainNiceMatrix:n` will be linked to `\OnlyMainNiceMatrix` in `\@@_pre_array:.` This command is “fully expandable” and that's why we have not protected it, even though this characteristic will probably not be used.

```

2010 \cs_new:Npn \@@_OnlyMainNiceMatrix:n #1
2011 {
2012   \int_compare:nNnF \c@iRow = \c_zero_int
2013   { \int_compare:nNnF \c@iRow = \l_@@_last_row_int { #1 } }
2014 }

```

17.14 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.

```

2015 \bool_new:N \l_@@_block_auto_columns_width_bool

```

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

```

2016 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
2017 {
2018   auto-columns-width .code:n =
2019   {
2020     \bool_set_true:N \l_@@_block_auto_columns_width_bool
2021     \dim_gzero_new:N \g_@@_max_cell_width_dim
2022     \bool_set_true:N \l_@@_auto_columns_width_bool
2023   }
2024 }

```

```

2025 \NewDocumentEnvironment { NiceMatrixBlock } { ! O { } }
2026 {
2027   \int_gincr:N \g_@@_NiceMatrixBlock_int
2028   \dim_zero:N \l_@@_columns_width_dim
2029   \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
2030   \bool_if:NT \l_@@_block_auto_columns_width_bool
2031   {
2032     \cs_if_exist:cT { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
2033     {
2034       \dim_set:Nx \l_@@_columns_width_dim
2035       { \use:c { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int } }
2036     }
2037   }
2038 }

```

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).

```

2039 {
2040   \bool_if:NT \l_@@_block_auto_columns_width_bool
2041   {
2042     \iow_now:Nn \@mainaux \ExplSyntaxOn
2043     \iow_now:Nx \@mainaux
2044     {
2045       \cs_gset:cpn
2046       { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
2047       { \dim_use:N \g_@@_max_cell_width_dim }
2048     }
2049     \iow_now:Nn \@mainaux \ExplSyntaxOff
2050   }
2051 }

```

17.15 The extra nodes

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

```

2052 \cs_generate_variant:Nn \dim_min:nn { v n }
2053 \cs_generate_variant:Nn \dim_max:nn { v n }

```

We have three macros of creation of nodes: \@@_create_medium_nodes:, \@@_create_large_nodes: and \@@_create_medium_and_large_nodes:. They must *not* be used in the code-after because the code-after is executed in a scope of prefix name of Tikz.

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

The command \@@_computations_for_medium_nodes: must be used in a {tikzpicture}. For each row i , we compute two dimensions $l_@@_row_i_min_dim$ and $l_@@_row_i_max_dim$. The dimension $l_@@_row_i_min_dim$ is the minimal y -value of all the cells of the row i . The dimension $l_@@_row_i_max_dim$ is the maximal y -value of all the cells of the row i . Similarly, for each column j , we compute two dimensions $l_@@_column_j_min_dim$ and $l_@@_column_j_max_dim$. The dimension $l_@@_column_j_min_dim$ is the minimal x -value of all the cells of the column j . The dimension $l_@@_column_j_max_dim$ is the maximal x -value of all the cells of the column j . Since these dimensions will be computed as maximum or minimum, we initialize them to \c_max_dim or -\c_max_dim.

```

2054 \cs_new_protected:Npn \@@_computations_for_medium_nodes:
2055 {
2056   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:

```



```

2057 {
2058   \dim_zero_new:c { l_@@_row\_@@_i: _min_dim }
2059   \dim_set_eq:cN { l_@@_row\_@@_i: _min_dim } \c_max_dim
2060   \dim_zero_new:c { l_@@_row\_@@_i: _max_dim }
2061   \dim_set:cn { l_@@_row\_@@_i: _max_dim } { - \c_max_dim }
2062 }
2063 \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
2064 {
2065   \dim_zero_new:c { l_@@_column\_@@_j: _min_dim }
2066   \dim_set_eq:cN { l_@@_column\_@@_j: _min_dim } \c_max_dim
2067   \dim_zero_new:c { l_@@_column\_@@_j: _max_dim }
2068   \dim_set:cn { l_@@_column\_@@_j: _max_dim } { - \c_max_dim }
2069 }

```

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

```

2070 \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
2071 {
2072   \int_step_variable:nnNn
2073     \l_@@_first_col_int \g_@@_col_total_int \@@_j:

```

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

```

2074   { \cs_if_exist:cT
2075     { pgf@sh@ns@nm - \int_use:N \g_@@_env_int - \@@_i: - \@@_j: }

```

We retrieve the coordinates of the anchor south west of the (normal) node of the cell (i - j). They will be stored in `\pgf@x` and `\pgf@y`.

```

2076   {
2077     \tikz@parse@node \pgfutil@firstofone
2078       ( nm - \int_use:N \g_@@_env_int
2079         - \@@_i: - \@@_j: .south-west )
2080     \dim_set:cn { l_@@_row\_@@_i: _min_dim }
2081     { \dim_min:vn { l_@@_row _ \@@_i: _min_dim } \pgf@y }
2082     \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
2083     {
2084       \dim_set:cn { l_@@_column _ \@@_j: _min_dim }
2085       { \dim_min:vn { l_@@_column _ \@@_j: _min_dim } \pgf@x }
2086     }

```

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`.

```

2087     \tikz@parse@node \pgfutil@firstofone
2088       ( nm - \int_use:N \g_@@_env_int - \@@_i: - \@@_j: .north-east )
2089     \dim_set:cn { l_@@_row _ \@@_i: _max_dim }
2090     { \dim_max:vn { l_@@_row _ \@@_i: _max_dim } \pgf@y }
2091     \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
2092     {
2093       \dim_set:cn { l_@@_column _ \@@_j: _max_dim }
2094       { \dim_max:vn { l_@@_column _ \@@_j: _max_dim } \pgf@x }
2095     }
2096   }
2097 }
2098 }
2099 }

```

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

```

2100 \cs_new_protected:Npn \@@_create_medium_nodes:
2101 {
2102   \begin { tikzpicture } [ remember-picture , overlay ]
2103     \@@_computations_for_medium_nodes:

```

Now, we can create the “medium nodes”. We use a command `\@@_create_nodes:` because this command will also be used for the creation of the “large nodes” (after changing the value of `name-suffix`).

```

2104 \tikzset { name~suffix = -medium }
2105 \@@_create_nodes:
2106 \end { tikzpicture }
2107 \cs_set_protected:Npn \@@_create_medium_nodes: { }
2108 \cs_set_protected:Npn \@@_create_medium_and_large_nodes:
2109 { \@@_create_large_nodes: }
2110 }

```

The command `\@@_create_large_nodes:` must be used when we want to create only the “large nodes” and not the medium ones (if we want to create both, we have to use the command `\@@_create_medium_and_large_nodes:`). However, the computation of the mathematical coordinates of the “large nodes” needs the computation of the mathematical coordinates of the “medium nodes”. That’s why we use first `\@@_computations_for_medium_nodes:` and then the command `\@@_computations_for_large_nodes:`.

```

2111 \cs_new_protected:Npn \@@_create_large_nodes:
2112 {
2113 \begin { tikzpicture } [ remember~picture , overlay ]
2114 \@@_computations_for_medium_nodes:
2115 \@@_computations_for_large_nodes:
2116 \tikzset { name~suffix = -large }
2117 \@@_create_nodes:
2118 \end { tikzpicture }
2119 \@@_compute_width_of_array:
2120 \cs_set_protected:Npn \@@_create_large_nodes: { }
2121 \cs_set_protected:Npn \@@_create_medium_and_large_nodes:
2122 { \@@_create_medium_nodes: }
2123 }
2124 \cs_new_protected:Npn \@@_create_medium_and_large_nodes:
2125 {
2126 \begin { tikzpicture } [ remember~picture , overlay ]
2127 \@@_computations_for_medium_nodes:
2128 % Now, we can create the ``medium nodes''. We use a command |\@@_create_nodes:|
2129 % because this command will also be used for the creation of the ``large nodes''
2130 % (after changing the value of |name-suffix|).
2131 % \begin{macrocode}
2132 \tikzset { name~suffix = -medium }
2133 \@@_create_nodes:
2134 \@@_computations_for_large_nodes:
2135 \tikzset { name~suffix = -large }
2136 \@@_create_nodes:
2137 \@@_compute_width_of_array:
2138 \end { tikzpicture }
2139 \@@_compute_width_of_array:
2140 \cs_set_protected:Npn \@@_create_medium_and_large_nodes: { }
2141 \cs_set_protected:Npn \@@_create_medium_nodes: { }
2142 \cs_set_protected:Npn \@@_create_large_nodes: { }
2143 }

```

For “large nodes”, the exterior rows and columns don’t interfere. That’s why the loop over the columns will start at 1 and stop at `\c@jCol` (and not `\g_@@_col_total_int`). Idem for the rows.

```

2144 \cs_new_protected:Npn \@@_computations_for_large_nodes:
2145 {
2146 \int_set:Nn \l_@@_first_row_int 1
2147 \int_set:Nn \l_@@_first_col_int 1

```

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

```

2148 \int_step_variable:nNn { \c@iRow - 1 } \@@_i:
2149 {
2150 \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim }

```

```

2151     {
2152     (
2153         \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } +
2154         \dim_use:c { l_@@_row _ \int_eval:n { \@@_i: + 1 } _ max _ dim }
2155     )
2156     / 2
2157     }
2158     \dim_set_eq:cc { l_@@_row _ \int_eval:n { \@@_i: + 1 } _ max _ dim }
2159     { l_@@_row_ \@@_i: _ min_dim }
2160 }
2161 \int_step_variable:nNn { \c@jCol - 1 } \@@_j:
2162 {
2163     \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim }
2164     {
2165     (
2166         \dim_use:c
2167         { l_@@_column _ \@@_j: _ max _ dim } +
2168         \dim_use:c
2169         { l_@@_column _ \int_eval:n { \@@_j: + 1 } _ min _ dim }
2170     )
2171     / 2
2172     }
2173     \dim_set_eq:cc { l_@@_column _ \int_eval:n { \@@_j: + 1 } _ min _ dim }
2174     { l_@@_column _ \@@_j: _ max _ dim }
2175 }
2176 \dim_sub:cn
2177 { l_@@_column _ 1 _ min _ dim }
2178 \l_@@_left_margin_dim
2179 \dim_add:cn
2180 { l_@@_column _ \int_use:N \c@jCol _ max _ dim }
2181 \l_@@_right_margin_dim
2182 }

```

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.

```

2183 \cs_new_protected:Npn \@@_create_nodes:
2184 {
2185     \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
2186     {
2187         \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:

```

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

```

2188     {
2189         \coordinate ( @@~south-west )
2190         at ( \dim_use:c { l_@@_column _ \@@_j: _ min_dim } ,
2191             \dim_use:c { l_@@_row _ \@@_i: _ min_dim } ) ;
2192         \coordinate ( @@~north-east )
2193         at ( \dim_use:c { l_@@_column _ \@@_j: _ max_dim } ,
2194             \dim_use:c { l_@@_row _ \@@_i: _ max_dim } ) ;

```

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

```

2195     \node
2196     [
2197         node~contents = { } ,
2198         fit = ( @@~south-west ) ( @@~north-east ) ,

```

```

2199         inner~sep = \c_zero_dim ,
2200         name = nm - \int_use:N \g_@@_env_int - \@@_i: - \@@_j: ,
2201         alias =
2202             \str_if_empty:NF \l_@@_name_str
2203             { \l_@@_name_str - \@@_i: - \@@_j: }
2204     ]
2205     ;
2206 }
2207 }

```

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 .

```

2208     \seq_mapthread_function:NNN
2209     \g_@@_multicolumn_cells_seq
2210     \g_@@_multicolumn_sizes_seq
2211     \@@_node_for_multicolumn:nn
2212 }

```

We can now compute the width of the array (used by `\hdottedline`). We should modify this point because it's a waste to construct all the “large nodes” only for computing the width of the array.

```

2213 \cs_new_protected:Npn \@@_compute_width_of_array:
2214 {
2215     \begin { tikzpicture } [ remember~picture , overlay ]
2216     \tikz@parse@node \pgfutil@firstofone
2217     ( nm - \int_use:N \g_@@_env_int - 1 - 1 - large .north~west )
2218     \dim_gset:Nn \g_tmpa_dim \pgf@x
2219     \tikz@parse@node \pgfutil@firstofone
2220     ( nm - \int_use:N \g_@@_env_int - 1 -
2221       \int_use:N \c_jCol - large .north~east )
2222     \dim_gset:Nn \g_tmpb_dim \pgf@x
2223     \end { tikzpicture }
2224     \iow_now:Nn \@mainaux \ExplSyntaxOn
2225     \iow_now:Nx \@mainaux
2226     {
2227         \cs_gset:cpn { @@_width_ \int_use:N \g_@@_env_int }
2228         { \dim_eval:n { \g_tmpb_dim - \g_tmpa_dim } }
2229     }
2230     \iow_now:Nn \@mainaux \ExplSyntaxOff
2231 }

```

```

2232 \cs_new_protected:Npn \@@_extract_coords: #1 - #2 \q_stop
2233 {
2234     \cs_set:Npn \@@_i: { #1 }
2235     \cs_set:Npn \@@_j: { #2 }
2236 }

```

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”).

```

2237 \cs_new_protected:Npn \@@_node_for_multicolumn:nn
2238 {
2239     \@@_extract_coords: #1 \q_stop
2240     \coordinate ( @@~south~west ) at
2241     (
2242         \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } ,
2243         \dim_use:c { l_@@_row _ \@@_i: _ min _ dim }
2244     ) ;
2245     \coordinate ( @@~north~east ) at
2246     (
2247         \dim_use:c { l_@@_column _ \int_eval:n { \@@_j: + #2 - 1 } _ max _ dim } ,
2248         \dim_use:c { l_@@_row _ \@@_i: _ max _ dim }
2249     ) ;

```

```

2250 \node
2251 [
2252   node~contents = { } ,
2253   fit = ( @@~south~west ) ( @@~north~east ) ,
2254   inner~sep = \c_zero_dim ,
2255   name = nm - \int_use:N \g_@@_env_int - \@@_i: - \@@_j: ,
2256   alias =
2257     \str_if_empty:NF \l_@@_name_str
2258     { \l_@@_name_str - \@@_i: - \@@_j: }
2259 ]
2260 ;
2261 }

```

17.16 Block matrices

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

The following command will be linked to `\Block` in the environments of `nicematrix`. We define it with `\NewDocumentCommand` of `xparse` because it has an optionnal argument between `<` and `>` (for TeX instructions put before the math mode of the label)

```

2262 \NewDocumentCommand \@@_Block: { m D < > { } m }
2263 { \@@_Block_i #1 \q_stop { #2 } { #3 } }

```

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.

```

2264 \cs_new:Npn \@@_Block_i #1-#2 \q_stop { \@@_Block_ii:nnnn { #1 } { #2 } }

```

Now, the arguments have been extracted: `#1` is *i* (the number of rows of the block), `#2` is *j* (the number of columns of the block), `#3` 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).

```

2265 \cs_new:Npn \@@_Block_ii:nnnn #1 #2 #3 #4
2266 {

```

In the case of a block of only one row, we create a special node of shape `coordinate` in order to remember the *y*-value of the baseline of the current row.

```

2267   \int_compare:nNnT { #1 } = 1
2268   {
2269     \begin { tikzpicture } [ remember~picture , baseline ]
2270     \coordinate
2271       ( nm - \int_use:N \g_@@_env_int
2272         - Block
2273         - \int_use:N \c@iRow
2274         - \int_use:N \c@jCol ) ;
2275     \end { tikzpicture }
2276   }
2277   \bool_gset_true:N \g_@@_medium_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).

```

2278   \tl_gput_left:Nx \g_@@_code_after_tl
2279   {
2280     \@@_Block_iii:nnnnn
2281     { \int_use:N \c@iRow }
2282     { \int_use:N \c@jCol }
2283     { \int_eval:n { \c@iRow + #1 - 1 } }
2284     { \int_eval:n { \c@jCol + #2 - 1 } }
2285     \exp_not:n { { #3 $ #4 $ } }
2286   }
2287 }

```

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

```

2288 \cs_new_protected:Npn \@@_Block_iii:nnnnn #1 #2 #3 #4 #5
2289 {
2290   \bool_if:nTF
2291     {
2292       \int_compare_p:nNn { #3 } > \c@iRow
2293       || \int_compare_p:nNn { #4 } > \c@jCol
2294     }
2295     { \msg_error:nnnn { nicematrix } { Block-too-large } { #1 } { #2 } }
2296     {

```

If the block has only one row, we have to do a special work in order to have the contents of the node aligned with the contents of the other rows of the array.

```

2297       \int_compare:nNnTF { #1 } = { #3 }
2298       {
2299         \begin { tikzpicture }

```

First, we compute in `\l_tmpa_dim` the y -value of the baseline of the row. We have constructed a special node of shape `coordinate` in this order.

```

2300         \tikz@parse@node \pgfutil@firstofone (Block-#1-#2)
2301         \dim_set:Nn \l_tmpa_dim \pgf@y
2302         \node
2303           [
2304             fit = ( #1 - #2 - medium . north-west )
2305                  ( #3 - #4 - medium . south-east ) ,
2306             inner-sep = 0 pt ,
2307           ]
2308         (#1-#2) { } ;

```

With the following instruction, we retrieve the x -value and the y -value of the center of the block. We will only use the x -value, available in `\pgf@x`.

```

2309         \tikz@parse@node \pgfutil@firstofone (#1-#2)
2310         \path (\pgf@x,\l_tmpa_dim) node [ anchor = base ] { #5 } ;
2311       \end { tikzpicture }
2312     }

```

If the number of rows is different of 1, it's necessary to create two Tikz nodes because we want the label `#5` really drawn in the *center* of the node.

```

2313     {
2314       \begin { tikzpicture }
2315       \node
2316         [
2317           fit = ( #1 - #2 - medium . north-west )
2318                ( #3 - #4 - medium . south-east ) ,
2319           inner-sep = 0 pt ,
2320         ]

```

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

```

2321         (#1-#2) { } ;
2322         \node at (#1-#2.center) { #5 } ;
2323       \end { tikzpicture }
2324     }
2325   }
2326 }

```

17.17 How to draw the dotted lines transparently

```

2327 \cs_set_protected:Npn \@@_renew_matrix:
2328 {
2329   \RenewDocumentEnvironment { pmatrix } { } {
2330     { \pNiceMatrix }
2331     { \endpNiceMatrix }
2332   } \RenewDocumentEnvironment { vmatrix } { } {
2333     { \vNiceMatrix }

```

```

2334     { \endvNiceMatrix }
2335 \RenewDocumentEnvironment { Vmatrix } { }
2336     { \VNiceMatrix }
2337     { \endVNiceMatrix }
2338 \RenewDocumentEnvironment { bmatrix } { }
2339     { \bNiceMatrix }
2340     { \endbNiceMatrix }
2341 \RenewDocumentEnvironment { Bmatrix } { }
2342     { \BNiceMatrix }
2343     { \endBNiceMatrix }
2344 }

```

17.18 Automatic arrays

```

2345 \cs_new_protected:Npn \@@_set_size:n #1-#2 \q_stop
2346 {
2347   \int_set:Nn \l_@@_nb_rows_int { #1 }
2348   \int_set:Nn \l_@@_nb_cols_int { #2 }
2349 }
2350 \NewDocumentCommand \AutoNiceMatrixWithDelims { m m O { } m O { } m ! O { } }
2351 {
2352   \int_zero_new:N \l_@@_nb_rows_int
2353   \int_zero_new:N \l_@@_nb_cols_int
2354   \@@_set_size:n #4 \q_stop
2355   \begin { NiceArrayWithDelims } { #1 } { #2 }
2356     { * { \l_@@_nb_cols_int } { C } } [ #3 , #5 , #7 ]
2357   \int_compare:nNnT \l_@@_first_row_int = \c_zero_int
2358     {
2359       \int_compare:nNnT \l_@@_first_col_int = \c_zero_int { & }
2360       \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
2361       \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
2362     }
2363   \prg_replicate:nn \l_@@_nb_rows_int
2364   {
2365     \int_compare:nNnT \l_@@_first_col_int = \c_zero_int { & }

```

You put { } before #6 to avoid a hasty expansion of an eventual `\arabic{iRow}` at the beginning of the row which would result in an incorrect value of that `iRow` (since `iRow` is incremented in the first cell of the row of the `\halign`).

```

2366     \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { { } #6 & } #6
2367     \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
2368   }
2369   \int_compare:nNnT \l_@@_last_row_int > { -2 }
2370   {
2371     \int_compare:nNnT \l_@@_first_col_int = \c_zero_int { & }
2372     \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
2373     \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
2374   }
2375   \end { NiceArrayWithDelims }
2376 }
2377 \cs_set_protected:Npn \@@_define_com:nnn #1 #2 #3
2378 {
2379   \cs_set_protected:cpn { #1 AutoNiceMatrix }
2380   {
2381     \str_gset:Nx \g_@@_type_env_str
2382     { command ~ \c_backslash_str #1 AutoNiceMatrix }
2383     \AutoNiceMatrixWithDelims { #2 } { #3 }
2384   }
2385 }
2386 \@@_define_com:nnn p ( )
2387 \@@_define_com:nnn b [ ]
2388 \@@_define_com:nnn v | |
2389 \@@_define_com:nnn V \ | \ |

```

```
2390 \@@_define_com:nnn B \{ \}
```

17.19 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.

```
2391 \keys_define:nn { NiceMatrix / Package }
2392 {
2393   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
2394   renew-dots .value_forbidden:n = true ,
2395   renew-matrix .code:n = \@@_renew_matrix: ,
2396   renew-matrix .value_forbidden:n = true ,
2397   transparent .meta:n = { renew-dots , renew-matrix } ,
2398   transparent .value_forbidden:n = true,
2399   obsolete-environments .code:n =
2400     \@@_msg_redirect_name:nn { Obsolete-environment } { none }
2401 }
2402 \ProcessKeysOptions { NiceMatrix / Package }
```

17.20 Error messages of the package

```
2403 \@@_msg_new:nn { unknown-cell-for-line-in-code-after }
2404 {
2405   Your-command-\token_to_str:N\line\{#1\}\{#2\}-in-the-'code-after'-
2406   can't-be-executed-because-a-Tikz-node-doesn't-exist.\\
2407   If-you-go-on-this-command-will-be-ignored.
2408 }
2409 \@@_msg_new:nn { last-col-non-empty-for-NiceArray }
2410 {
2411   In-the-\g_@@_type_env_str,~you-must-use-the-option~
2412   'last-col'~without-value.\\
2413   However,~you-can-go-on-for-this-time~
2414   (the-value-\l_keys_value_tl~will-be-ignored).
2415 }
2416 \@@_msg_new:nn { last-col-empty-for-NiceMatrix }
2417 {
2418   In-the-\g_@@_type_env_str, you-can't-use-the-option~
2419   'last-col'~without-value.~You-must-give-the-number-of-that-last-column.\\
2420   If-you-go-on-this-option-will-be-ignored.
2421 }
2422 \@@_msg_new:nn { Block-too-large }
2423 {
2424   You-try-to-draw-a-block-in-the-cell~#1-#2-of-your-matrix-but-the-matrix-is~
2425   too-small-for-that-block.\\
2426   If-you-go-on,~this-command-will-be-ignored.
2427 }
2428 \@@_msg_new:nn { Impossible-line }
2429 {
2430   A-dotted-line-can't-be-drawn-because-you-have-not-put~
2431   all-the-ampersands-required-on-the-row-#1.\\
2432   If-you-go-on,~this-dotted-line-will-be-ignored.
2433 }
2434 \@@_msg_new:nn { Wrong-last-row }
2435 {
2436   You-have-used-'last-row=\int_use:N \l_@@_last_row_int'~but-your~
2437   \g_@@_type_env_str\ seems-to-have-\int_use:N \c@iRow \ rows.~
2438   If-you-go-on,~the-value-of~\int_use:N \c@iRow \ will-be-used-for~
```



```

2439     last~row~.~You~can~avoid~this~problem~by~using~'last~row'~
2440     without~value~(more~compilations~might~be~necessary).
2441 }
2442 \@@_msg_new:nn { Yet~in~env }
2443 {
2444     Environments~\{NiceArray\}~(or~\{NiceMatrix\},~etc.)~can't~be~nested.\\
2445     This~error~is~fatal.
2446 }
2447 \@@_msg_new:nn { Outside~math~mode }
2448 {
2449     The~\g_@@_type_env_str\ can~be~used~only~in~math~mode~
2450     (and~not~in~\token_to_str:N \vcenter).\\
2451     This~error~is~fatal.
2452 }
2453 \@@_msg_new:nn { Option~Transparent~suppressed }
2454 {
2455     The~option~'Transparent'~has~been~renamed~'transparent'.\\
2456     However,~you~can~go~on~for~this~time.
2457 }
2458 \@@_msg_new:nn { Option~RenewMatrix~suppressed }
2459 {
2460     The~option~'RenewMatrix'~has~been~renamed~'renew-matrix'.\\
2461     However,~you~can~go~on~for~this~time.
2462 }
2463 \@@_msg_new:nn { Bad~value~for~letter~for~dotted~lines }
2464 {
2465     The~value~of~key~'\tl_use:N\l_keys_key_tl'~must~be~of~length~1.\\
2466     If~you~go~on,~it~will~be~ignored.
2467 }
2468 \@@_msg_new:nnn { Unknown~key~for~NiceMatrixOptions }
2469 {
2470     The~key~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~command~
2471     \token_to_str:N \NiceMatrixOptions. \\
2472     If~you~go~on,~it~will~be~ignored. \\
2473     For~a~list~of~the~available~keys,~type~H~<return>.
2474 }
2475 {
2476     The~available~options~are~(in~alphabetic~order):~
2477     allow~duplicate~names,~
2478     code~for~first~col,~
2479     code~for~first~row,~
2480     code~for~last~col,~
2481     code~for~last~row,~
2482     create~extra~nodes,~
2483     create~medium~nodes,~
2484     create~large~nodes,~
2485     exterior~arraycolsep,~
2486     hlines,~
2487     left~margin,~
2488     letter~for~dotted~lines,~
2489     nullify~dots,~
2490     parallelize~diags,~
2491     renew~dots,~
2492     renew~matrix,~
2493     right~margin,~
2494     small~
2495     and~transparent
2496 }
2497 \@@_msg_new:nnn { Unknown~option~for~NiceArray }
2498 {
2499     The~option~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~environment~

```

```

2500 \{NiceArray\}. \\
2501 If-you-go-on,~it~will~be~ignored. \\
2502 For~a~list~of~the~available~options,~type~H~<return>.
2503 }
2504 {
2505 The~available~options~are~(in~alphabetic~order):~
2506 b,~
2507 c,~
2508 code-after,~
2509 code-for-first-col,~
2510 code-for-first-row,~
2511 code-for-last-col,~
2512 code-for-last-row,~
2513 columns-width,~
2514 create-extra-nodes,~
2515 create-medium-nodes,~
2516 create-large-nodes,~
2517 extra-left-margin,~
2518 extra-right-margin,~
2519 first-col,~
2520 first-row,~
2521 hlines,~
2522 last-col,~
2523 last-row,~
2524 left-margin,~
2525 name,~
2526 nullify-dots,~
2527 parallelize-diags,~
2528 renew-dots,~
2529 right-margin,~
2530 small~
2531 and~t.
2532 }

```

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).

```

2533 \@@_msg_new:nnn { Unknown~option~for~NiceMatrix }
2534 {
2535 The~option~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~
2536 \g_@@_type_env_str. \\
2537 If-you-go-on,~it~will~be~ignored. \\
2538 For~a~list~of~the~available~options,~type~H~<return>.
2539 }
2540 {
2541 The~available~options~are~(in~alphabetic~order):~
2542 code-after,~
2543 code-for-first-col,~
2544 code-for-first-row,~
2545 code-for-last-col,~
2546 code-for-last-row,~
2547 columns-width,~
2548 create-extra-nodes,~
2549 create-medium-nodes,~
2550 create-large-nodes,~
2551 extra-left-margin,~
2552 extra-right-margin,~
2553 first-col,~
2554 first-row,~
2555 hlines,~
2556 last-col,~
2557 last-row,~
2558 left-margin,~
2559 name,~
2560 nullify-dots,~

```

```

2561 parallelize-diags,~
2562 renew-dots,~
2563 right-margin~
2564 and~small.
2565 }

```

The following message should be changed because, normally, there can be any longer artefact in the environments of amsmath.

```

2566 \@@_msg_new:nnn { Duplicate-name }
2567 {
2568   The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
2569   the~same~environment~name~twice.~You~can~go~on,~but,~
2570   maybe,~you~will~have~incorrect~results~especially~
2571   if~you~use~'columns-width=auto'.~If~you~use~nicematrix~inside~some~
2572   environments~of~amsmath,~this~error~may~be~an~artefact.~In~this~case,~
2573   use~the~option~'allow-duplicate-names'.\\
2574   For~a~list~of~the~names~already~used,~type~H~<return>. \\
2575 }
2576 {
2577   The~names~already~defined~in~this~document~are:~
2578   \seq_use:Nnnn \g_@@_names_seq { ,~ } { ,~ } { ~and~ }.
2579 }
2580 \@@_msg_new:nn { Option-auto-for-columns-width }
2581 {
2582   You~can't~give~the~value~'auto'~to~the~option~'columns-width'~here.~
2583   If~you~go~on,~the~option~will~be~ignored.
2584 }
2585 \@@_msg_new:nn { Zero~row }
2586 {
2587   There~is~a~problem.~Maybe~your~\g_@@_type_env_str\ is~empty.~
2588   Maybe~you~have~used~l,~c~and~r~instead~of~L,~C~and~R~in~the~preamble~
2589   of~your~environment. \\
2590   If~you~go~on,~the~result~may~be~incorrect.
2591 }
2592 \@@_msg_new:nn { Use-of-:~in-first-position }
2593 {
2594   You~can't~use~the~column~specifier~'\l_@@_letter_for_dotted_lines_str'~in~the~
2595   first~position~of~the~preamble~of~the~\g_@@_type_env_str. \\
2596   If~you~go~on,~this~dotted~line~will~be~ignored.
2597 }

```

17.21 Obsolete environments

```

2598 \@@_msg_new:nn { Obsolete~environment }
2599 {
2600   The~environment~\{@currentenv\}~is~obsolete.~We~should~use~#1~instead.~
2601   However,~you~can~go~on~for~this~time.~
2602   If~you~don't~want~to~see~this~error~again,~you~should~load~'nicematrix'~
2603   with~the~option~'obsolete-environments'.
2604 }
2605 \NewDocumentEnvironment { pNiceArrayC } { }
2606 {
2607   \@@_error:nn { Obsolete~environment }
2608     { the~option~'last-col' }
2609   \int_set:Nn \l_@@_last_col_int \c_zero_dim
2610   \pNiceArray
2611 }
2612 { \endpNiceArray }
2613 \NewDocumentEnvironment { bNiceArrayC } { }
2614 {
2615   \@@_error:nn { Obsolete~environment }
2616     { the~option~'last-col' }
2617   \int_set:Nn \l_@@_last_col_int \c_zero_dim

```

```

2618     \bNiceArray
2619 }
2620 { \endbNiceArray }

2621 \NewDocumentEnvironment { BNiceArrayC } { }
2622 {
2623     \@@_error:nn { Obsolete-environment }
2624     { the~option~'last-col' }
2625     \int_set:Nn \l_@@_last_col_int \c_zero_dim
2626     \BNiceArray
2627 }
2628 { \endBNiceArray }

2629 \NewDocumentEnvironment { vNiceArrayC } { }
2630 {
2631     \@@_error:nn { Obsolete-environment }
2632     { the~option~'last-col' }
2633     \int_set:Nn \l_@@_last_col_int \c_zero_dim
2634     \vNiceArray
2635 }
2636 { \endvNiceArray }

2637 \NewDocumentEnvironment { VNiceArrayC } { }
2638 {
2639     \@@_error:nn { Obsolete-environment }
2640     { the~option~'last-col' }
2641     \int_set:Nn \l_@@_last_col_int \c_zero_dim
2642     \VNiceArray
2643 }
2644 { \endVNiceArray }

2645 \NewDocumentEnvironment { pNiceArrayRC } { }
2646 {
2647     \@@_error:nn { Obsolete-environment }
2648     { the~options~'last-col'~and~'first-row' }
2649     \int_set:Nn \l_@@_last_col_int \c_zero_dim
2650     \int_set:Nn \l_@@_first_row_int \c_zero_int
2651     \pNiceArray
2652 }
2653 { \endpNiceArray }

2654 \NewDocumentEnvironment { bNiceArrayRC } { }
2655 {
2656     \@@_error:nn { Obsolete-environment }
2657     { the~options~'last-col'~and~'first-row' }
2658     \int_set:Nn \l_@@_last_col_int \c_zero_dim
2659     \int_set:Nn \l_@@_first_row_int \c_zero_int
2660     \bNiceArray
2661 }
2662 { \endbNiceArray }

2663 \NewDocumentEnvironment { BNiceArrayRC } { }
2664 {
2665     \@@_error:nn { Obsolete-environment }
2666     { the~options~'last-col'~and~'first-row' }
2667     \int_set:Nn \l_@@_last_col_int \c_zero_dim
2668     \int_set:Nn \l_@@_first_row_int \c_zero_int
2669     \BNiceArray
2670 }
2671 { \endBNiceArray }

2672 \NewDocumentEnvironment { vNiceArrayRC } { }
2673 {
2674     \@@_error:nn { Obsolete-environment }
2675     { the~options~'last-col'~and~'first-row' }
2676     \bool_set_true:N \l_@@_last_col_bool
2677     \int_set:Nn \l_@@_first_row_int \c_zero_int
2678     \vNiceArray

```

```

2679 }
2680 { \endvNiceArray }

2681 \NewDocumentEnvironment { VNiceArrayRC } { }
2682 {
2683   \@@_error:nn { Obsolete-environment }
2684   { the~options~'last-col'~and~'first-row' }
2685   \int_set:Nn \l_@@_last_col_int \c_zero_dim
2686   \int_set:Nn \l_@@_first_row_int \c_zero_int
2687   \VNiceArray
2688 }
2689 { \endVNiceArray }

2690 \NewDocumentEnvironment { NiceArrayCwithDelims } { }
2691 {
2692   \@@_error:nn { Obsolete-environment }
2693   { the~option~'last-col' }
2694   \int_set:Nn \l_@@_last_col_int \c_zero_dim
2695   \NiceArrayWithDelims
2696 }
2697 { \endNiceArrayWithDelims }

2698 \NewDocumentEnvironment { NiceArrayRCwithDelims } { }
2699 {
2700   \@@_error:nn { Obsolete-environment }
2701   { the~options~'last-col'~and~'first-row' }
2702   \int_set:Nn \l_@@_last_col_int \c_zero_dim
2703   \int_set:Nn \l_@@_first_row_int \c_zero_int
2704   \NiceArrayWithDelims
2705 }
2706 { \endNiceArrayWithDelims }

```

18 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 StackExchange³⁴, Tikz externalization is now deactivated in the environments of the extension `nicematrix`.³⁵

Changes between version 2.1 and 2.1.2

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

Changes between version 2.1.2 and 2.1.3

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

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

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`.

Changes between version 2.1.5 and 2.2

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

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

³⁴cf. tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package

³⁵Before this version, there was an error when using `nicematrix` with Tikz externalization. In any case, it’s not possible to externalize the Tikz elements constructed by `nicematrix` because they use the options `overlay` and `remember picture`.

Changes between version 2.2 and 2.2.1

Improvement of the vertical dotted lines drawn by the specifier “:” in the preamble.
Modification of the position of the dotted lines drawn by `\hdottedline`.

Changes between version 2.2.1 and 2.3

Compatibility with the column type `S` of `siunitx`.
Option `hlines`.
A warning is issued when the `draft` mode is used. In this case, the dotted lines are not drawn.

Changes between version 2.3 and 3.0

Modification of `\Hdotsfor`. Now `\Hdotsfor` erases the `\vlines` (of “|”) as `\hdotsfor` does.
Composition of exterior rows and columns on the four sides of the matrix (and not only on two sides) with the options `first-row`, `last-row`, `first-col` and `last-col`.

Changes between version 3.0 and 3.1

Command `\Block` to draw block matrices.
Error message when the user gives an incorrect value for `last-row`.
A dotted line can no longer cross another dotted line (except the dotted lines drawn by `\cdottedline`, the symbol `:` (in the preamble of the array) and `\line` in `code-after`.
The starred versions of `\Cdots`, `\Ldots`, etc. are now deprecated because, with the new implementation, they become pointless. These starred versions are no longer documented.
The vertical rules in the matrices (drawn by `|`) are now compatible with the color fixed by `colortbl`.
Correction of a bug: it was not possible to use the colon `:` in the preamble of an array when `pdflatex` was used with `french-babel` (because `french-babel` activates the colon in the beginning of the document).

Changes between version 3.1 and 3.2 (and 3.2a)

Option `small`.

Changes between version 3.2 and 3.3

The options `first-row`, `last-row`, `first-col` and `last-col` are now available in the environments `{NiceMatrix}`, `{pNiceMatrix}`, `{bNiceMatrix}`, etc.
The option `columns-width=auto` doesn't need any more a second compilation.
The options `renew-dots`, `renew-matrix` and `transparent` are now available as package options (as said in the documentation).
The previous version of `nicematrix` was incompatible with a recent version of `expl3` (released 2019/09/30). This version is compatible.

Changes between version 3.3 and 3.4

Following a discussion on TeX StackExchange³⁶, optimization of Tikz externalization is disabled in the environments of `nicematrix` when the class `standalone` or the package `standalone` is used.

Changes between version 3.4 and 3.5

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

³⁶cf. tex.stackexchange.com/questions/510841/nicematrix-and-tikz-external-optimize

Changes between version 3.5 and 3.6

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

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

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

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

Changes between version 3.6 and 3.7

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

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

Changes between version 3.7 and 3.8

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

Changes between version 3.8 and 3.9

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

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

Index

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

Symbols	
@@ commands:	
<code>\@@_Block:</code>	526, 2262
<code>\@@_Block_i</code>	2263, 2264
<code>\@@_Block_ii:nnnn</code>	2264, 2265
<code>\@@_Block_iii:nnnnn</code>	2280, 2288
<code>\@@_Cdots</code>	518, 531, 1700
<code>\g_@@_Cdots_lines_tl</code>	573, 1156
<code>\@@_Cell:</code>	122, 288, 373, 514, 515, 516
<code>\@@_Ddots</code>	520, 533, 1712
<code>\g_@@_Ddots_lines_tl</code>	576, 1154
<code>\@@_Hdotsfor:</code>	524, 536, 1741
<code>\@@_Hdotsfor_i</code>	1744, 1748, 1752
<code>\g_@@_Hdotsfor_lines_tl</code> . . .	578, 1152, 1754
<code>\@@_Hspace:</code>	523, 1724
<code>\@@_Iddots</code>	521, 534, 1718
<code>\g_@@_Iddots_lines_tl</code>	577, 1155
<code>\@@_Ldots</code>	517, 530, 535, 1694
<code>\g_@@_Ldots_lines_tl</code>	574, 1157
<code>\l_@@_NiceArray_bool</code>	54, 601, 627, 658, 667, 771, 967, 1880
<code>\g_@@_NiceMatrixBlock_int</code>	50, 2027, 2032, 2035, 2046
<code>\@@_OnlyMainNiceMatrix:n</code>	527, 2010
<code>\@@_Vdots</code>	519, 532, 1706
<code>\g_@@_Vdots_lines_tl</code>	575, 1153
<code>\@@_actualization_for_first_and_last_row:</code>	315, 340, 851, 922
<code>\@@_actually_draw_Ldots:</code> . .	1368, 1371, 1800
<code>\@@_adapt_S_column:</code>	97, 112, 587
<code>\@@_add_to_empty_cells:</code>	1688, 1698, 1704, 1710, 1716, 1722, 1726
<code>\@@_adjust_with_col_nodes:</code>	1332, 1404, 1452
<code>\@@_after_array:</code>	824, 1088
<code>\@@_after_array_i:</code>	1091, 1097
<code>\@@_array:</code>	436, 677
<code>\l_@@_auto_columns_width_bool</code>	136, 185, 557, 682, 695, 2022
<code>\@@_begin_of_row:</code>	294, 308, 830
<code>\l_@@_block_auto_columns_width_bool</code>	598, 696, 2015, 2020, 2030, 2040
<code>\@@_cdots</code>	1684, 1703
<code>\g_@@_code_after_tl</code>	77, 195, 564, 1168, 1169, 2278
<code>\l_@@_code_for_first_col_tl</code>	150, 844
<code>\l_@@_code_for_first_row_tl</code>	152, 302
<code>\l_@@_code_for_last_col_tl</code>	151, 915
<code>\l_@@_code_for_last_row_tl</code>	153, 305
<code>\g_@@_col_total_int</code>	296, 297, 542, 711, 712, 900, 901, 1102, 2063, 2073, 2187

\c_@@_colortbl_loaded_bool ...	62, 67, 484	\l_@@_extra_right_margin_dim	147, 170, 741, 936
\l_@@_columns_width_dim	51, 186, 243, 558, 683, 704, 2028, 2034	@@_extract_coords:	2232, 2239
@@_computations_for_large_nodes: ...	2115, 2134, 2144	@@_fatal:n	20, 58, 589
@@_computations_for_medium_nodes: ..	2054, 2103, 2114, 2127	@@_fatal:nn	21
@@_compute_width_of_array:	2119, 2137, 2139, 2213	\l_@@_final_i_int	1143, 1184, 1189, 1192, 1213, 1218, 1224, 1235, 1242, 1325, 1396, 1444, 1769, 1791
@@_create_large_nodes:	1140, 2109, 2111, 2120, 2142	\l_@@_final_j_int	1144, 1185, 1190, 1198, 1204, 1214, 1218, 1225, 1236, 1326, 1397, 1445, 1786, 1792, 1794
@@_create_medium_and_large_nodes: ..	1137, 2108, 2121, 2124, 2140	\l_@@_final_open_bool	1146, 1191, 1195, 1201, 1207, 1211, 1239, 1308, 1346, 1390, 1407, 1438, 1455, 1476, 1489, 1523, 1565, 1617, 1628, 1651, 1652, 1767, 1787, 1795, 1798, 1824, 1908
@@_create_medium_nodes:	1138, 1307, 1308, 1799, 2100, 2107, 2122, 2141	@@_find_extremities_of_line:nnnn ...	1179, 1367, 1418, 1466, 1513, 1555
@@_create_nodes:	2105, 2117, 2128, 2133, 2136, 2183	\l_@@_first_col_int	82, 83, 196, 279, 293, 655, 687, 766, 1884, 1962, 2063, 2073, 2147, 2187, 2359, 2365, 2371
@@_ddots	1686, 1715	\l_@@_first_row_int	80, 81, 197, 283, 540, 754, 773, 1966, 2056, 2070, 2146, 2185, 2357, 2650, 2659, 2668, 2677, 2686, 2703
\l_@@_ddots_int	1127, 1529, 1530	@@_hdottedline:	522, 1868
@@_define_com:nnn	2377, 2386, 2387, 2388, 2389, 2390	\l_@@_hlines_bool	134, 155, 454
@@_define_env:n	1013, 1032, 1033, 1034, 1035, 1036, 1037	\g_@@_ht_last_row_dim	313, 329, 330, 502, 503, 763
\l_@@_delta_x_one_dim	1129, 1532, 1542	\g_@@_ht_row_one_dim .	326, 327, 498, 499, 778
\l_@@_delta_x_two_dim	1131, 1574, 1584	\g_@@_ht_row_zero_dim	321, 322, 496, 497, 757, 778
\l_@@_delta_y_one_dim	1130, 1534, 1542	@@_i:	2056, 2058, 2059, 2060, 2061, 2070, 2075, 2079, 2080, 2081, 2082, 2088, 2089, 2090, 2091, 2148, 2150, 2153, 2154, 2158, 2159, 2185, 2191, 2194, 2200, 2203, 2234, 2243, 2248, 2255, 2258
\l_@@_delta_y_two_dim	1132, 1576, 1584	@@_iddots	1687, 1721
@@_dotfill:	1850, 1852, 1892	\l_@@_iddots_int	1128, 1571, 1572
@@_double_int_eval:n	1804, 1811, 1812	@@_if_not_empty_cell:nn	1038
\g_@@_dp_ante_last_row_dim	311, 500, 501, 789	@@_if_not_empty_cell:nnTF	1218, 1278, 1777, 1791
\g_@@_dp_last_row_dim	311, 312, 331, 332, 504, 505, 763, 789	\l_@@_impossible_line_bool	61, 1243, 1304, 1366, 1368, 1417, 1419, 1465, 1467, 1512, 1514, 1554, 1556
\g_@@_dp_row_zero_dim	319, 320, 494, 495, 757	\l_@@_in_env_bool	53, 589, 590
\c_@@_draft_bool	10, 11, 35, 422, 1746, 1817, 1849, 1901	\l_@@_initial_i_int	1141, 1182, 1249, 1252, 1273, 1279, 1285, 1296, 1303, 1319, 1381, 1429, 1768, 1777
@@_draw_Cdots:nn	1413	\l_@@_initial_j_int	1142, 1183, 1250, 1258, 1264, 1274, 1279, 1286, 1297, 1320, 1382, 1430, 1772, 1778, 1780
@@_draw_Ddots:nn	1508	\l_@@_initial_open_bool	1145, 1251, 1255, 1261, 1267, 1271, 1300, 1307, 1334, 1375, 1405, 1423, 1453, 1471, 1484, 1518, 1560, 1615, 1650, 1766, 1773, 1781, 1798, 1823, 1907
@@_draw_Hdotsfor:nnn	1756, 1764	@@_instruction_of_type:n	423, 425, 1696, 1702, 1708, 1714, 1720
@@_draw_Iddots:nn	1550	\l_@@_inter_dots_dim	72, 73, 1150, 1620, 1624, 1631, 1635, 1641, 1646, 1658, 1666, 1855, 1858
@@_draw_Ldots:nn	1362	@@_j:	2063, 2065, 2066, 2067, 2068, 2073, 2075, 2079, 2082, 2084, 2085, 2088, 2091, 2093, 2094, 2161,
@@_draw_Vdots:nn	1461		
@@_draw_tikz_line:	1411, 1457, 1504, 1546, 1588, 1594, 1841, 1956		
@@_end_Cell: ...	124, 334, 377, 514, 515, 516		
\g_@@_env_int	49, 351, 405, 597, 610, 613, 691, 708, 721, 733, 872, 950, 1048, 1061, 1071, 1112, 1164, 1234, 1295, 1318, 1324, 1337, 1341, 1350, 1356, 1380, 1395, 1428, 1443, 1593, 1692, 1827, 1829, 1873, 1874, 1910, 1942, 2075, 2078, 2088, 2200, 2217, 2220, 2227, 2255, 2271		
@@_error:n ...	17, 232, 236, 242, 251, 254, 263, 265, 273, 275, 281, 286, 749, 1093, 1899		
@@_error:nn	18, 191, 2607, 2615, 2623, 2631, 2639, 2647, 2656, 2665, 2674, 2683, 2692, 2700		
@@_error:nnn	19, 1844		
@@_everycr:	449, 489, 492		
@@_everycr_i:	450, 451		
\l_@@_exterior_arraycolsep_bool	131, 239, 660, 669, 1882		
\l_@@_extra_left_margin_dim	146, 169, 675, 888		

2163, 2167, 2169, 2173, 2174, 2187, 2190, 2193, 2200, 2203, 2235, 2242, 2247, 2255, 2258	\@@_renew_matrix: . 230, 233, 237, 2327, 2395
\l_@@_l_dim . 1596, 1597, 1613, 1620, 1624, 1631, 1635, 1641, 1646, 1658, 1659, 1666, 1667	\@@_renewcolumnntype:nn 367, 546, 547
\g_@@_large_nodes_bool 141, 481, 560, 1136, 1140, 1872	\@@_restore_iRow_jCol: . . . 1094, 1172, 1174
\l_@@_large_nodes_bool 140, 160, 481	\@@_retrieve_coords:nn 1310, 1331, 1373, 1421, 1469, 1482, 1516, 1558
\l_@@_last_col_bool 2676	\c_@@_revtex_bool 26, 28, 31, 438
\g_@@_last_col_found_bool 89, 572, 710, 819, 898, 1103	\g_@@_right_delim_dim . . . 626, 630, 650, 934
\l_@@_last_col_int 87, 88, 264, 274, 282, 664, 1023, 1025, 2361, 2367, 2373, 2609, 2617, 2625, 2633, 2641, 2649, 2658, 2667, 2685, 2694, 2702	\l_@@_right_margin_dim 143, 165, 740, 935, 1890, 2181
\l_@@_last_row_int 84, 85, 198, 284, 304, 458, 605, 612, 619, 743, 747, 750, 760, 782, 839, 841, 910, 912, 1105, 1970, 1977, 1985, 1995, 2002, 2013, 2369, 2436	\g_@@_row_total_int 541, 1104, 1113, 1120, 2056, 2070, 2185
\l_@@_last_row_without_value_bool 86, 607, 745, 1107	\l_@@_save_iRow_int 78, 469, 1176
\g_@@_last_vdotted_col_int 561, 563, 569, 571	\l_@@_save_jCol_int 79, 472, 1177
\@@_ldots 1683, 1697	\@@_set_size:n 2345, 2354
\g_@@_left_delim_dim 625, 629, 642, 886	\c_@@_siunitx_loaded_bool . . 90, 94, 99, 570
\l_@@_left_margin_dim 142, 163, 674, 887, 1890, 2178	\l_@@_small_bool 154, 300, 475, 833, 904, 1147, 1854, 1861
\l_@@_letter_for_dotted_lines_str 250, 256, 257, 549, 550, 2594	\l_@@_stop_loop_bool 1186, 1187, 1215, 1219, 1246, 1247, 1275, 1280
\@@_line:nn 1167, 1806	\c_@@_table_collect_begin_tl . 107, 109, 122
\@@_line_i:nn 1810, 1815	\c_@@_table_print_tl 110, 111, 124
\g_@@_max_cell_width_dim 338, 339, 599, 700, 2021, 2047	\@@_test_if_math_mode: 55, 588, 977, 985, 993, 1001, 1009
\g_@@_medium_nodes_bool 139, 480, 1134, 2277	\l_@@_the_array_box 653, 673, 794, 807
\l_@@_medium_nodes_bool 138, 159, 480	\g_@@_type_env_str 76, 582, 584, 968, 969, 975, 976, 983, 984, 991, 992, 999, 1000, 1007, 1008, 1017, 1171, 2381, 2411, 2418, 2437, 2449, 2536, 2587, 2595
\@@_msg_new:nn 22, 33, 2403, 2409, 2416, 2422, 2428, 2434, 2442, 2447, 2453, 2468, 2463, 2580, 2585, 2592, 2598	\@@_vdots 1685, 1709
\@@_msg_new:nnn . . . 23, 2468, 2497, 2533, 2566	\@@_vdottedline:n 565, 1896
\@@_msg_redirect_name:nn 24, 245, 2400	\@@_vline: 600, 1960
\@@_multicolumn:nnn 525, 1730	\@@_vline_i: 68, 1971, 1978, 1986, 1996, 2003, 2008
\g_@@_multicolumn_cells_seq 538, 1735, 2082, 2091, 2209	\g_@@_width_first_col_dim 145, 769, 852, 855
\g_@@_multicolumn_sizes_seq 539, 1737, 2210	\g_@@_width_last_col_dim 144, 821, 923, 926
\l_@@_name_str 137, 193, 355, 357, 409, 411, 608, 617, 620, 876, 878, 954, 956, 1115, 1119, 2202, 2203, 2257, 2258	\g_@@_x_final_dim 1314, 1327, 1357, 1481, 1494, 1500, 1502, 1533, 1541, 1575, 1583, 1603, 1640, 1656, 1821, 1838, 1905, 1916, 1938, 1950, 1953
\g_@@_names_seq 52, 190, 192, 2578	\g_@@_x_initial_dim . . . 1312, 1321, 1342, 1481, 1494, 1497, 1500, 1502, 1533, 1541, 1575, 1583, 1604, 1640, 1654, 1656, 1674, 1677, 1819, 1835, 1903, 1915, 1934, 1947, 1952
\l_@@_nb_cols_int 2348, 2353, 2356, 2360, 2366, 2372	\g_@@_y_final_dim 1315, 1328, 1406, 1408, 1410, 1454, 1456, 1535, 1538, 1577, 1580, 1607, 1645, 1664, 1822, 1839, 1906, 1917, 1939
\l_@@_nb_rows_int 2347, 2352, 2363	\g_@@_y_initial_dim 1313, 1322, 1406, 1408, 1409, 1454, 1456, 1535, 1540, 1577, 1582, 1608, 1645, 1662, 1664, 1674, 1678, 1820, 1836, 1904, 1919, 1925, 1926, 1935
\@@_node_for_multicolumn:nn 2211, 2237	\
\l_@@_nullify_dots_bool 135, 158, 1697, 1703, 1709, 1715, 1721	\
\@@_old_multicolumn 1729, 1732	\
\l_@@_parallelize_diags_bool 132, 133, 156, 1125, 1527, 1569	\{ 994, 2390, 2405, 2444, 2500, 2600
\l_@@_pos_env_str 129, 130, 269, 270, 271, 447, 775, 784	\} 994, 2390, 2405, 2444, 2500, 2600
\@@_pre_array: 466, 624	\l 1010, 2389
\c_@@_preamble_first_col_tl 656, 826	
\c_@@_preamble_last_col_tl 665, 894	
\l_@@_radius_dim 74, 75, 1149, 1675	
\@@_renew_NC@rewrite@S: 115, 570	
\l_@@_renew_dots_bool . . 157, 238, 528, 2393	

A

\array 446

`\arraycolsep` 164, 166, 168, 478,
 629, 630, 689, 692, 700, 704, 729, 736, 768,
 806, 808, 822, 891, 929, 1886, 1889, 1952, 1953
`\arrayrulewidth` 460, 461
`\arraystretch` 477
`\AtBeginDocument` 63, 91
`\AutoNiceMatrixWithDelims` 2350, 2383

B

`\begin` 1019, 1067, 1316, 1339, 1354,
 1670, 1832, 1912, 1922, 1931, 1944, 2102,
 2113, 2126, 2131, 2215, 2269, 2299, 2314, 2355
`\bgroup` 362
`\Block` 526
`\BNiceArray` 2626, 2669
`\bNiceArray` 2618, 2660
`\BNiceMatrix` 2342
`\bNiceMatrix` 2339

bool commands:

`\bool_do_until:Nn` 1187, 1247
`\bool_gset_eq:NN` 480, 481
`\bool_gset_false:N` 572
`\bool_gset_true:N` 560, 898, 1872, 2277
`\bool_if:NTF` 35, 99, 300, 422, 438,
 454, 475, 484, 528, 570, 589, 598, 601, 627,
 658, 660, 667, 669, 745, 771, 819, 833, 904,
 1064, 1107, 1125, 1136, 1140, 1147, 1211,
 1239, 1271, 1300, 1308, 1334, 1346, 1368,
 1375, 1390, 1405, 1407, 1419, 1423, 1438,
 1453, 1455, 1467, 1471, 1476, 1484, 1489,
 1495, 1499, 1514, 1518, 1523, 1527, 1556,
 1560, 1565, 1569, 1615, 1617, 1628, 1650,
 1651, 1652, 1697, 1703, 1709, 1715, 1721,
 1746, 1817, 1849, 1854, 1861, 1901, 2030, 2040
`\bool_if:nTF` 555, 680,
 693, 710, 834, 905, 1103, 1134, 1307, 1696,
 1702, 1708, 1714, 1720, 1798, 1825, 1878, 2290
`\bool_new:N`
 .. 10, 26, 53, 54, 61, 62, 86, 89, 90, 131,
 132, 134, 135, 136, 138, 139, 140, 141, 2015
`\bool_set:Nn` 1480, 1493
`\bool_set_false:N` 1040, 1054,
 1145, 1146, 1186, 1191, 1246, 1251, 1366,
 1417, 1465, 1512, 1554, 1766, 1767, 1823, 1824
`\bool_set_true:N` 11,
 28, 31, 67, 94, 133, 185, 238, 590, 607, 967,
 1062, 1195, 1201, 1207, 1215, 1219, 1243,
 1255, 1261, 1267, 1275, 1280, 1304, 1773,
 1781, 1787, 1795, 1907, 1908, 2020, 2022, 2676
`\l_tmpa_bool` 1040, 1054, 1062, 1064, 1480, 1499
`\l_tmpb_bool` 1493, 1495

box commands:

`\box_clear_new:N` 653
`\box_dp:N` .. 320, 332, 382, 495, 501, 505, 816
`\box_ht:N` .. 322, 327, 330, 497, 499, 503, 815
`\box_move_down:nn` 383, 786
`\box_move_up:nn` 397, 777
`\box_set_dp:Nn` 816
`\box_set_ht:Nn` 815
`\box_use:N` 363, 381, 397, 883, 961
`\box_use_drop:N` 794, 807, 817
`\box_wd:N` 339, 387, 642, 650, 856, 927

`\l_tmpa_box` 298, 320, 322, 327, 330,
 332, 339, 363, 372, 381, 635, 642, 643, 650,
 797, 815, 816, 817, 831, 856, 883, 902, 927, 961
`\l_tmpb_box` 380, 382, 387, 397

C

`\Cdots` 518
`\cdots` 531, 1684
`\cleaders` 1856
`\coordinate` 391, 396, 691,
 708, 719, 731, 2189, 2192, 2240, 2245, 2270
`\crrcr` 686
cs commands:
`\cs_generate_variant:Nn` 366, 1331, 2052, 2053
`\cs_gset:Npn` 1112, 1119, 2045, 2227
`\cs_gset:Npx` 1690
`\cs_gset_eq:NN` 112, 488
`\cs_if_exist:NTF`
 468, 471, 591, 594, 610, 617, 1041,
 1055, 1100, 1176, 1177, 1336, 1348, 1377,
 1392, 1425, 1440, 1873, 1909, 1941, 2032, 2074
`\cs_if_exist_p:N` 1827, 1829
`\cs_if_free:NTF`
 1045, 1231, 1292, 1364, 1415, 1463, 1510, 1552
`\cs_new:Npn`
 449, 1730, 1741, 1804, 1868, 2010, 2264, 2265
`\cs_new_protected:Nn` 288,
 308, 334, 367, 1088, 1097, 1174, 1179,
 1310, 1332, 1362, 1371, 1413, 1461, 1508,
 1550, 1594, 1688, 1724, 1764, 1815, 1896, 2237
`\cs_new_protected:Npn` 17, 18, 19,
 20, 21, 22, 23, 24, 55, 115, 315, 425, 436,
 451, 466, 1013, 1806, 1960, 2054, 2100,
 2111, 2124, 2144, 2183, 2213, 2232, 2288, 2345
`\cs_set:Npn` 443, 477, 482,
 506, 1181, 1221, 1282, 1802, 1852, 2234, 2235
`\cs_set_eq:NN` 103, 440,
 441, 442, 517, 518, 519, 520, 521, 522, 523,
 524, 525, 526, 527, 530, 531, 532, 533, 534,
 535, 536, 543, 544, 545, 547, 1167, 1683,
 1684, 1685, 1686, 1687, 1729, 1850, 2008, 2009
`\cs_set_protected:Npn`
 68, 97, 423, 600, 2107, 2108,
 2120, 2121, 2140, 2141, 2142, 2327, 2377, 2379

D

`\Ddots` 520
`\ddots` 533, 1686
`\DeclareOption` 11, 12
dim commands:
`\dim_abs:n` 1081
`\dim_add:Nn` 2179
`\dim_compare:nNnTF` 1080, 1613
`\dim_compare_p:nNn` ... 558, 683, 1481, 1494
`\dim_eval:n` 2228
`\dim_gadd:Nn`
 1409, 1410, 1654, 1662, 1677, 1678, 1952, 1953
`\dim_gset:Nn` 319,
 321, 326, 329, 331, 338, 495, 497, 499, 501,
 503, 505, 629, 630, 642, 650, 699, 703, 852,
 923, 1076, 1078, 1321, 1322, 1327, 1328,
 1342, 1357, 1497, 1538, 1580, 1835, 1836,
 1838, 1839, 1915, 1916, 1917, 1919, 1925,
 1934, 1935, 1938, 1939, 1947, 1950, 2218, 2222

<code>\dim_gset_eq:NN</code>	311, 1406, 1408, 1454, 1456, 1502
<code>\dim_gzero:N</code>	312, 313
<code>\dim_gzero_new:N</code>	494, 496, 498, 500, 502, 504, 599, 625, 626, 1312, 1313, 1314, 1315, 2021
<code>\dim_max:nn</code>	320, 322, 327, 330, 332, 339, 854, 925, 1499, 1926, 2053, 2090, 2094
<code>\dim_min:nn</code>	1499, 2052, 2081, 2085
<code>\dim_new:N</code>	51, 72, 74, 142, 143, 144, 145, 146, 147
<code>\dim_ratio:nn</code>	1542, 1584, 1620, 1624, 1631, 1635, 1641, 1646, 1657, 1665
<code>\dim_set:Nn</code>	73, 75, 186, 243, 366, 382, 478, 756, 762, 1149, 1150, 1532, 1534, 1574, 1576, 1597, 1638, 1643, 1855, 1875, 2034, 2061, 2068, 2080, 2084, 2089, 2093, 2150, 2163, 2301
<code>\dim_set_eq:NN</code>	634, 645, 1874, 2059, 2066, 2158, 2173
<code>\dim_sub:Nn</code>	2176
<code>\dim_use:N</code>	1603, 1604, 1607, 1608, 2047, 2153, 2154, 2166, 2168, 2190, 2191, 2193, 2194, 2242, 2243, 2247, 2248
<code>\dim_zero:N</code>	759, 765, 2028
<code>\dim_zero_new:N</code>	1129, 1130, 1131, 1132, 1596, 1819, 1820, 1821, 1822, 1903, 1904, 1905, 1906, 2058, 2060, 2065, 2067
<code>\c_max_dim</code>	1919, 2059, 2061, 2066, 2068
<code>\g_tmpa_dim</code>	699, 703, 706, 717, 728, 1076, 1081, 2218, 2228
<code>\l_tmpa_dim</code>	382, 383, 397, 756, 759, 778, 803, 815, 1638, 1677, 1874, 1875, 1889, 2301, 2310
<code>\g_tmpb_dim</code>	1078, 1081, 2222, 2228
<code>\l_tmpb_dim</code>	762, 765, 788, 810, 816, 1643, 1678
<code>\c_zero_dim</code>	344, 345, 415, 558, 634, 645, 683, 863, 864, 941, 942, 1613, 2199, 2254, 2609, 2617, 2625, 2633, 2641, 2649, 2658, 2667, 2685, 2694, 2702
<code>\dots</code>	535
E	
<code>\egroup</code>	364
else commands:	
<code>\else:</code>	57
<code>\end</code>	1030, 1079, 1329, 1343, 1358, 1680, 1840, 1918, 1927, 1940, 1951, 2106, 2118, 2138, 2223, 2275, 2311, 2323, 2375
<code>\endarray</code>	738
<code>\endBNiceArray</code>	2628, 2671
<code>\endbNiceArray</code>	2620, 2662
<code>\endBNiceMatrix</code>	2343
<code>\endbNiceMatrix</code>	2340
<code>\endNiceArrayWithDelims</code>	972, 980, 988, 996, 1004, 1012, 2697, 2706
<code>\endpNiceArray</code>	2612, 2653
<code>\endpNiceMatrix</code>	2331
<code>\endVNiceArray</code>	2644, 2689
<code>\endvNiceArray</code>	2636, 2680
<code>\endVNiceMatrix</code>	2337
<code>\endvNiceMatrix</code>	2334
<code>\everycr</code>	492, 508
exp commands:	
<code>\exp_after:wN</code>	119
<code>\exp_args:NV</code>	550, 677
<code>\exp_not:n</code>	2285
<code>\ExplSyntaxOff</code>	1123, 2049, 2230
<code>\ExplSyntaxOn</code>	1109, 2042, 2224
F	
<code>\fi</code>	194
fi commands:	
<code>\fi:</code>	59
fp commands:	
<code>\fp_to_dim:n</code>	1599
G	
group commands:	
<code>\group_begin:</code>	101, 633, 1099
<code>\group_end:</code>	106, 651, 1170
H	
<code>\halign</code>	510, 512
hbox commands:	
<code>\hbox:n</code>	42, 44, 46, 393, 804
<code>\hbox_overlap_left:n</code>	858
<code>\hbox_overlap_right:n</code>	930, 1876
<code>\hbox_set:Nn</code>	380, 635, 643, 797
<code>\hbox_set:Nw</code>	298, 372, 673, 831, 902
<code>\hbox_set_end:</code>	337, 378, 742, 850, 921
<code>\hbox_to_wd:nn</code>	387, 1857, 1887
<code>\Hdotsfor</code>	524
<code>\hdotsfor</code>	536
<code>\hdottedline</code>	522
<code>\hfil</code>	389, 547
<code>\hfill</code>	1865
<code>\hrule</code>	460
<code>\Hspace</code>	523
<code>\hspace</code>	1727
<code>\hss</code>	547, 1862
I	
<code>\ialign</code>	482, 506
<code>\iddots</code>	521
<code>\iddots</code>	37, 534, 1687
if commands:	
<code>\if_mode_math:</code>	57
<code>\ifstandalone</code>	594
int commands:	
<code>\int_add:Nn</code>	1189, 1190, 1273, 1274
<code>\int_compare:nNnTF</code>	291, 293, 301, 302, 304, 317, 324, 456, 458, 561, 605, 655, 664, 687, 743, 747, 754, 760, 766, 773, 782, 1023, 1058, 1090, 1105, 1192, 1194, 1198, 1200, 1204, 1206, 1252, 1254, 1258, 1260, 1264, 1266, 1530, 1572, 1733, 1770, 1784, 1898, 1962, 1964, 1966, 1968, 1970, 1975, 1977, 1983, 1985, 1991, 1993, 1995, 2000, 2002, 2012, 2013, 2267, 2297, 2357, 2359, 2361, 2365, 2367, 2369, 2371, 2373
<code>\int_compare:nTF</code>	249
<code>\int_compare_p:nNn</code>	836, 839, 841, 907, 910, 912, 1884, 2292, 2293
<code>\int_eval:n</code>	1025, 1736, 1805, 2154, 2158, 2169, 2173, 2247, 2283, 2284
<code>\int_gadd:Nn</code>	1739

<code>\int_gdecr:N</code>	1103	<code>\l_keys_key_tl</code>	2465, 2470, 2499, 2535
<code>\int_gincr:N</code>	290, 310, 597, 716, 727, 899, 2027	<code>\keys_set:nn</code>	259, 602, 603, 1018, 2029
<code>\int_gset:Nn</code>	296, 540, 571, 709, 900	<code>\l_keys_value_tl</code>	2414, 2568
<code>\int_gset_eq:NN</code>	563, 750, 1102, 1104, 1176, 1177	L	
<code>\int_gsub:Nn</code>	1106	<code>\ldots</code>	517
<code>\int_gzero:N</code>	453	<code>\ldots</code>	530, 1683
<code>\int_gzero_new:N</code>	470, 473, 541, 542, 569	<code>\left</code>	638, 647, 800
<code>\int_incr:N</code>	1529, 1571	<code>\line</code>	1167, 2405
<code>\int_max:nn</code>	297, 901	<code>\lineskip</code>	757, 763
<code>\int_new:N</code>	49, 50, 78, 79, 80, 82, 84, 87	M	
<code>\int_set:Nn</code>	81, 83, 85, 88, 264, 612, 619, 1182, 1183, 1184, 1185, 1619, 1623, 1630, 1634, 1648, 1768, 1769, 1772, 1776, 1778, 1780, 1786, 1790, 1792, 1794, 2146, 2147, 2347, 2348, 2609, 2617, 2625, 2633, 2641, 2649, 2650, 2658, 2659, 2667, 2668, 2677, 2685, 2686, 2694, 2702, 2703	<code>\makebox</code>	381
<code>\int_set_eq:NN</code>	469, 472	math commands:	
<code>\int_step_inline:nn</code>	1920	<code>\c_math_toggle_token</code>	299, 336, 637, 639, 646, 648, 676, 739, 799, 813, 832, 849, 903, 920, 1860, 1863
<code>\int_step_variable:nnn</code>	1671, 1801	<code>\mathinner</code>	39
<code>\int_step_variable:nNn</code>	2148, 2161	<code>\mkern</code>	41, 43, 45, 46
<code>\int_step_variable:nnNn</code>	2056, 2063, 2070, 2072, 2185, 2187	msg commands:	
<code>\int_sub:Nn</code>	1213, 1214, 1249, 1250	<code>\msg_error:nn</code>	17
<code>\int_use:N</code>	351, 352, 353, 358, 359, 405, 406, 407, 412, 413, 431, 432, 565, 610, 613, 691, 708, 721, 722, 733, 734, 872, 873, 879, 950, 951, 952, 957, 958, 1042, 1048, 1049, 1050, 1056, 1059, 1071, 1072, 1073, 1112, 1113, 1120, 1164, 1224, 1225, 1234, 1235, 1236, 1242, 1285, 1286, 1295, 1296, 1297, 1303, 1318, 1319, 1320, 1324, 1325, 1326, 1337, 1341, 1350, 1351, 1356, 1380, 1381, 1382, 1395, 1396, 1397, 1428, 1429, 1430, 1443, 1444, 1445, 1593, 1691, 1692, 1736, 1757, 1758, 1827, 1829, 1873, 1874, 1910, 1937, 1942, 1949, 2032, 2035, 2046, 2075, 2078, 2088, 2180, 2200, 2217, 2220, 2221, 2227, 2255, 2271, 2273, 2274, 2281, 2282, 2436, 2437, 2438	<code>\msg_error:nnn</code>	18, 1241, 1302
<code>\int_zero:N</code>	196, 197, 274, 279, 282, 283	<code>\msg_error:nnnn</code>	19, 2295
<code>\int_zero_new:N</code>	1127, 1128, 1141, 1142, 1143, 1144, 2352, 2353	<code>\msg_fatal:nn</code>	20, 21
<code>\c_one_int</code>	249, 291, 293, 324, 1106, 1367, 1418, 1466, 1513, 1530, 1572	<code>\msg_new:nnn</code>	22
<code>\g_tmpa_int</code>	709, 716, 722, 727, 734	<code>\msg_new:nnnn</code>	23
<code>\l_tmpa_int</code>	1619, 1623, 1630, 1634, 1658, 1666, 1671, 1776, 1777, 1790, 1791	<code>\msg_redirect_name:nnn</code>	25
<code>\l_tmpb_int</code>	1648, 1660, 1668	<code>\msg_warning:nn</code>	36
<code>\c_zero_int</code>	301, 302, 317, 655, 754, 766, 773, 836, 907, 1090, 1367, 1418, 1466, 1884, 1898, 1962, 1964, 1966, 1968, 1975, 1983, 1991, 2000, 2012, 2357, 2359, 2365, 2371, 2650, 2659, 2668, 2677, 2686, 2703	<code>\multicolumn</code>	525, 1729, 1743, 1749, 1761
iow commands:		<code>\myfiledate</code>	7
<code>\iow_now:Nn</code>	1109, 1110, 1117, 1123, 2042, 2043, 2049, 2224, 2225, 2230	<code>\myfileversion</code>	8
K		N	
<code>\kern</code>	46	<code>\newcolumntype</code>	369, 514, 515, 516, 550
keys commands:		<code>\NewDocumentCommand</code>	258, 1694, 1700, 1706, 1712, 1718, 1748, 1752, 2262, 2350
<code>\keys_define:nn</code>	148, 181, 206, 228, 260, 267, 277, 2016, 2391	<code>\NewDocumentEnvironment</code>	580, 965, 973, 981, 989, 997, 1005, 1015, 2025, 2605, 2613, 2621, 2629, 2637, 2645, 2654, 2663, 2672, 2681, 2690, 2698
		<code>\NewExpandableDocumentCommand</code>	1592
		<code>\NiceArrayWithDelims</code>	970, 978, 986, 994, 1002, 1010, 2695, 2704
		<code>\NiceMatrixLastEnv</code>	1592
		<code>\NiceMatrixOptions</code>	258, 2471
		<code>\noalign</code>	450, 488, 1870
		<code>\node</code>	348, 402, 867, 945, 2195, 2250, 2302, 2315, 2322
		<code>\normalbaselines</code>	474
		<code>\nulldelimiterspace</code>	634, 645
		O	
		<code>\omit</code>	687, 688, 715, 726
		<code>\OnlyMainNiceMatrix</code>	527, 2009
		P	
		<code>\path</code>	1833, 2310
		<code>\pgfpathcircle</code>	1673
		<code>\pgfpoint</code>	1674
		<code>\pgfpointanchor</code>	1075, 1077
		<code>\pgfusepath</code>	1676
		<code>\phantom</code>	1697, 1703, 1709, 1715, 1721
		<code>\pNiceArray</code>	2610, 2651
		<code>\pNiceMatrix</code>	2330

15	Technical remarks	13
15.1	Definition of new column types	13
15.2	Intersections of dotted lines	14
15.3	The names of the Tikz nodes created by <code>nicematrix</code>	14
15.4	Diagonal lines	14
15.5	The “empty” cells	15
15.6	The option <code>exterior-arraycolsep</code>	15
15.7	The class option <code>draft</code>	16
15.8	A technical problem with the argument of <code>\</code>	16
15.9	Obsolete environments	16
16	Examples	17
16.1	Dotted lines	17
16.2	Width of the columns	19
16.3	How to highlight cells of the matrix	19
16.4	Direct utilisation of the Tikz nodes	22
17	Implementation	23
17.1	Declaration of the package and extensions loaded	23
17.2	Technical definitions	24
17.2.1	Variables for the exterior rows and columns	26
17.2.2	The column <code>S</code> of <code>siunitx</code>	27
17.3	The options	29
17.4	Important code used by <code>{NiceArrayWithDelims}</code>	33
17.5	The environment <code>{NiceArrayWithDelims}</code>	41
17.6	The environment <code>{NiceMatrix}</code> and its variants	49
17.7	How to know whether a cell is “empty”	50
17.8	After the construction of the array	51
17.9	The actual instructions for drawing the dotted line with Tikz	62
17.10	User commands available in the new environments	64
17.11	The command <code>\line</code> accessible in code-after	67
17.12	The commands to draw dotted lines to separate columns and rows	68
17.13	The vertical rules	70
17.14	The environment <code>{NiceMatrixBlock}</code>	71
17.15	The extra nodes	72
17.16	Block matrices	77
17.17	How to draw the dotted lines transparently	78
17.18	Automatic arrays	79
17.19	We process the options	80
17.20	Error messages of the package	80
17.21	Obsolete environments	83
18	History	85
	Index	88