

The package `nicematrix`*

F. Pantigny
fpantigny@wanadoo.fr

January 2, 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 3.8 of `nicematrix`, at the date of 2020/01/02.

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>	$\left[\begin{array}{cccc} a_1 & \cdots & \cdots & a_1 \\ \vdots & & & \\ \vdots & a_2 & \cdots & a_2 \\ \vdots & \vdots & \ddots & \\ a_1 & a_2 & \cdots & a_n \end{array} \right]$
<code>\Vdots & a_2 & \Cdots & a_2 \\</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>	$\left[\begin{array}{ccc} 0 & \cdots & 0 \\ \vdots & & \vdots \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{array} \right]$
<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. 14).

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

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

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

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

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

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

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

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

3.1 The option `nullify-dots`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.2 The command `\Hdotsfor`

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

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

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

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

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

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

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

3.3 How to generate the continuous dotted lines transparently

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

- The option `renew-dots`

With this option, the commands `\ldots`, `\cdots`, `\vdots`, `\ddots`, `\iddots`³ and `\hdotsfor` are redefined within the environments provided by `nicematrix` and behave like `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots` and `\Hdotsfor`; the command `\dots` (“automatic dots” of `amsmath`) is also redefined to behave like `\Ldots`.

- The option `renew-matrix`

With this option, the environment `{matrix}` is redefined and behave like `{NiceMatrix}`, and so on for the five variants.

Therefore, with the option `transparent`, a classical code gives directly the output of `nicematrix`.

```
\NiceMatrixOptions{transparent}
\begin{pmatrix}
1 & \cdots & \cdots & 1 \\
0 & \ddots & & \vdots \\
\vdots & \ddots & \ddots & \vdots \\
0 & \cdots & 0 & 1
\end{pmatrix}
```

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

4 The Tikz nodes created by `nicematrix`

The package `nicematrix` creates a Tikz node for each cell of the considered array. These nodes are used to draw the dotted lines between the cells of the matrix. However, the user may wish to use directly these nodes. It's possible. First, the user have to give a name to the array (with the key called `name`). Then, the nodes are accessible through the names “`name-i-j`” where `name` is the name given to the array and `i` and `j` the numbers of the row and the column of the considered cell.

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

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

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

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

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

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

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

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

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

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

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

The “large nodes” of the first column and last column may appear too small for some usage. That’s why it’s possible to use the options `left-margin` and `right-margin` to add space on both sides of the array and also space in the “large nodes” of the first column and last column. In the following example, we have used the options `left-margin` and `right-margin`.⁹

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

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

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

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

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

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

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

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

⁹The options `left-margin` and `right-margin` take dimensions as values but, if no value is given, the default value is used, which is `\arraycolsep` (by default: 5 pt). There is also an option `margin` to fix both `left-margin` and `right-margin` to the same value.

6 The environment `{NiceArray}`

The environment `{NiceArray}` is similar to the environment `{array}`. As for `{array}`, the mandatory argument is the preamble of the array. However, for technical reasons, in this preamble, the user must use the letters L, C and R¹⁰ instead of l, c and r. It's possible to use the constructions `w{...}{...}`, `W{...}{...}`, `l, >{...}`, `<{...}`, `@{...}`, `!{...}` and `*{n}{...}` but the letters p, m and b should not be used.¹¹

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

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

```


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


```

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

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.

```


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


```

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`.

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

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

```

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

```

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

We have several remarks to do.

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

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

```

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

```

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

Remarks

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

8 The dotted lines to separate rows or columns

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

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

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

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

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

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

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

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

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

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

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

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

¹²The latter is not true when the extension `arydshln` is loaded besides `nicematrix`. In fact, `nicematrix` and `arydshln` are not totally compatible because `arydshln` redefines many internals of `array`. On another note, if one really wants a vertical rule running in the first and in the last row, he should use `!\vline` instead of `|` in the preamble of the array.

9 The width of the columns

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Several compilations may be necessary to achieve the job.

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

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

10 Block matrices

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}
```

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

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

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

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

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

11 The option small

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

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

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

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

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

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

12 The 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 col¹⁵. Of course, the user must not change the value of these counters which are used internally by `nicematrix`.

```

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

```

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

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).

```

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

```

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

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”).

¹⁵We 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.

```


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


```

	<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`.

```


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


```

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

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

15 Technical remarks

15.1 Intersections of dotted lines

Since the version 3.1 of `nicematrix`, the dotted lines created by `\Cdots`, `\Ldots`, `\Vdots`, etc. can't intersect.¹⁶

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

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

```


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


```

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

15.2 Diagonal lines

By default, all the diagonal lines¹⁷ 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:

¹⁶Of 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.

¹⁷We speak of the lines created by `\Ddots` and not the lines created by a command `\line` in `code-after`.

Example with parallelization (default):

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

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

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

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

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

The same example without parallelization:

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

15.3 The “empty” cells

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

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

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

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

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

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

15.4 The option `exterior-arraycolsep`

The environment `{array}` inserts an horizontal space equal to `\arraycolsep` before and after each column. In particular, there is a space equal to `\arraycolsep` before and after the array. This feature of the environment `{array}` was probably not a good idea¹⁸. 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).

15.5 The class option `draft`

The package `nicematrix` is rather slow when drawing the dotted lines (generated by `\Cdots`, `\Ldots`, `\Ddots`, etc. but also by `\hdottedline` or the specifier `:`).¹⁹ That's why, when the class option `draft` is used, the dotted lines are not drawn, for a faster compilation.

15.6 A technical problem with the argument of `\`

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

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

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

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

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

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

15.7 Obsolete environments

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

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

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

¹⁸In the documentation of `{amsmath}`, we can read: *The extra space of `\arraycolsep` that `array` adds on each side is a waste so we remove it [in `{matrix}`] (perhaps we should instead remove it from `array` in general, but that's a harder task).* It's possible to suppress these spaces for a given environment `{array}` with a construction like `\begin{array}{@{}cccc@{}}...\end{array}`.

¹⁹The main reason is that we want dotted lines with round dots (and not square dots) with the same space on both extremities of the lines. To achieve this goal, we have to construct our own system of dotted lines.

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

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 future version of `nicematrix`.

16 Examples

16.1 Dotted lines

A tridiagonal matrix:

```


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


```

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

A permutation matrix:

```


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


```

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

An example with `\Iddots`:

```


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


```

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

An example with `\multicolumn`:

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

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

An example with `\Hdotsfor`:

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

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

An example for the resultant of two polynomials:

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

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

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

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

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

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

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

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`.

```

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

```

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

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

```

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

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

```

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

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

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

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

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

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

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

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

\begin{tikzpicture}[myoptions]
\node [fit = (1-1) (1-3)] {} ;
\node [fit = (2-1) (2-3)] {} ;
\node [fit = (3-1) (3-3)] {} ;
\end{tikzpicture}
```

We obtain the following matrix.

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

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

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

We obtain the following matrix.

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

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

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

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

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}
& \\
& \end{array}
```

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}
```

$$L_i \begin{bmatrix} a_{11} & \cdots & a_{nn} \\ \vdots & & \vdots \\ a_{i1} & \cdots & a_{in} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} b_{11} & \cdots & b_{1j} & \cdots & b_{1n} \\ \vdots & & \vdots & & \vdots \\ \vdots & & b_{kj} & & \vdots \\ \vdots & & \vdots & & \vdots \\ b_{n1} & \cdots & b_{nj} & \cdots & b_{nn} \end{bmatrix}$$

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.²⁰

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

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

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

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

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

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

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

We load `array` and `amsmath`.

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

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 redefine `\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. }
```

²⁰cf. tex.stackexchange.com/questions/57424/using-of-usetikzlibrary-in-an-expl3-package-fails

```

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.²¹

```

61 \bool_new:N \l_@@_impossible_line_bool

```

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).

²¹Of course, the user should solve the problem by adding the lacking ampersands.

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”.²²

```
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:`.

²²We 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.

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:

```

\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.²³

```

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

²³Indeed, `nicematrix` may be used in the preamble of the LaTeX document. For example, in this document, we compose a matrix in the box `\ExampleOne` before loading `arydshln` (because `arydshln` is not totally compatible with `nicematrix`).

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

```
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 } }
```

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_nodes_bool` will be used to indicate whether the “medium nodes” and “large nodes” are created in the array.

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

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

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

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

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

```
142 \dim_new:N \g_@@_width_last_col_dim
```

```
143 \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`.

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

```
145 \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.

```
146 \keys_define:nn { NiceMatrix / Global }
147   {
148     code-for-first-col .tl_set:N = \l_@@_code_for_first_col_tl ,
149     code-for-last-col .tl_set:N = \l_@@_code_for_last_col_tl ,
150     code-for-first-row .tl_set:N = \l_@@_code_for_first_row_tl ,
151     code-for-last-row .tl_set:N = \l_@@_code_for_last_row_tl ,
152     small .bool_set:N = \l_@@_small_bool ,
153     hlines .bool_set:N = \l_@@_hlines_bool ,
154     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`.

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

```
156
```

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

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

```

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

```

169 {*comment}
170   code-for-first-col .value_required:n = true ,
171   code-for-last-col .value_required:n = true ,
172   code-for-first-row .value_required:n = true ,
173   code-for-last-row .value_required:n = true ,
174   renew-dots .value_forbidden:n = true ,
175 {/comment}
176 }

```

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

```

177 \keys_define:nn { NiceMatrix / Env }
178 {
179   columns-width .code:n =
180     \str_if_eq:nnTF { #1 } { auto }
181     { \bool_set_true:N \l_@@_auto_columns_width_bool }
182     { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,
183   name .code:n =
184     \unless \ifmeasuring@
185       \str_set:Nn \l_tmpa_str { #1 }
186       \seq_if_in:NVTF \g_@@_names_seq \l_tmpa_str
187       { \@@_error:nn { Duplicate-name } { #1 } }
188       { \seq_gput_left:NV \g_@@_names_seq \l_tmpa_str }
189       \str_set_eq:NN \l_@@_name_str \l_tmpa_str
190     \fi ,
191   code-after .tl_gset:N = \g_@@_code_after_tl ,
192   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
193   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
194   last-row .int_set:N = \l_@@_last_row_int ,
195   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`.

```

196 {*comment}
197   columns-width .value_required:n = true ,
198   name .value_required:n = true ,
199   code-after .value_required:n = true ,
200 {/comment}
201 }

```

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

```

202 \keys_define:nn { NiceMatrix }
203 {
204   NiceMatrixOptions .inherit:n =
205   {
206     NiceMatrix / Global ,
207   } ,
208   NiceMatrix .inherit:n =
209   {
210     NiceMatrix / Global ,
211     NiceMatrix / Env
212   } ,
213   NiceArray .inherit:n =
214   {
215     NiceMatrix / Global ,
216     NiceMatrix / Env ,
217   } ,
218   pNiceArray .inherit:n =
219   {
220     NiceMatrix / Global ,
221     NiceMatrix / Env ,
222   }
223 }

```

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

```

224 \keys_define:nn { NiceMatrix / NiceMatrixOptions }
225 {

```

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

```

226   renew-matrix .code:n = \@@_renew_matrix: ,
227   renew-matrix .value_forbidden:n = true ,
228   RenewMatrix .code:n = \@@_error:n { Option~RenewMatrix~suppressed }
229                       \@@_renew_matrix: ,
230   transparent .meta:n = { renew-dots , renew-matrix } ,
231   transparent .value_forbidden:n = true,
232   Transparent .code:n = \@@_error:n { Option~Transparent~suppressed }
233                       \@@_renew_matrix:
234                       \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}`.

```

235   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.

```

236   columns-width .code:n =
237   \str_if_eq:nnTF { #1 } { auto }
238   { \@@_error:n { Option~auto~for~columns~width } }
239   { \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.

```

240   allow-duplicate-names .code:n =
241   \@@_msg_redirect_name:nn { Duplicate~name } { none } ,
242   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`).

```

243   letter-for-dotted-lines .code:n =
244   {
245     \int_compare:nTF { \tl_count:n { #1 } = \c_one_int }
246     { \str_set:Nx \l_@@_letter_for_dotted_lines_str { #1 } }
247     { \@@_error:n { Bad~value~for~letter~for~dotted~lines } }
248   } ,
249   letter-for-dotted-lines .value_required:n = true ,

250   unknown .code:n = \@@_error:n { Unknown~key~for~NiceMatrixOptions }
251 }

252 \str_new:N \l_@@_letter_for_dotted_lines_str
253 \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.

```

254 \NewDocumentCommand \NiceMatrixOptions { m }
255 { \keys_set:nn { NiceMatrix / NiceMatrixOptions } { #1 } }

```

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

```

256 \keys_define:nn { NiceMatrix / NiceMatrix }
257 {
258   last-col .code:n = \tl_if_empty:nTF {#1}
259   { \@@_error:n { last-col~empty~for~NiceMatrix } }
260   { \int_set:Nn \l_@@_last_col_int { #1 } } ,
261   unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix }
262 }

```

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

```

263 \keys_define:nn { NiceMatrix / NiceArray }
264 {

```

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

```

265   c .code:n = \str_set:Nn \l_@@_pos_env_str c ,
266   t .code:n = \str_set:Nn \l_@@_pos_env_str t ,
267   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.

```

268   last-col .code:n = \tl_if_empty:nF {#1}
269   { \@@_error:n { last-col~non~empty~for~NiceArray } }
270   { \int_zero:N \l_@@_last_col_int ,
271   unknown .code:n = \@@_error:n { Unknown~option~for~NiceArray }
272 }

273 \keys_define:nn { NiceMatrix / pNiceArray }
274 {
275   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
276   last-col .code:n = \tl_if_empty:nF {#1}
277   { \@@_error:n { last-col~non~empty~for~NiceArray } }
278   { \int_zero:N \l_@@_last_col_int ,
279   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
280   last-row .int_set:N = \l_@@_last_row_int ,
281   last-row .default:n = -1 ,
282   unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix }
283 }

```


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}`).

```
284 \cs_new_protected:Nn \@@_Cell:
285 {
```

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

```
286 \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.

```
287 \int_compare:nNnT \c@jCol = \c_one_int
288 {
289 \int_compare:nNnT \l_@@_first_col_int = \c_one_int
290 \@@_begin_of_row:
291 }
292 \int_gset:Nn \g_@@_col_total_int
293 { \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).

```
294 \hbox_set:Nw \l_tmpa_box
295 \c_math_toggle_token
296 \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).

```
297 \int_compare:nNnTF \c@iRow = \c_zero_int
298 { \int_compare:nNnT \c@jCol > \c_zero_int \l_@@_code_for_first_row_tl }
299 {
300 \int_compare:nNnT \c@iRow = \l_@@_last_row_int
301 \l_@@_code_for_last_row_tl
302 }
303 }
```

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.

```
304 \cs_new_protected:Nn \@@_begin_of_row:
305 {
306 \int_gincr:N \c@iRow
307 \dim_gset_eq:NN \g_@@_dp_ante_last_row_dim \g_@@_dp_last_row_dim
308 \dim_gzero:N \g_@@_dp_last_row_dim
309 \dim_gzero:N \g_@@_ht_last_row_dim
310 }
```

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.

```
311 \cs_new_protected:Npn \@@_actualization_for_first_and_last_row:
312 {
313 \int_compare:nNnT \c@iRow = \c_zero_int
314 {
315 \dim_gset:Nn \g_@@_dp_row_zero_dim
316 { \dim_max:nn \g_@@_dp_row_zero_dim { \box_dp:N \l_tmpa_box } } }
```

```

317     \dim_gset:Nn \g_@@_ht_row_zero_dim
318     { \dim_max:nn \g_@@_ht_row_zero_dim { \box_ht:N \l_tmpa_box } }
319   }
320   \int_compare:nNnT \c@iRow = \c_one_int
321   {
322     \dim_gset:Nn \g_@@_ht_row_one_dim
323     { \dim_max:nn \g_@@_ht_row_one_dim { \box_ht:N \l_tmpa_box } }
324   }
325   \dim_gset:Nn \g_@@_ht_last_row_dim
326   { \dim_max:nn \g_@@_ht_last_row_dim { \box_ht:N \l_tmpa_box } }
327   \dim_gset:Nn \g_@@_dp_last_row_dim
328   { \dim_max:nn \g_@@_dp_last_row_dim { \box_dp:N \l_tmpa_box } }
329 }
330 \cs_new_protected:Nn \@@_end_Cell:
331 {
332   \c_math_toggle_token
333   \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”).

```

334   \dim_gset:Nn \g_@@_max_cell_width_dim
335   { \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”.

```

336   \@@_actualization_for_first_and_last_row:

```

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

```

337   \tikz
338   [
339     remember~picture ,
340     inner~sep = \c_zero_dim ,
341     minimum~width = \c_zero_dim ,
342     baseline
343   ]
344   \node
345   [
346     anchor = base ,
347     name = nm - \int_use:N \g_@@_env_int -
348             \int_use:N \c@iRow -
349             \int_use:N \c@jCol ,
350     alias =
351       \str_if_empty:NF \l_@@_name_str
352       {
353         \l_@@_name_str -
354         \int_use:N \c@iRow -
355         \int_use:N \c@jCol
356       }
357   ]
358   \bgroup
359   \box_use:N \l_tmpa_box
360   \egroup ;
361 }
362 \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 `\@@_renewcolumntype: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.

```

363 \cs_new_protected:Nn \@@_renewcolumntype:nn
364 {
365   \newcolumntype #1 [ 2 ]

```

```

366 {
367   > {
368     \hbox_set:Nw \l_tmpa_box
369     \@@_Cell:
370   }
371   c
372   < {
373     \@@_end_Cell:
374     \hbox_set_end:
375     #2
376     \hbox_set:Nn \l_tmpb_box
377     { \makebox [ ##2 ] [ ##1 ] { \box_use:N \l_tmpa_box } }
378     \dim_set:Nn \l_tmpa_dim { \box_dp:N \l_tmpb_box }
379     \box_move_down:nn \l_tmpa_dim
380     {
381       \vbox:n
382       {
383         \hbox_to_wd:nn { \box_wd:N \l_tmpb_box }
384         {
385           \hfil
386           \tikz [ remember~picture , overlay ]
387             \coordinate (@@~north~east) ;
388         }
389         \hbox:n
390         {
391           \tikz [ remember~picture , overlay ]
392             \coordinate (@@~south~west) ;
393           \box_move_up:nn \l_tmpa_dim { \box_use:N \l_tmpb_box }
394         }
395       }
396     }

```

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 pdf_latex (why?).

```

397     \tikz [ remember~picture , overlay ]
398     \node
399     [
400       node~contents = { } ,
401       name = nm - \int_use:N \g_@@_env_int -
402               \int_use:N \c@iRow -
403               \int_use:N \c@jCol - w,
404       alias =
405         \str_if_empty:NF \l_@@_name_str
406         {
407           \l_@@_name_str -
408           \int_use:N \c@iRow -
409           \int_use:N \c@jCol - w
410         } ,
411       inner~sep = \c_zero_dim ,
412       fit = (@@~south~west) (@@~north~east)
413     ]
414     ;
415   }
416 }
417 }

```

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.

```
418 \bool_if:NTF \c_@@_draft_bool
419 { \cs_set_protected:Npn \@@_instruction_of_type:n #1 { } }
420 {
421   \cs_new_protected:Npn \@@_instruction_of_type:n #1
422   {
```

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).

```
423     \tl_gput_right:cx
424     { g_@@_ #1 _ lines _ tl }
425     {
426       \use:c { @@ _ draw _ #1 : nn }
427       { \int_use:N \c@iRow }
428       { \int_use:N \c@jCol }
429     }
430   }
431 }
```

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`.

```
432 \cs_new_protected:Npn \@@_array:
433 {
434   \bool_if:NTF \c_@@_revtex_bool
435   {
436     \cs_set_eq:NN \@acol1 \@arrayacol
437     \cs_set_eq:NN \@acolr \@arrayacol
438     \cs_set_eq:NN \@acol \@arrayacol
439     \cs_set:Npn \@halignto { }
440     \@array@array
441   }
442   \array
```

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

```
443   [ \l_@@_pos_env_str ]
444 }
```

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

```
445 \cs_new:Npn \@@_everycr:
446 { \noalign { \@@_everycr_i: } }
447 \cs_new_protected:Npn \@@_everycr_i:
448 {
449   \int_gzero:N \c@jCol
450   \bool_if:NT \l_@@_hlines_bool
451   {
```

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.

```

452     \int_compare:nNnT \c@iRow > { -1 }
453     {
454         \int_compare:nNnF \c@iRow = \l_@@_last_row_int
455         {
456             \hrule \@height \arrayrulewidth
457             \skip_vertical:n { - \arrayrulewidth }
458         }
459     }
460 }
461 }

```

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

```

462 \cs_new_protected:Npn \@@_pre_array:
463 {
464     \cs_if_exist:NT \theiRow
465     { \int_set_eq:NN \l_@@_save_iRow_int \c@iRow }
466     \int_gzero_new:N \c@iRow
467     \cs_if_exist:NT \thejCol
468     { \int_set_eq:NN \l_@@_save_jCol_int \c@jCol }
469     \int_gzero_new:N \c@jCol
470     \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}`).

```

471     \bool_if:NT \l_@@_small_bool
472     {
473         \cs_set:Npn \arraystretch { 0.47 }
474         \dim_set:Nn \arraycolsep { 1.45 pt }
475     }

```

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

```

476     \bool_gset_eq:NN \g_@@_extra_nodes_bool \l_@@_extra_nodes_bool

```

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

```

477     \cs_set:Npn \ialign
478     {
479         \bool_if:NTF \c_@@_colortbl_loaded_bool
480         {
481             \CT@everycr
482             {
483                 \noalign { \cs_gset_eq:NN \CT@row@color \prg_do_nothing: }
484                 \@@_everycr:
485             }
486         }
487         { \everycr { \@@_everycr: } }
488         \tabskip = \c_zero_skip

```

The box `\@arstrutbox` is a box constructed in the beginning of the environment `{array}`. The construction of that box takes into account the current values of `\arraystretch`²⁴ and `\extrarowheight` (of `array`). That box is inserted (via `\@arstrut`) in the beginning of each row of the array. That’s

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

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`.

```

489     \dim_gzero_new:N \g_@@_dp_row_zero_dim
490     \dim_gset:Nn \g_@@_dp_row_zero_dim { \box_dp:N \@arstrutbox }
491     \dim_gzero_new:N \g_@@_ht_row_zero_dim
492     \dim_gset:Nn \g_@@_ht_row_zero_dim { \box_ht:N \@arstrutbox }
493     \dim_gzero_new:N \g_@@_ht_row_one_dim
494     \dim_gset:Nn \g_@@_ht_row_one_dim { \box_ht:N \@arstrutbox }
495     \dim_gzero_new:N \g_@@_dp_ante_last_row_dim
496     \dim_gset:Nn \g_@@_dp_ante_last_row_dim { \box_dp:N \@arstrutbox }
497     \dim_gzero_new:N \g_@@_ht_last_row_dim
498     \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
499     \dim_gzero_new:N \g_@@_dp_last_row_dim
500     \dim_gset:Nn \g_@@_dp_last_row_dim { \box_dp:N \@arstrutbox }

```

After its first utilisation, the definition of `\ialign` will revert automatically to its default definition. With this programming, we will have, in the cells of the array, a clean version of `\ialign`.²⁵

```

501     \cs_set:Npn \ialign
502     {
503         \everycr { }
504         \tabskip = \c_zero_skip
505         \halign
506     }
507     \halign
508 }

```

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}`.

```

509     \newcolumnntype L { > \@@_Cell: l < \@@_end_Cell: }
510     \newcolumnntype C { > \@@_Cell: c < \@@_end_Cell: }
511     \newcolumnntype R { > \@@_Cell: r < \@@_end_Cell: }

512     \cs_set_eq:NN \Ldots \@@_Ldots
513     \cs_set_eq:NN \Cdots \@@_Cdots
514     \cs_set_eq:NN \Vdots \@@_Vdots
515     \cs_set_eq:NN \Ddots \@@_Ddots
516     \cs_set_eq:NN \Iddots \@@_Iddots
517     \cs_set_eq:NN \hdottedline \@@_hdottedline:
518     \cs_set_eq:NN \Hspace \@@_Hspace:
519     \cs_set_eq:NN \Hdotsfor \@@_Hdotsfor:
520     \cs_set_eq:NN \multicolumn \@@_multicolumn:nnn
521     \cs_set_eq:NN \Block \@@_Block:
522     \bool_if:NT \l_@@_renew_dots_bool
523     {
524         \cs_set_eq:NN \ldots \@@_Ldots
525         \cs_set_eq:NN \cdots \@@_Cdots
526         \cs_set_eq:NN \vdots \@@_Vdots
527         \cs_set_eq:NN \ddots \@@_Ddots
528         \cs_set_eq:NN \iddots \@@_Iddots
529         \cs_set_eq:NN \dots \@@_Ldots
530         \cs_set_eq:NN \hdotsfor \@@_Hdotsfor:
531     }

```

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).

```

532     \seq_gclear_new:N \g_@@_multicolumn_cells_seq
533     \seq_gclear_new:N \g_@@_multicolumn_sizes_seq

```

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

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).

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

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

```
535 \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.

```
536 \int_gzero_new:N \g_@@_col_total_int
```

```
537 \cs_set_eq:NN \@ifnextchar \new@ifnextchar
```

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

```
538 \cs_set_eq:NN \NC@find@w \relax
```

```
539 \cs_set_eq:NN \NC@find@W \relax
```

```
540 \@@_renewcolumnntype:nn w { }
```

```
541 \@@_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).

```
542 \tl_set_rescan:Nno
```

```
543 \l_@@_letter_for_dotted_lines_str { } \l_@@_letter_for_dotted_lines_str
```

```
544 \exp_args:NV \newcolumnntype \l_@@_letter_for_dotted_lines_str
```

```
545 {
```

```
546 !
```

```
547 {
```

```
548 \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.

```
549 \bool_if:nF
```

```
550 {
```

```
551 \l_@@_auto_columns_width_bool
```

```
552 || \dim_compare_p:nNn \l_@@_columns_width_dim > \c_zero_dim
```

```
553 }
```

```
554 { \bool_gset_true:N \g_@@_extra_nodes_bool }
```

Consider the following code:

```
\begin{NiceArray}{C:CC:C}
a & b
c & d \\
e & f & g & h \\
i & j & k & l
\end{NiceArray}
```

The first “`:`” in the preamble will be encountered during the first row of the environment `{NiceArray}` but the second one will be encountered only in the third row. We have to issue a command `\vdottedline:n` in the code-after only one time for each “`:`” in the preamble. That's why we keep a counter `\g_@@_last_vdotted_col_int` and with this counter, we know whether a letter “`:`” encountered during the parsing has already been taken into account in the code-after.

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

```
556 {
```

```
557 \int_gset_eq:NN \g_@@_last_vdotted_col_int \c@jCol
```

```
558 \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`.

```

559         { \@@_vdottedline:n { \int_use:N \c@jCol } }
560     }
561 }
562 }
563 \int_gzero_new:N \g_@@_last_vdotted_col_int
564 \bool_if:NT \c_@@_siunitx_loaded_bool \@@_renew_NC@rewrite@S:
565 \int_gset:Nn \g_@@_last_vdotted_col_int { -1 }
566 \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.

```

567 \tl_gclear_new:N \g_@@_Cdots_lines_tl
568 \tl_gclear_new:N \g_@@_Ldots_lines_tl
569 \tl_gclear_new:N \g_@@_Vdots_lines_tl
570 \tl_gclear_new:N \g_@@_Ddots_lines_tl
571 \tl_gclear_new:N \g_@@_Iddots_lines_tl
572 \tl_gclear_new:N \g_@@_Hdotsfor_lines_tl
573 }

```

17.5 The environment `{NiceArrayWithDelims}`

```

574 \NewDocumentEnvironment { NiceArrayWithDelims } { m m O { } m ! O { } }
575 {
576     \str_if_empty:NT \g_@@_type_env_str
577     {
578         \str_gset:Nn \g_@@_type_env_str
579             { environment ~ { NiceArrayWithDelims } }
580     }
581     \@@_adapt_S_column:
582     \@@_test_if_math_mode:
583     \bool_if:NT \l_@@_in_env_bool { \@@_fatal:n { Yet~in~env } }
584     \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).

```

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

```

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

```

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

```

594     \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`.

```

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


```

599 \int_compare:nNnT \l_@@_last_row_int = { -1 }
600 {
601   \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.

```

602   \str_if_empty:NTF \l_@@_name_str
603   {
604     \cs_if_exist:cT { @@_last_row_ \int_use:N \g_@@_env_int }
605     {
606       \int_set:Nn \l_@@_last_row_int
607       { \use:c { @@_last_row_ \int_use:N \g_@@_env_int } }
608     }
609   }
610   {
611     \cs_if_exist:cT { @@_last_row_ \l_@@_name_str }
612     {
613       \int_set:Nn \l_@@_last_row_int
614       { \use:c { @@_last_row_ \l_@@_name_str } }
615     }
616   }
617 }

```

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

```

618 \@@_pre_array:

```

We compute the width of the two delimiters.

```

619 \dim_gzero_new:N \g_@@_left_delim_dim
620 \dim_gzero_new:N \g_@@_right_delim_dim
621 \bool_if:NTF \l_@@_NiceArray_bool
622 {
623   \dim_gset:Nn \g_@@_left_delim_dim { 2 \arraycolsep }
624   \dim_gset:Nn \g_@@_right_delim_dim { 2 \arraycolsep }
625 }
626 {
627   \group_begin:
628   \dim_set_eq:NN \nulldelimiterspace \c_zero_dim
629   \hbox_set:Nn \l_tmpa_box
630   {
631     \c_math_toggle_token
632     \left #1 \vcenter to 3 cm { } \right.
633     \c_math_toggle_token
634   }
635   \dim_gset:Nn \g_@@_left_delim_dim { \box_wd:N \l_tmpa_box }
636   \hbox_set:Nn \l_tmpa_box
637   {
638     \dim_set_eq:NN \nulldelimiterspace \c_zero_dim
639     \c_math_toggle_token
640     \left. \vcenter to 3 cm { } \right #2
641     \c_math_toggle_token
642   }
643   \dim_gset:Nn \g_@@_right_delim_dim { \box_wd:N \l_tmpa_box }
644   \group_end:
645 }
646 }

```

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

```

647 \box_clear_new:N \l_@@_the_array_box

```

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

```

648 \tl_set:Nn \l_tmpa_tl { #4 }
649 \int_compare:nNnTF \l_@@_first_col_int = \c_zero_int
650 { \tl_put_left:NV \l_tmpa_tl \c_@@_preamble_first_col_tl }
651 {
652   \bool_if:NT \l_@@_NiceArray_bool

```

```

653     {
654         \bool_if:NF \l_@@_exterior_arraycolsep_bool
655         { \tl_put_left:Nn \l_tmpa_tl { @ { } } }
656     }
657 }
658 \int_compare:nNnTF \l_@@_last_col_int > { -1 }
659 { \tl_put_right:NV \l_tmpa_tl \c_@@_preamble_last_col_tl }
660 {
661     \bool_if:NT \l_@@_NiceArray_bool
662     {
663         \bool_if:NF \l_@@_exterior_arraycolsep_bool
664         { \tl_put_right:Nn \l_tmpa_tl { @ { } } }
665     }
666 }

```

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).

```

667     \hbox_set:Nw \l_@@_the_array_box
668     \skip_horizontal:n \l_@@_left_margin_dim
669     \skip_horizontal:n \l_@@_extra_left_margin_dim
670     \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`).

```

671     \exp_args:NV \@@_array: \l_tmpa_tl
672 }

```

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`).

```

673 {
674     \bool_if:nT
675     {
676         \l_@@_auto_columns_width_bool
677         || \dim_compare_p:nNn \l_@@_columns_width_dim > \c_zero_dim
678     }
679     {
680         \crrc
681         \int_compare:nNnTF \l_@@_first_col_int = 0 { \omit & }
682         \omit

```

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

```

683         \skip_horizontal:N \arraycolsep
684         \tikz [ remember~picture , overlay ]
685         \coordinate [ name = nm - \int_use:N \g_@@_env_int - col - 0 ] ;
686         \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.

```

687     \bool_if:nTF
688     {
689         \l_@@_auto_columns_width_bool
690         && ! \l_@@_block_auto_columns_width_bool
691     }
692     {
693         \dim_gset:Nn \g_tmpa_dim
694         { \g_@@_max_cell_width_dim + 2 \arraycolsep }
695     }
696 {

```

```

697         \dim_gset:Nn \g_tmpa_dim
698         { \l_@@_columns_width_dim + 2 \arraycolsep }
699     }
700     \skip_horizontal:N \g_tmpa_dim
701     \tikz [ remember~picture , overlay ]
702     \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.

```

703     \int_gset:Nn \g_tmpa_int 1
704     \bool_if:nTF \g_@@_last_col_found_bool
705     { \prg_replicate:nn { \g_@@_col_total_int - 3 } }
706     { \prg_replicate:nn { \g_@@_col_total_int - 2 } }
707     {
708         &
709         \omit

```

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

```

710         \int_gincr:N \g_tmpa_int
711         \skip_horizontal:N \g_tmpa_dim

```

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

```

712         \tikz [ remember~picture , overlay ]
713         \coordinate
714         [
715             name = nm - \int_use:N \g_@@_env_int -
716                 col - \int_use:N \g_tmpa_int
717         ] ;
718     }

```

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

```

719     &
720     \omit
721     \int_gincr:N \g_tmpa_int
722     \skip_horizontal:N \g_tmpa_dim
723     \skip_horizontal:n { - \arraycolsep }
724     \tikz [ remember~picture , overlay ]
725     \coordinate
726     [
727         name = nm - \int_use:N \g_@@_env_int -
728             col - \int_use:N \g_tmpa_int
729     ] ;
730     \skip_horizontal:N \arraycolsep
731 }
732 \endarray
733 \c_math_toggle_token
734 \skip_horizontal:n \l_@@_right_margin_dim
735 \skip_horizontal:n \l_@@_extra_right_margin_dim
736 \hbox_set_end:

737 \int_compare:nNnT \l_@@_last_row_int > { -2 }
738 {
739     \bool_if:NF \l_@@_last_row_without_value_bool
740     {
741         \int_compare:nNnF \l_@@_last_row_int = \c_iRow
742         {
743             \@@_error:n { Wrong~last~row }
744             \int_gset_eq:NN \l_@@_last_row_int \c_iRow
745         }
746     }
747 }

```

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).

```

748     \int_compare:nNnTF \l_@@_first_row_int = \c_zero_int

```

```

749 {
750   \dim_set:Nn \l_tmpa_dim
751   { \g_@@_dp_row_zero_dim + \lineskip + \g_@@_ht_row_zero_dim }
752 }
753 { \dim_zero:N \l_tmpa_dim }

```

We compute `\l_tmpb_dim` which is the vertical dimension of the “last row” below the array (when the key `last-row` is used). A value of `-2` for `\l_@@_last_row_int` means that there is no “last row”.²⁶

```

754 \int_compare:nNnTF \l_@@_last_row_int > { -2 }
755 {
756   \dim_set:Nn \l_tmpb_dim
757   { \g_@@_ht_last_row_dim + \lineskip + \g_@@_dp_last_row_dim }
758 }
759 { \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).

```

760 \int_compare:nNnT \l_@@_first_col_int = \c_zero_int
761 {
762   \skip_horizontal:n \arraycolsep
763   \skip_horizontal:n \g_@@_width_first_col_dim
764 }

```

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}`.

```

765 \bool_if:NTF \l_@@_NiceArray_bool
766 {
767   \int_compare:nNnT \l_@@_first_row_int = \c_zero_int
768   {
769     \str_if_eq:VnTF \l_@@_pos_env_str { t }
770     {
771       \box_move_up:nn
772       { \l_tmpa_dim - \g_@@_ht_row_zero_dim + \g_@@_ht_row_one_dim }
773     }
774   }
775   {
776     \int_compare:nNnT \l_@@_last_row_int > 0
777     {
778       \str_if_eq:VnT \l_@@_pos_env_str { b }
779       {
780         \box_move_down:nn
781         {
782           \l_tmpb_dim
783           - \g_@@_dp_last_row_dim + \g_@@_dp_ante_last_row_dim
784         }
785       }
786     }
787   }
788   { \box_use_drop:N \l_@@_the_array_box }
789 }

```

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

```

790 {
791   \hbox_set:Nn \l_tmpa_box
792   {
793     \c_math_toggle_token
794     \left #1
795     \vcenter
796     {

```

²⁶A value of `-1` for `\l_@@_last_row_int` means that there is a “last row” but the number of that row is unknown (the user have not set the value with the option `last row`).

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

```

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

```

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

```

804         \skip_vertical:n { - \l_tmpb_dim }
805     }
806     \right #2
807     \c_math_toggle_token
808 }
809 \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }
810 \box_set_dp:Nn \l_tmpa_box { \box_dp:N \l_tmpa_box + \l_tmpb_dim }
811 \box_use_drop:N \l_tmpa_box
812 }

```

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. 47).

```

813 \bool_if:NT \g_@@_last_col_found_bool
814 {
815     \skip_horizontal:n \g_@@_width_last_col_dim
816     \skip_horizontal:n \arraycolsep
817 }
818 \@@_after_array:
819 }

```

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

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

```

820 \tl_const:Nn \c_@@_preamble_first_col_tl
821 {
822     >
823     {
824         \@@_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.

```

825         \hbox_set:Nw \l_tmpa_box
826         \c_math_toggle_token
827         \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”.

```

828         \bool_if:nT
829         {
830             \int_compare_p:nNn \c@iRow > \c_zero_int
831             &&
832             (
833                 \int_compare_p:nNn \l_@@_last_row_int < 0
834                 ||
835                 \int_compare_p:nNn \c@iRow < \l_@@_last_row_int
836             )
837         }
838         { \l_@@_code_for_first_col_tl }
839     }
840     1
841     <
842     {
843         \c_math_toggle_token
844         \hbox_set_end:
845         \@@_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.

```

846     \dim_gset:Nn \g_@@_width_first_col_dim
847     {
848         \dim_max:nn
849         \g_@@_width_first_col_dim
850         { \box_wd:N \l_tmpa_box }
851     }

```

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

```

852     \hbox_overlap_left:n
853     {
854         \tikz
855         [
856             remember~picture ,
857             inner~sep = \c_zero_dim ,
858             minimum~width = \c_zero_dim ,
859             baseline
860         ]
861         \node
862         [
863             anchor = base ,
864             name =
865                 nm -
866                 \int_use:N \g_@@_env_int -
867                 \int_use:N \c@iRow -
868                 0 ,
869             alias =
870                 \str_if_empty:NF \l_@@_name_str
871                 {
872                     \l_@@_name_str -
873                     \int_use:N \c@iRow -
874                     0
875                 }
876             { \box_use:N \l_tmpa_box } ;
877         \skip_horizontal:n
878         {
879             \g_@@_left_delim_dim +
880             \l_@@_left_margin_dim +
881             \l_@@_extra_left_margin_dim
882         }
883     }
884 }
885 \skip_horizontal:n { - 2 \arraycolsep }
886 }
887 }

```

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

```

888 \tl_const:Nn \c_@@_preamble_last_col_tl
889 {
890     >
891     {

```

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

```

892         \bool_gset_true:N \g_@@_last_col_found_bool
893         \int_gincr:N \c@jCol
894         \int_gset:Nn \g_@@_col_total_int
895         { \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.

```

896         \hbox_set:Nw \l_tmpa_box
897         \c_math_toggle_token
898         \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”.

```

899     \bool_if:nT
900     {
901         \int_compare_p:nNn \c@iRow > \c_zero_int
902         &&
903         (
904             \int_compare_p:nNn \l_@@_last_row_int < 0
905             ||
906             \int_compare_p:nNn \c@iRow < \l_@@_last_row_int
907         )
908     }
909     { \l_@@_code_for_last_col_tl }
910 }
911 l
912 <
913 {
914     \c_math_toggle_token
915     \hbox_set_end:
916     \@@_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.

```

917     \dim_gset:Nn \g_@@_width_last_col_dim
918     {
919         \dim_max:nn
920         \g_@@_width_last_col_dim
921         { \box_wd:N \l_tmpa_box }
922     }
923     \skip_horizontal:n { - 2 \arraycolsep }

```

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

```

924     \hbox_overlap_right:n
925     {
926         \skip_horizontal:n
927         {
928             \g_@@_right_delim_dim +
929             \l_@@_right_margin_dim +
930             \l_@@_extra_right_margin_dim
931         }
932         \tikz
933         [
934             remember~picture ,
935             inner~sep = \c_zero_dim ,
936             minimum~width = \c_zero_dim ,
937             baseline
938         ]
939         \node
940         [
941             anchor = base ,
942             name =
943             nm -
944             \int_use:N \g_@@_env_int -
945             \int_use:N \c@iRow -
946             \int_use:N \c@jCol ,
947             alias =
948             \str_if_empty:NF \l_@@_name_str
949             {
950                 \l_@@_name_str -
951                 \int_use:N \c@iRow -
952                 \int_use:N \c@jCol
953             }
954         ]
955         { \box_use:N \l_tmpa_box } ;
956     }
957 }
958 }

```

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.

```

959 \NewDocumentEnvironment { NiceArray } { }
960 {
961   \bool_set_true:N \l_@@_NiceArray_bool
962   \str_if_empty:NT \g_@@_type_env_str
963   { \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).

```

964   \NiceArrayWithDelims . .
965 }
966 { \endNiceArrayWithDelims }

```

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

```

967 \NewDocumentEnvironment { pNiceArray } { }
968 {
969   \str_if_empty:NT \g_@@_type_env_str
970   { \str_gset:Nn \g_@@_type_env_str { environment ~ { pNiceArray } } }
971   \@@_test_if_math_mode:
972   \NiceArrayWithDelims ( )
973 }
974 { \endNiceArrayWithDelims }

975 \NewDocumentEnvironment { bNiceArray } { }
976 {
977   \str_if_empty:NT \g_@@_type_env_str
978   { \str_gset:Nn \g_@@_type_env_str { environment ~ { bNiceArray } } }
979   \@@_test_if_math_mode:
980   \NiceArrayWithDelims [ ]
981 }
982 { \endNiceArrayWithDelims }

983 \NewDocumentEnvironment { BNiceArray } { }
984 {
985   \str_if_empty:NT \g_@@_type_env_str
986   { \str_gset:Nn \g_@@_type_env_str { environment ~ { BNiceArray } } }
987   \@@_test_if_math_mode:
988   \NiceArrayWithDelims \{ \}
989 }
990 { \endNiceArrayWithDelims }

991 \NewDocumentEnvironment { vNiceArray } { }
992 {
993   \str_if_empty:NT \g_@@_type_env_str
994   { \str_gset:Nn \g_@@_type_env_str { environment ~ { vNiceArray } } }
995   \@@_test_if_math_mode:
996   \NiceArrayWithDelims | |
997 }
998 { \endNiceArrayWithDelims }

999 \NewDocumentEnvironment { VNiceArray } { }
1000 {
1001   \str_if_empty:NT \g_@@_type_env_str
1002   { \str_gset:Nn \g_@@_type_env_str { environment ~ { VNiceArray } } }
1003   \@@_test_if_math_mode:
1004   \NiceArrayWithDelims \| \|
1005 }
1006 { \endNiceArrayWithDelims }

```

17.6 The environment `{NiceMatrix}` and its variants

```

1007 \cs_new_protected:Npn \@@_define_env:n #1

```



```

1008 {
1009   \NewDocumentEnvironment { #1 NiceMatrix } { ! O { } }
1010   {
1011     \str_gset:Nn \g_@@_type_env_str { environment ~ { #1 NiceMatrix } }
1012     \keys_set:nn { NiceMatrix / NiceMatrix } { ##1 }
1013     \begin { #1 NiceArray }
1014       {
1015         *
1016         {
1017           \int_compare:nNnTF \l_@@_last_col_int = { -1 }
1018             \c@MaxMatrixCols
1019             { \int_eval:n { \l_@@_last_col_int - 1 } }
1020         }
1021         C
1022       }
1023     }
1024     { \end { #1 NiceArray } }
1025   }
1026   \@@_define_env:n { }
1027   \@@_define_env:n p
1028   \@@_define_env:n b
1029   \@@_define_env:n B
1030   \@@_define_env:n v
1031   \@@_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.

```

1032 \prg_set_conditional:Npnn \@@_if_not_empty_cell:nn #1 #2 { T , TF }
1033 {

```

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.

```

1034   \bool_set_false:N \l_tmpa_bool
1035   \cs_if_exist:cTF
1036     { @@ _ dotted _ \int_use:N #1 - \int_use:N #2 }
1037     \prg_return_true:
1038     {

```

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.

```

1039   \cs_if_free:cTF
1040   {
1041     pgf@sh@ns@nm -
1042     \int_use:N \g_@@_env_int -
1043     \int_use:N #1 -
1044     \int_use:N #2
1045   }
1046   { \prg_return_false: }
1047   {

```

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.

```

1048         \bool_set_false:N \l_tmpa_bool
1049         \cs_if_exist:cT
1050         { @@ _ empty _ \int_use:N #1 - \int_use:N #2 }
1051         {
1052             \int_compare:nNnT
1053             { \use:c { @@ _ empty _ \int_use:N #1 - \int_use:N #2 } }
1054             =
1055             \g_@@_env_int
1056             { \bool_set_true:N \l_tmpa_bool }
1057         }
1058         \bool_if:NTF \l_tmpa_bool
1059         \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.

```

1060         {
1061             \begin { pgfpicture }

```

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

```

1062             \tl_set:Nx \l_tmpa_tl
1063             {
1064                 nm -
1065                 \int_use:N \g_@@_env_int -
1066                 \int_use:N #1 -
1067                 \int_use:N #2
1068             }
1069             \pgfpointanchor \l_tmpa_tl { east }
1070             \dim_gset:Nn \g_tmpa_dim \pgf@x
1071             \pgfpointanchor \l_tmpa_tl { west }
1072             \dim_gset:Nn \g_tmpb_dim \pgf@x
1073             \end { pgfpicture }
1074             \dim_compare:nNnTF
1075             { \dim_abs:n { \g_tmpb_dim - \g_tmpa_dim } } < { 0.5 pt }
1076             \prg_return_false:
1077             \prg_return_true:
1078         }
1079     }
1080 }
1081 }

```

17.8 After the construction of the array

```

1082 \cs_new_protected:Nn \@@_after_array:
1083 {
1084     \int_compare:nNnTF \c@iRow > \c_zero_int
1085     \@@_after_array_i:
1086     {
1087         \@@_error:n { Zero~row }
1088         \@@_restore_iRow_jCol:
1089     }
1090 }

```

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

```

1091 \cs_new_protected:Nn \@@_after_array_i:
1092 {

```

```

1093 \group_begin:
1094 \cs_if_exist:NT \tikz@library@external@loaded
1095 { \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”.²⁷

```

1096 \int_gset_eq:NN \c@jCol \g_@@_col_total_int
1097 \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.

```

1098 \int_gset_eq:NN \g_@@_row_total_int \c@iRow
1099 \int_compare:nNnT \l_@@_last_row_int > { -1 }
1100 { \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.

```

1101 \bool_if:NT \l_@@_last_row_without_value_bool
1102 {
1103   \iow_now:Nn \@mainaux \ExplSyntaxOn
1104   \iow_now:Nx \@mainaux
1105   {
1106     \cs_gset:cpn { @@_last_row_ \int_use:N \g_@@_env_int }
1107     { \int_use:N \g_@@_row_total_int }
1108   }

```

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.

```

1109   \str_if_empty:NF \l_@@_name_str
1110   {
1111     \iow_now:Nx \@mainaux
1112     {
1113       \cs_gset:cpn { @@_last_row_ \l_@@_name_str }
1114       { \int_use:N \g_@@_row_total_int }
1115     }
1116   }
1117   \iow_now:Nn \@mainaux \ExplSyntaxOff
1118 }

```

By default, the diagonal lines will be parallelized²⁸. There are two types of diagonals lines: the `\Ddots` diagonals and the `\Iddots` diagonals. We have to count both types in order to know whether a diagonal is the first of its type in the current `{NiceArray}` environment.

```

1119 \bool_if:NT \l_@@_parallelize_diags_bool
1120 {
1121   \int_zero_new:N \l_@@_ddots_int
1122   \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.

```

1123   \dim_zero_new:N \l_@@_delta_x_one_dim
1124   \dim_zero_new:N \l_@@_delta_y_one_dim
1125   \dim_zero_new:N \l_@@_delta_x_two_dim
1126   \dim_zero_new:N \l_@@_delta_y_two_dim
1127 }

```

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

```

1128 \bool_if:NT \g_@@_extra_nodes_bool \@@_create_extra_nodes:
1129 \int_zero_new:N \l_@@_initial_i_int
1130 \int_zero_new:N \l_@@_initial_j_int
1131 \int_zero_new:N \l_@@_final_i_int
1132 \int_zero_new:N \l_@@_final_j_int

```

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

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

```

1133 \bool_set_false:N \l_@@_initial_open_bool
1134 \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.
1135 \bool_if:NT \l_@@_small_bool
1136 {
1137     \dim_set:Nn \l_@@_radius_dim { 0.37 pt }
1138     \dim_set:Nn \l_@@_inter_dots_dim { 0.25 em }
1139 }

```

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

```

1140 \g_@@_Hdotsfor_lines_tl
1141 \g_@@_Vdots_lines_tl
1142 \g_@@_Ddots_lines_tl
1143 \g_@@_Iddots_lines_tl
1144 \g_@@_Cdots_lines_tl
1145 \g_@@_Ldots_lines_tl

```

Now, the code-after.

```

1146 \tikzset
1147 {
1148     every-picture / .style =
1149     {
1150         overlay ,
1151         remember-picture ,
1152         name-prefix = nm - \int_use:N \g_@@_env_int -
1153     }
1154 }
1155 \cs_set_eq:NN \line \@@_line:nn
1156 \g_@@_code_after_tl
1157 \tl_gclear:N \g_@@_code_after_tl
1158 \group_end:
1159 \str_gclear:N \g_@@_type_env_str
1160 \@@_restore_iRow_jCol:
1161 }
1162 \cs_new_protected:Nn \@@_restore_iRow_jCol:
1163 {
1164     \cs_if_exist:NT \theiRow { \int_gset_eq:NN \c@iRow \l_@@_save_iRow_int }
1165     \cs_if_exist:NT \thejCol { \int_gset_eq:NN \c@jCol \l_@@_save_jCol_int }
1166 }

```

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

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

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

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

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

- the first argument is the row of the cell where the command was issued;
- the second argument is the column of the cell where the command was issued;

- the third argument is the x -value of the orientation vector of the line;
- the fourth argument is the y -value of the orientation vector of the line;

This command computes:

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

```
1167 \cs_new_protected:Nn \@@_find_extremities_of_line:nnnn
1168 {
```

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

```
1169   \cs_set:cpn { @@ _ dotted _ #1 - #2 } { }
```

Initialization of variables.

```
1170   \int_set:Nn \l_@@_initial_i_int { #1 }
1171   \int_set:Nn \l_@@_initial_j_int { #2 }
1172   \int_set:Nn \l_@@_final_i_int { #1 }
1173   \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.

```
1174   \bool_set_false:N \l_@@_stop_loop_bool
1175   \bool_do_until:Nn \l_@@_stop_loop_bool
1176   {
1177     \int_add:Nn \l_@@_final_i_int { #3 }
1178     \int_add:Nn \l_@@_final_j_int { #4 }
```

We test if we are still in the matrix.

```
1179     \bool_set_false:N \l_@@_final_open_bool
1180     \int_compare:nNnTF \l_@@_final_i_int > \c@iRow
1181     {
1182       \int_compare:nNnT { #3 } = 1
1183       { \bool_set_true:N \l_@@_final_open_bool }
1184     }
1185     {
1186       \int_compare:nNnTF \l_@@_final_j_int < 1
1187       {
1188         \int_compare:nNnT { #4 } = { -1 }
1189         { \bool_set_true:N \l_@@_final_open_bool }
1190       }
1191       {
1192         \int_compare:nNnT \l_@@_final_j_int > \c@jCol
1193         {
1194           \int_compare:nNnT { #4 } = 1
1195           { \bool_set_true:N \l_@@_final_open_bool }
1196         }
1197       }
1198     }
1199     \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.

```
1200   {
```

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).

```
1201     \int_sub:Nn \l_@@_final_i_int { #3 }
1202     \int_sub:Nn \l_@@_final_j_int { #4 }
1203     \bool_set_true:N \l_@@_stop_loop_bool
1204   }
```

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`.

```

1205     {
1206         \@@_if_not_empty_cell:nNTF \l_@@_final_i_int \l_@@_final_j_int
1207         { \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.

```

1208     {
1209         \cs_set:cpn
1210         {
1211             @@_dotted_
1212             \int_use:N \l_@@_final_i_int -
1213             \int_use:N \l_@@_final_j_int
1214         }
1215         { }
1216     }
1217 }
1218 }

```

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).

```

1219 \cs_if_free:cT
1220 {
1221     pgf@sh@ns@nm -
1222     \int_use:N \g_@@_env_int -
1223     \int_use:N \l_@@_final_i_int -
1224     \int_use:N \l_@@_final_j_int
1225 }
1226 {
1227     \bool_if:NF \l_@@_final_open_bool
1228     {
1229         \msg_error:nmx { nicematrix } { Impossible~line }
1230         { \int_use:N \l_@@_final_i_int }
1231         \bool_set_true:N \l_@@_impossible_line_bool
1232     }
1233 }

```

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

```

1234 \bool_set_false:N \l_@@_stop_loop_bool
1235 \bool_do_until:Nn \l_@@_stop_loop_bool
1236 {
1237     \int_sub:Nn \l_@@_initial_i_int { #3 }
1238     \int_sub:Nn \l_@@_initial_j_int { #4 }
1239     \bool_set_false:N \l_@@_initial_open_bool
1240     \int_compare:nNnTF \l_@@_initial_i_int < 1
1241     {
1242         \int_compare:nNnT { #3 } = 1
1243         { \bool_set_true:N \l_@@_initial_open_bool }
1244     }
1245     {
1246         \int_compare:nNnTF \l_@@_initial_j_int < 1
1247         {
1248             \int_compare:nNnT { #4 } = 1
1249             { \bool_set_true:N \l_@@_initial_open_bool }
1250         }
1251         {
1252             \int_compare:nNnT \l_@@_initial_j_int > \c@jCol

```

```

1253         {
1254             \int_compare:nNtT { #4 } = { -1 }
1255             { \bool_set_true:N \l_@@_initial_open_bool }
1256         }
1257     }
1258 }
1259 \bool_if:NTF \l_@@_initial_open_bool
1260 {
1261     \int_add:Nn \l_@@_initial_i_int { #3 }
1262     \int_add:Nn \l_@@_initial_j_int { #4 }
1263     \bool_set_true:N \l_@@_stop_loop_bool
1264 }
1265 {
1266     \@@_if_not_empty_cell:nnTF
1267     \l_@@_initial_i_int \l_@@_initial_j_int
1268     { \bool_set_true:N \l_@@_stop_loop_bool }
1269     {
1270         \cs_set:cpn
1271         {
1272             @@ _ dotted _
1273             \int_use:N \l_@@_initial_i_int -
1274             \int_use:N \l_@@_initial_j_int
1275         }
1276         { }
1277     }
1278 }
1279 }

```

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).

```

1280 \cs_if_free:cT
1281 {
1282     pgf@sh@ns@nm -
1283     \int_use:N \g_@@_env_int -
1284     \int_use:N \l_@@_initial_i_int -
1285     \int_use:N \l_@@_initial_j_int
1286 }
1287 {
1288     \bool_if:NF \l_@@_initial_open_bool
1289     {
1290         \msg_error:nnx { nicematrix } { Impossible~line }
1291         { \int_use:N \l_@@_initial_i_int }
1292         \bool_set_true:N \l_@@_impossible_line_bool
1293     }
1294 }

```

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

```

1295 \bool_if:nT \l_@@_initial_open_bool \@@_create_extra_nodes:
1296 \bool_if:nT \l_@@_final_open_bool \@@_create_extra_nodes:
1297 }

```

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

²⁹We should change this. Indeed, for an open extremity of an *horizontal* dotted line, we use the `w` node, if it exists, and not the “medium node”.

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

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}`.

```

1298 \cs_new_protected:Nn \@@_retrieve_coords:nn
1299 {
1300   \dim_gzero_new:N \g_@@_x_initial_dim
1301   \dim_gzero_new:N \g_@@_y_initial_dim
1302   \dim_gzero_new:N \g_@@_x_final_dim
1303   \dim_gzero_new:N \g_@@_y_final_dim
1304   \begin { tikzpicture } [ remember~picture ]
1305     \tikz@parse@node \pgfutil@firstofone
1306       ( nm - \int_use:N \g_@@_env_int -
1307         \int_use:N \l_@@_initial_i_int -
1308         \int_use:N \l_@@_initial_j_int #1 )
1309     \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1310     \dim_gset:Nn \g_@@_y_initial_dim \pgf@y
1311     \tikz@parse@node \pgfutil@firstofone
1312       ( nm - \int_use:N \g_@@_env_int -
1313         \int_use:N \l_@@_final_i_int -
1314         \int_use:N \l_@@_final_j_int #2 )
1315     \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1316     \dim_gset:Nn \g_@@_y_final_dim \pgf@y
1317   \end { tikzpicture }
1318 }
1319 \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`).

```

1320 \cs_new_protected:Nn \@@_adjust_with_col_nodes:
1321 {
1322   \bool_if:NT \l_@@_initial_open_bool
1323   {
1324     \cs_if_exist:cT
1325       { pgf@sh@ns@nm - \int_use:N \g_@@_env_int - col - 0 }
1326     {
1327       \begin { tikzpicture } [ remember~picture ]
1328         \tikz@parse@node \pgfutil@firstofone
1329           ( nm - \int_use:N \g_@@_env_int - col - 0 )
1330         \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1331         \end { tikzpicture }
1332       }
1333     }
1334   \bool_if:NT \l_@@_final_open_bool
1335   {
1336     \cs_if_exist:cT
1337       {
1338         pgf@sh@ns@nm - \int_use:N \g_@@_env_int - col -
1339         \int_use:N \c@jCol
1340       }
1341     {
1342       \begin { tikzpicture } [ remember~picture ]
1343         \tikz@parse@node \pgfutil@firstofone
1344           ( nm - \int_use:N \g_@@_env_int - col - \int_use:N \c@jCol )
1345         \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1346         \end { tikzpicture }
1347       }
1348     }
1349   }

```



```

1350 \cs_new_protected:Nn \@@_draw_Ldots:nn
1351 {
1352   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1353   {
1354     \bool_set_false:N \l_@@_impossible_line_bool
1355     \@@_find_extremities_of_line:nnnn { #1 } { #2 } \c_zero_int \c_one_int
1356     \bool_if:NF \l_@@_impossible_line_bool \@@_actually_draw_Ldots:
1357   }
1358 }

```

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`.

```

1359 \cs_new_protected:Nn \@@_actually_draw_Ldots:
1360 {
1361   \@@_retrieve_coords:xx
1362   {
1363     \bool_if:NTF \l_@@_initial_open_bool
1364     {

```

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

```

1365       \cs_if_exist:cTF
1366       {
1367         pgf@sh@ns@nm
1368         - \int_use:N \g_@@_env_int
1369         - \int_use:N \l_@@_initial_i_int
1370         - \int_use:N \l_@@_initial_j_int - w
1371       }
1372       { - w.base-west }
1373       { - medium.base-west }
1374     }
1375     { .base-east }
1376   }
1377   {
1378     \bool_if:NTF \l_@@_final_open_bool
1379     {
1380       \cs_if_exist:cTF
1381       {
1382         pgf@sh@ns@nm
1383         - \int_use:N \g_@@_env_int
1384         - \int_use:N \l_@@_final_i_int
1385         - \int_use:N \l_@@_final_j_int - w
1386       }
1387       { - w.base-east }
1388       { - medium.base-east }
1389     }
1390     { .base-west }
1391   }
1392   \@@_adjust_with_col_nodes:
1393   \bool_if:NT \l_@@_initial_open_bool
1394   { \dim_gset_eq:NN \g_@@_y_initial_dim \g_@@_y_final_dim }
1395   \bool_if:NT \l_@@_final_open_bool
1396   { \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.

```

1397   \dim_gadd:Nn \g_@@_y_initial_dim { 0.53 pt }
1398   \dim_gadd:Nn \g_@@_y_final_dim { 0.53 pt }
1399   \@@_draw_tikz_line:
1400 }

```

```

1401 \cs_new_protected:Nn \@@_draw_Cdots:nn
1402 {

```

```

1403 \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1404 {
1405   \bool_set_false:N \l_@@_impossible_line_bool
1406   \@@_find_extremities_of_line:nnnn { #1 } { #2 } \c_zero_int \c_one_int
1407   \bool_if:NF \l_@@_impossible_line_bool
1408   {
1409     \@@_retrieve_coords:xx
1410     {
1411       \bool_if:NTF \l_@@_initial_open_bool
1412       {
1413         \cs_if_exist:cTF
1414         {
1415           pgf@sh@ns@nm
1416           - \int_use:N \g_@@_env_int
1417           - \int_use:N \l_@@_initial_i_int
1418           - \int_use:N \l_@@_initial_j_int - w
1419         }
1420         { - w.mid~west }
1421         { - medium.mid~west }
1422       }
1423       { .mid~east }
1424     }
1425     {
1426       \bool_if:NTF \l_@@_final_open_bool
1427       {
1428         \cs_if_exist:cTF
1429         {
1430           pgf@sh@ns@nm
1431           - \int_use:N \g_@@_env_int
1432           - \int_use:N \l_@@_final_i_int
1433           - \int_use:N \l_@@_final_j_int - w
1434         }
1435         { - w.mid~east }
1436         { - medium.mid~east }
1437       }
1438       { .mid~west }
1439     }
1440     \@@_adjust_with_col_nodes:
1441     \bool_if:NT \l_@@_initial_open_bool
1442     { \dim_gset_eq:NN \g_@@_y_initial_dim \g_@@_y_final_dim }
1443     \bool_if:NT \l_@@_final_open_bool
1444     { \dim_gset_eq:NN \g_@@_y_final_dim \g_@@_y_initial_dim }
1445     \@@_draw_tikz_line:
1446   }
1447 }
1448 }

```

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?).

```

1449 \cs_new_protected:Nn \@@_draw_Vdots:nn
1450 {
1451   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1452   {
1453     \bool_set_false:N \l_@@_impossible_line_bool
1454     \@@_find_extremities_of_line:nnnn { #1 } { #2 } \c_one_int \c_zero_int
1455     \bool_if:NF \l_@@_impossible_line_bool
1456     {
1457       \@@_retrieve_coords:xx
1458       {
1459         \bool_if:NTF \l_@@_initial_open_bool
1460         { - medium.north~west }
1461         { .south~west }

```

```

1462     }
1463     {
1464         \bool_if:NTF \l_@@_final_open_bool
1465         { - medium.south~west }
1466         { .north~west }
1467     }

```

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

```

1468     \bool_set:Nn \l_tmpa_bool
1469     { \dim_compare_p:nNn \g_@@_x_initial_dim = \g_@@_x_final_dim }
1470     \@@_retrieve_coords:xx
1471     {
1472         \bool_if:NTF \l_@@_initial_open_bool
1473         { - medium.north }
1474         { .south }
1475     }
1476     {
1477         \bool_if:NTF \l_@@_final_open_bool
1478         { - medium.south }
1479         { .north }
1480     }

```

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

```

1481     \bool_set:Nn \l_tmpb_bool
1482     { \dim_compare_p:nNn \g_@@_x_initial_dim = \g_@@_x_final_dim }
1483     \bool_if:NF \l_tmpb_bool
1484     {
1485         \dim_gset:Nn \g_@@_x_initial_dim
1486         {
1487             \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
1488             \g_@@_x_initial_dim \g_@@_x_final_dim
1489         }
1490         \dim_gset_eq:NN \g_@@_x_final_dim \g_@@_x_initial_dim
1491     }
1492     \@@_draw_tikz_line:
1493 }
1494 }
1495 }

```

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.

```

1496 \cs_new_protected:Nn \@@_draw_Ddots:nn
1497 {
1498     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1499     {
1500         \bool_set_false:N \l_@@_impossible_line_bool
1501         \@@_find_extremities_of_line:nnnn { #1 } { #2 } \c_one_int \c_one_int
1502         \bool_if:NF \l_@@_impossible_line_bool
1503         {
1504             \@@_retrieve_coords:xx
1505             {
1506                 \bool_if:NTF \l_@@_initial_open_bool
1507                 { - medium.north~west }
1508                 { .south~east }
1509             }
1510             {
1511                 \bool_if:NTF \l_@@_final_open_bool
1512                 { - medium.south~east }
1513                 { .north~west }
1514             }

```

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.

```

1515         \bool_if:NT \l_@@_parallelize_diags_bool
1516         {
1517             \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).

```

1518         \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.

```

1519         {
1520             \dim_set:Nn \l_@@_delta_x_one_dim
1521             { \g_@@_x_final_dim - \g_@@_x_initial_dim }
1522             \dim_set:Nn \l_@@_delta_y_one_dim
1523             { \g_@@_y_final_dim - \g_@@_y_initial_dim }
1524         }

```

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`.

```

1525         {
1526             \dim_gset:Nn \g_@@_y_final_dim
1527             {
1528                 \g_@@_y_initial_dim +
1529                 ( \g_@@_x_final_dim - \g_@@_x_initial_dim ) *
1530                 \dim_ratio:nn \l_@@_delta_y_one_dim \l_@@_delta_x_one_dim
1531             }
1532         }
1533     }

```

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

```

1534         \@@_draw_tikz_line:
1535     }
1536 }
1537 }

```

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

```

1538 \cs_new_protected:Nn \@@_draw_Iddots:nn
1539 {
1540     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1541     {
1542         \bool_set_false:N \l_@@_impossible_line_bool
1543         \@@_find_extremities_of_line:nnnn { #1 } { #2 } 1 { -1 }
1544         \bool_if:NF \l_@@_impossible_line_bool
1545         {
1546             \@@_retrieve_coords:xx
1547             {
1548                 \bool_if:NTF \l_@@_initial_open_bool
1549                 { - medium.north-east }
1550                 { .south-west }
1551             }
1552             {
1553                 \bool_if:NTF \l_@@_final_open_bool
1554                 { - medium.south-west }
1555                 { .north-east }
1556             }
1557             \bool_if:NT \l_@@_parallelize_diags_bool
1558             {
1559                 \int_incr:N \l_@@_iddots_int
1560                 \int_compare:nNnTF \l_@@_iddots_int = \c_one_int
1561                 {
1562                     \dim_set:Nn \l_@@_delta_x_two_dim
1563                     { \g_@@_x_final_dim - \g_@@_x_initial_dim }

```

```

1564         \dim_set:Nn \l_@@_delta_y_two_dim
1565         { \g_@@_y_final_dim - \g_@@_y_initial_dim }
1566     }
1567     {
1568         \dim_gset:Nn \g_@@_y_final_dim
1569         {
1570             \g_@@_y_initial_dim +
1571             ( \g_@@_x_final_dim - \g_@@_x_initial_dim ) *
1572             \dim_ratio:nn \l_@@_delta_y_two_dim \l_@@_delta_x_two_dim
1573         }
1574     }
1575 }
1576 \@@_draw_tikz_line:
1577 }
1578 }
1579 }

```

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`.

```

1580 \cs_new_protected:Nn \@@_draw_tikz_line:
1581 {

```

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.

```

1582     \dim_zero_new:N \l_@@_l_dim
1583     \dim_set:Nn \l_@@_l_dim
1584     {
1585         \fp_to_dim:n
1586         {
1587             sqrt
1588             (
1589                 ( \dim_use:N \g_@@_x_final_dim
1590                   - \dim_use:N \g_@@_x_initial_dim
1591                 ) ^ 2
1592             +
1593                 ( \dim_use:N \g_@@_y_final_dim
1594                   - \dim_use:N \g_@@_y_initial_dim
1595                 ) ^ 2
1596             )
1597         }
1598     }

```

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

```

1599     \dim_compare:nNnF \l_@@_l_dim = \c_zero_dim

```

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

```

1600     {
1601         \bool_if:NTF \l_@@_initial_open_bool
1602         {
1603             \bool_if:NTF \l_@@_final_open_bool
1604             {
1605                 \int_set:Nn \l_tmpa_int
1606                 { \dim_ratio:nn \l_@@_l_dim \l_@@_inter_dots_dim }
1607             }
1608             {
1609                 \int_set:Nn \l_tmpa_int
1610                 { \dim_ratio:nn { \l_@@_l_dim - 0.3 em } \l_@@_inter_dots_dim }
1611             }

```

```

1612     }
1613     {
1614         \bool_if:NTF \l_@@_final_open_bool
1615         {
1616             \int_set:Nn \l_tmpa_int
1617             { \dim_ratio:nn { \l_@@_l_dim - 0.3 em } \l_@@_inter_dots_dim }
1618         }
1619         {
1620             \int_set:Nn \l_tmpa_int
1621             { \dim_ratio:nn { \l_@@_l_dim - 0.6 em } \l_@@_inter_dots_dim }
1622         }
1623     }

```

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

```

1624     \dim_set:Nn \l_tmpa_dim
1625     {
1626         ( \g_@@_x_final_dim - \g_@@_x_initial_dim ) *
1627         \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
1628     }
1629     \dim_set:Nn \l_tmpb_dim
1630     {
1631         ( \g_@@_y_final_dim - \g_@@_y_initial_dim ) *
1632         \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
1633     }

```

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`.

```

1634     \int_set:Nn \l_tmpb_int
1635     {
1636         \bool_if:NTF \l_@@_initial_open_bool
1637         { \bool_if:NTF \l_@@_final_open_bool 1 0 }
1638         { \bool_if:NTF \l_@@_final_open_bool 2 1 }
1639     }

```

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.

```

1640     \dim_gadd:Nn \g_@@_x_initial_dim
1641     {
1642         ( \g_@@_x_final_dim - \g_@@_x_initial_dim ) *
1643         \dim_ratio:nn
1644         { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
1645         { \l_@@_l_dim * 2 }
1646         * \l_tmpb_int
1647     }

```

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

```

1648     \dim_gadd:Nn \g_@@_y_initial_dim
1649     {
1650         ( \g_@@_y_final_dim - \g_@@_y_initial_dim ) *
1651         \dim_ratio:nn
1652         { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
1653         { \l_@@_l_dim * 2 } *
1654         \l_tmpb_int
1655     }
1656     \begin { tikzpicture } [ overlay ]
1657         \int_step_inline:nnn 0 \l_tmpa_int
1658         {
1659             \pgfpathcircle
1660             { \pgfpoint { \g_@@_x_initial_dim } { \g_@@_y_initial_dim } }
1661             { \l_@@_radius_dim }

```

```

1662         \pgfusepath { fill }
1663         \dim_gadd:Nn \g_@@_x_initial_dim \l_tmpa_dim
1664         \dim_gadd:Nn \g_@@_y_initial_dim \l_tmpb_dim
1665     }
1666     \end { tikzpicture }
1667 }
1668 }

```

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).

```

1669 \cs_set_eq:NN \@@_ldots \ldots
1670 \cs_set_eq:NN \@@_cdots \cdots
1671 \cs_set_eq:NN \@@_vdots \vdots
1672 \cs_set_eq:NN \@@_ddots \ddots
1673 \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).

```

1674 \cs_new_protected:Nn \@@_add_to_empty_cells:
1675 {
1676     \cs_gset:cpx
1677     { @@ _ empty _ \int_use:N \c@iRow - \int_use:N \c@jCol }
1678     { \int_use:N \g_@@_env_int }
1679 }

```

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.

```

1680 \NewDocumentCommand \@@_Ldots { s }
1681 {
1682     \bool_if:nF { #1 } { \@@_instruction_of_type:n { Ldots } }
1683     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_ldots }
1684     \@@_add_to_empty_cells:
1685 }

```

```

1686 \NewDocumentCommand \@@_Cdots { s }
1687 {
1688     \bool_if:nF { #1 } { \@@_instruction_of_type:n { Cdots } }
1689     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_cdots }
1690     \@@_add_to_empty_cells:
1691 }

```

```

1692 \NewDocumentCommand \@@_Vdots { s }
1693 {
1694     \bool_if:nF { #1 } { \@@_instruction_of_type:n { Vdots } }
1695     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_vdots }
1696     \@@_add_to_empty_cells:
1697 }

```

```

1698 \NewDocumentCommand \@@_Ddots { s }
1699 {
1700     \bool_if:nF { #1 } { \@@_instruction_of_type:n { Ddots } }
1701     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_ddots }
1702     \@@_add_to_empty_cells:
1703 }

```

```

1704 \NewDocumentCommand \@@_Iddots { s }
1705 {
1706   \bool_if:nF { #1 } { \@@_instruction_of_type:n { Iddots } }
1707   \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_iddots }
1708   \@@_add_to_empty_cells:
1709 }

```

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

```

1710 \cs_new_protected:Nn \@@_Hspace:
1711 {
1712   \@@_add_to_empty_cells:
1713   \hspace
1714 }

```

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

```

1715 \cs_set_eq:NN \@@_old_multicolumn \multicolumn
1716 \cs_new:Npn \@@_multicolumn:nnn #1 #2 #3
1717 {
1718   \@@_old_multicolumn { #1 } { #2 } { #3 }
1719   \int_compare:nNnT #1 > 1
1720   {
1721     \seq_gput_left:Nx \g_@@_multicolumn_cells_seq
1722     { \int_eval:n \c@iRow - \int_use:N \c@jCol }
1723     \seq_gput_left:Nn \g_@@_multicolumn_sizes_seq { #1 }
1724   }
1725   \int_gadd:Nn \c@jCol { #1 - 1 }
1726 }

```

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`.

```

1727 \cs_new:Npn \@@_Hdotsfor:
1728 {
1729   \multicolumn { 1 } { C } { }
1730   \@@_Hdotsfor_i
1731 }

```

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:`).

```

1732 \bool_if:NTF \c_@@_draft_bool
1733 {
1734   \NewDocumentCommand \@@_Hdotsfor_i { 0 { } m }
1735   { \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { C } { } } }
1736 }
1737 {
1738   \NewDocumentCommand \@@_Hdotsfor_i { 0 { } m }
1739   {
1740     \tl_gput_right:Nx \g_@@_Hdotsfor_lines_tl
1741     {
1742       \@@_draw_Hdotsfor:nnn
1743       { \int_use:N \c@iRow }
1744       { \int_use:N \c@jCol }
1745       { #2 }
1746     }
1747     \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { C } { } }
1748   }
1749 }

```



```

1750 \cs_new_protected:Nn \@@_draw_Hdotsfor:nnn
1751 {
1752   \bool_set_false:N \l_@@_initial_open_bool
1753   \bool_set_false:N \l_@@_final_open_bool

```

For the row, it's easy.

```

1754   \int_set:Nn \l_@@_initial_i_int { #1 }
1755   \int_set:Nn \l_@@_final_i_int { #1 }

```

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

```

1756   \int_compare:nNnTF #2 = 1
1757   {
1758     \int_set:Nn \l_@@_initial_j_int 1
1759     \bool_set_true:N \l_@@_initial_open_bool
1760   }
1761   {
1762     \int_set:Nn \l_tmpa_int { #2 - 1 }
1763     \@@_if_not_empty_cell:nnTF \l_@@_initial_i_int \l_tmpa_int
1764     { \int_set:Nn \l_@@_initial_j_int { #2 - 1 } }
1765     {
1766       \int_set:Nn \l_@@_initial_j_int {#2}
1767       \bool_set_true:N \l_@@_initial_open_bool
1768     }
1769   }
1770   \int_compare:nNnTF { #2 + #3 - 1 } = \c@jCol
1771   {
1772     \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
1773     \bool_set_true:N \l_@@_final_open_bool
1774   }
1775   {
1776     \int_set:Nn \l_tmpa_int { #2 + #3 }
1777     \@@_if_not_empty_cell:nnTF \l_@@_final_i_int \l_tmpa_int
1778     { \int_set:Nn \l_@@_final_j_int { #2 + #3 } }
1779     {
1780       \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
1781       \bool_set_true:N \l_@@_final_open_bool
1782     }
1783   }
1784   \bool_if:nT { \l_@@_initial_open_bool || \l_@@_final_open_bool }
1785   \@@_create_extra_nodes:
1786   \@@_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).

```

1787   \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
1788   { \cs_set:cpn { @@ _ dotted _ #1 - ##1 } { } }
1789 }

```

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).³¹

```

1790 \cs_new:Npn \@@_double_int_eval:n #1-#2 \q_stop
1791 { \int_eval:n { #1 } - \int_eval:n { #2 } }

```

³¹Indeed, we want that the user may use the command `\line` in `code-after` with LaTeX counters in the arguments — with the command `\value`.

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).

```

1792 \cs_new_protected:Npn \@@_line:nn #1 #2
1793 {
1794   \use:x
1795   {
1796     \@@_line_i:nn
1797     { \@@_double_int_eval:n #1 \q_stop }
1798     { \@@_double_int_eval:n #2 \q_stop }
1799   }
1800 }
1801 \cs_new_protected:Nn \@@_line_i:nn
1802 {
1803   \bool_if:NF \c_@@_draft_bool
1804   {
1805     \dim_zero_new:N \g_@@_x_initial_dim
1806     \dim_zero_new:N \g_@@_y_initial_dim
1807     \dim_zero_new:N \g_@@_x_final_dim
1808     \dim_zero_new:N \g_@@_y_final_dim
1809     \bool_set_false:N \l_@@_initial_open_bool
1810     \bool_set_false:N \l_@@_final_open_bool
1811     \bool_if:nTF
1812     {
1813       \cs_if_exist_p:c { pgf@sh@ns@nm - \int_use:N \g_@@_env_int - #1 }
1814       &&
1815       \cs_if_exist_p:c { pgf@sh@ns@nm - \int_use:N \g_@@_env_int - #2 }
1816     }
1817     {
1818       \begin { tikzpicture }
1819         \path~(#1)~---~(#2)~node[at~start]~(i)~{ }~node[at~end]~(f)~{ } ;
1820         \tikz@parse@node \pgfutil@firstofone ( i )
1821         \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1822         \dim_gset:Nn \g_@@_y_initial_dim \pgf@y
1823         \tikz@parse@node \pgfutil@firstofone ( f )
1824         \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1825         \dim_gset:Nn \g_@@_y_final_dim \pgf@y
1826       \end { tikzpicture }
1827       \@@_draw_tikz_line:
1828     }
1829     {
1830       \@@_error:nnn { unknown~cell~for~line~in~code~after }
1831       { #1 } { #2 }
1832     }
1833   }
1834 }

```

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).

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 position³² as the line created by `\hline` (or `\hdashline` of `arydshln`). That's why we use a `\noalign` to insert a box with a `\dotfill`.

³²In 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`.

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.

```

1835 \bool_if:NTF \c_@@_draft_bool
1836 { \cs_set_eq:NN \@@dotfill: \prg_do_nothing: }
1837 {
1838   \cs_set:Npn \@@dotfill:
1839     {

```

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.

```

1840   \bool_if:NT \l_@@_small_bool
1841   { \dim_set:Nn \l_@@_inter_dots_dim { 0.25 em } }
1842   \cleaders
1843   \hbox_to_wd:nn
1844   { \l_@@_inter_dots_dim }
1845   {
1846     \c_math_toggle_token
1847     \bool_if:NT \l_@@_small_bool \scriptstyle
1848     \hss . \hss
1849     \c_math_toggle_token
1850   }
1851   \hfill
1852 }
1853 }

```

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

```

1854 \cs_new:Npn \@@hdottedline:
1855 {
1856   \noalign
1857   {
1858     \bool_gset_true:N \g_@@_extra_nodes_bool
1859     \cs_if_exist:cTF { @@_width_ \int_use:N \g_@@_env_int }
1860     { \dim_set_eq:Nc \l_tmpa_dim { @@_width_ \int_use:N \g_@@_env_int } }
1861     { \dim_set:Nn \l_tmpa_dim { 5 mm } }
1862     \hbox_overlap_right:n
1863     {
1864       \bool_if:nT
1865       {
1866         \l_@@_NiceArray_bool
1867         &&
1868         ! \l_@@_exterior_arraycolsep_bool
1869         &&
1870         \int_compare_p:nNn \l_@@_first_col_int > \c_zero_int
1871       }
1872       { \skip_horizontal:n { - \arraycolsep } }
1873       \hbox_to_wd:nn
1874       {
1875         \l_tmpa_dim + 2 \arraycolsep
1876         - \l_@@_left_margin_dim - \l_@@_right_margin_dim
1877       }
1878       \@@dotfill:
1879     }
1880   }
1881 }

```

```

1882 \cs_new_protected:Nn \@@vdottedline:n
1883 {

```

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

```

1884   \int_compare:nNnTF #1 = \c_zero_int

```

```

1885 { \@@_error:n { Use-of-~in-first-position } }
1886 {
1887   \bool_if:NF \c_@@_draft_bool
1888   {
1889     \dim_zero_new:N \g_@@_x_initial_dim
1890     \dim_zero_new:N \g_@@_y_initial_dim
1891     \dim_zero_new:N \g_@@_x_final_dim
1892     \dim_zero_new:N \g_@@_y_final_dim
1893     \bool_set_true:N \l_@@_initial_open_bool
1894     \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.

```

1895   \cs_if_exist:cTF
1896   { pgf@sh@ns@nm -\int_use:N \g_@@_env_int - col - #1 }
1897   {
1898     \begin { tikzpicture } [ remember-picture ]
1899       \tikz@parse@node\pgfutil@firstofone
1900       ( col - #1 )
1901       \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1902       \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1903       \dim_gset:Nn \g_@@_y_final_dim \pgf@y
1904     \end { tikzpicture }
1905     \dim_gset:Nn \g_@@_y_initial_dim { - \c_max_dim }
1906     \int_step_inline:nn \c@jCol
1907     {
1908       \begin { tikzpicture } [ remember-picture ]
1909         \tikz@parse@node\pgfutil@firstofone
1910         ( 1 - ##1 . north-east )
1911         \dim_gset:Nn \g_@@_y_initial_dim
1912         { \dim_max:nn \g_@@_y_initial_dim \pgf@y }
1913       \end { tikzpicture }
1914     }
1915   }

```

If not, we use the “large node”.

```

1916   {
1917     \begin { tikzpicture } [ remember-picture ]
1918       \tikz@parse@node\pgfutil@firstofone
1919       ( 1 - #1 - large .north-east )
1920       \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1921       \dim_gset:Nn \g_@@_y_initial_dim \pgf@y
1922       \tikz@parse@node\pgfutil@firstofone
1923       ( \int_use:N \c@iRow - #1 - large .south-east )
1924       \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1925       \dim_gset:Nn \g_@@_y_final_dim \pgf@y
1926     \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).

```

1927   \cs_if_exist:cTF
1928   { pgf@sh@ns@nm -\int_use:N \g_@@_env_int - 1 - #1 - w }
1929   {
1930     \begin { tikzpicture } [ remember-picture ]
1931       \tikz@parse@node\pgfutil@firstofone
1932       ( 1 - #1 - w .north-east )
1933       \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1934       \tikz@parse@node\pgfutil@firstofone
1935       ( \int_use:N \c@iRow - #1 - w .south-east )
1936       \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1937     \end { tikzpicture }
1938     \dim_gadd:Nn \g_@@_x_initial_dim \arraycolsep
1939     \dim_gadd:Nn \g_@@_x_final_dim \arraycolsep
1940   }
1941 }
1942 \@@_draw_tikz_line:

```

```

1943     }
1944   }
1945 }

```

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:nNnF \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).

```

1946 \cs_new_protected:Npn \@@_vline:
1947 {
1948   \int_compare:nNnTF \l_@@_first_col_int = \c_zero_int
1949   {
1950     \int_compare:nNnTF \c@jCol = \c_zero_int
1951     {
1952       \int_compare:nNnTF \l_@@_first_row_int = \c_zero_int
1953       {
1954         \int_compare:nNnF \c@iRow = \c_zero_int
1955         {
1956           \int_compare:nNnF \c@iRow = \l_@@_last_row_int
1957           \@@_vline_i:
1958         }
1959       }
1960     }
1961     \int_compare:nNnF \c@iRow = \c_zero_int
1962     {
1963       \int_compare:nNnF \c@iRow = \l_@@_last_row_int
1964       \@@_vline_i:
1965     }
1966   }
1967 }
1968 {
1969   \int_compare:nNnF \c@iRow = \c_zero_int
1970   {
1971     \int_compare:nNnF \c@iRow = \l_@@_last_row_int
1972     \@@_vline_i:
1973   }
1974 }
1975 }
1976 {
1977   \int_compare:nNnTF \c@jCol = \c_zero_int
1978   {
1979     \int_compare:nNnF \c@iRow = { -1 }
1980     {
1981       \int_compare:nNnF \c@iRow = { \l_@@_last_row_int - 1 }
1982       \@@_vline_i:
1983     }
1984   }
1985   {
1986     \int_compare:nNnF \c@iRow = \c_zero_int

```

```

1987         {
1988             \int_compare:nNnF \c@iRow = \l_@@_last_row_int
1989             \@@_vline_i:
1990         }
1991     }
1992 }
1993 }

```

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

```

1994 \cs_set_eq:NN \@@_vline_i: \vline

```

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.

```

1995 \bool_new:N \l_@@_block_auto_columns_width_bool

```

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

```

1996 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
1997 {
1998     auto-columns-width .code:n =
1999     {
2000         \bool_set_true:N \l_@@_block_auto_columns_width_bool
2001         \dim_gzero_new:N \g_@@_max_cell_width_dim
2002         \bool_set_true:N \l_@@_auto_columns_width_bool
2003     }
2004 }

2005 \NewDocumentEnvironment { NiceMatrixBlock } { ! 0 { } }
2006 {
2007     \int_gincr:N \g_@@_NiceMatrixBlock_int
2008     \dim_zero:N \l_@@_columns_width_dim
2009     \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
2010     \bool_if:NT \l_@@_block_auto_columns_width_bool
2011     {
2012         \cs_if_exist:cT { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
2013         {
2014             \dim_set:Nx \l_@@_columns_width_dim
2015             { \use:c { @@_max_cell_width _ \int_use:N \g_@@_NiceMatrixBlock_int } }
2016         }
2017     }
2018 }

```

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`).

```

2019 {
2020     \bool_if:NT \l_@@_block_auto_columns_width_bool
2021     {
2022         \iow_now:Nn \@mainaux \ExplSyntaxOn
2023         \iow_now:Nx \@mainaux
2024         {
2025             \cs_gset:cpn
2026             { @@ _ max _ cell _ width _ \int_use:N \g_@@_NiceMatrixBlock_int }
2027             { \dim_use:N \g_@@_max_cell_width_dim }
2028         }
2029         \iow_now:Nn \@mainaux \ExplSyntaxOff
2030     }
2031 }

```

17.15 The extra nodes

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

```
2032 \cs_generate_variant:Nn \dim_min:nn { v n }
2033 \cs_generate_variant:Nn \dim_max:nn { v n }
```

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

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

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

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

```
2034 \cs_new_protected:Nn \@@_create_extra_nodes:
2035 {
2036   \begin { tikzpicture } [ remember~picture , overlay ]
2037     \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
2038     {
2039       \dim_zero_new:c { l_@@_row_\@@_i: _min_dim }
2040       \dim_set_eq:cN { l_@@_row_\@@_i: _min_dim } \c_max_dim
2041       \dim_zero_new:c { l_@@_row_\@@_i: _max_dim }
2042       \dim_set:cn { l_@@_row_\@@_i: _max_dim } { - \c_max_dim }
2043     }
2044     \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
2045     {
2046       \dim_zero_new:c { l_@@_column_\@@_j: _min_dim }
2047       \dim_set_eq:cN { l_@@_column_\@@_j: _min_dim } \c_max_dim
2048       \dim_zero_new:c { l_@@_column_\@@_j: _max_dim }
2049       \dim_set:cn { l_@@_column_\@@_j: _max_dim } { - \c_max_dim }
2050     }
2051   }
```

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

```
2051   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
2052   {
2053     \int_step_variable:nnNn
2054     \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.

```
2055     { \cs_if_exist:cT
2056       { 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`.

```
2057     {
2058       \tikz@parse@node \pgfutil@firstofone
2059       ( nm - \int_use:N \g_@@_env_int
2060         - \@@_i: - \@@_j: .south~west )
2061       \dim_set:cn { l_@@_row_\@@_i: _min_dim }
2062       { \dim_min:vn { l_@@_row_\@@_i: _min_dim } \pgf@y }
2063       \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
2064       {
2065         \dim_set:cn { l_@@_column_\@@_j: _min_dim }
2066         { \dim_min:vn { l_@@_column_\@@_j: _min_dim } \pgf@x }
2067       }
2068     }
```

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`.

```

2068      \tikz@parse@node \pgfutil@firstofone
2069      ( nm - \int_use:N \g_@@_env_int - \@@_i: - \@@_j: .north-east )
2070      \dim_set:cn { l_@@_row _ \@@_i: _ max_dim }
2071      { \dim_max:vn { l_@@_row _ \@@_i: _ max_dim } \pgf@y }
2072      \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
2073      {
2074          \dim_set:cn { l_@@_column _ \@@_j: _ max_dim }
2075          { \dim_max:vn { l_@@_column _ \@@_j: _ max_dim } \pgf@x }
2076      }
2077  }
2078 }
2079 }

```

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`).

```

2080      \tikzset { name-suffix = -medium }
2081      \@@_create_nodes:

```

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

```

2082      \int_set:Nn \l_@@_first_row_int 1
2083      \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`.

```

2084      \int_step_variable:nNn { \c@iRow - 1 } \@@_i:
2085      {
2086          \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim }
2087          {
2088              (
2089                  \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } +
2090                  \dim_use:c { l_@@_row _ \int_eval:n { \@@_i: + 1 } _ max _ dim }
2091              )
2092              / 2
2093          }
2094          \dim_set_eq:cc { l_@@_row _ \int_eval:n { \@@_i: + 1 } _ max _ dim }
2095          { l_@@_row _ \@@_i: _ min_dim }
2096      }
2097      \int_step_variable:nNn { \c@jCol - 1 } \@@_j:
2098      {
2099          \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim }
2100          {
2101              (
2102                  \dim_use:c
2103                  { l_@@_column _ \@@_j: _ max _ dim } +
2104                  \dim_use:c
2105                  { l_@@_column _ \int_eval:n { \@@_j: + 1 } _ min _ dim }
2106              )
2107              / 2
2108          }
2109          \dim_set_eq:cc { l_@@_column _ \int_eval:n { \@@_j: + 1 } _ min _ dim }
2110          { l_@@_column _ \@@_j: _ max _ dim }
2111      }
2112      \dim_sub:cn
2113      { l_@@_column _ 1 _ min _ dim }
2114      \l_@@_left_margin_dim
2115      \dim_add:cn
2116      { l_@@_column _ \int_use:N \c@jCol _ max _ dim }
2117      \l_@@_right_margin_dim

```


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

```

2118     \tikzset { name~suffix = -large }
2119     \@@_create_nodes:
2120     \end{tikzpicture}

```

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

```

2121     \cs_set:Npn \@@_create_extra_nodes: { }

```

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

```

2122     \begin { tikzpicture } [ remember~picture , overlay ]
2123     \tikz@parse@node \pgfutil@firstofone
2124     ( nm - \int_use:N \g_@@_env_int - 1 - 1 - large .north~west )
2125     \dim_gset:Nn \g_tmpa_dim \pgf@x
2126     \tikz@parse@node \pgfutil@firstofone
2127     ( nm - \int_use:N \g_@@_env_int - 1 -
2128       \int_use:N \c@jCol - large .north~east )
2129     \dim_gset:Nn \g_tmpb_dim \pgf@x
2130     \end { tikzpicture }
2131     \iow_now:Nn \@mainaux \ExplSyntaxOn
2132     \iow_now:Nx \@mainaux
2133     {
2134         \cs_gset:cpn { @@_width_ \int_use:N \g_@@_env_int }
2135         { \dim_eval:n { \g_tmpb_dim - \g_tmpa_dim } }
2136     }
2137     \iow_now:Nn \@mainaux \ExplSyntaxOff
2138 }

```

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.

```

2139 \cs_new_protected:Nn \@@_create_nodes:
2140 {
2141     \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
2142     {
2143         \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:

```

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

```

2144     {
2145         \coordinate ( @@~south~west )
2146         at ( \dim_use:c { l_@@_column_ \@@_j: _min_dim } ,
2147             \dim_use:c { l_@@_row_ \@@_i: _min_dim } ) ;
2148         \coordinate ( @@~north~east )
2149         at ( \dim_use:c { l_@@_column_ \@@_j: _max_dim } ,
2150             \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`.

```

2151     \node
2152     [
2153         node~contents = { } ,
2154         fit = ( @@~south~west ) ( @@~north~east ) ,
2155         inner~sep = \c_zero_dim ,
2156         name = nm - \int_use:N \g_@@_env_int - \@@_i: - \@@_j: ,
2157         alias =
2158         \str_if_empty:NF \l_@@_name_str

```

```

2159         { \l_@@_name_str - \@@_i: - \@@_j: }
2160     ]
2161     ;
2162 }
2163 }

```

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 .

```

2164 \seq_mapthread_function:NNN
2165 \g_@@_multicolumn_cells_seq
2166 \g_@@_multicolumn_sizes_seq
2167 \@@_node_for_multicolumn:nn
2168 }
2169 \cs_new_protected:Npn \@@_extract_coords: #1 - #2 \q_stop
2170 {
2171     \cs_set:Npn \@@_i: { #1 }
2172     \cs_set:Npn \@@_j: { #2 }
2173 }

```

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”).

```

2174 \cs_new_protected:Nn \@@_node_for_multicolumn:nn
2175 {
2176     \@@_extract_coords: #1 \q_stop
2177     \coordinate ( @@~south~west ) at
2178     (
2179         \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } ,
2180         \dim_use:c { l_@@_row _ \@@_i: _ min _ dim }
2181     ) ;
2182     \coordinate ( @@~north~east ) at
2183     (
2184         \dim_use:c { l_@@_column _ \int_eval:n { \@@_j: + #2 - 1 } _ max _ dim } ,
2185         \dim_use:c { l_@@_row _ \@@_i: _ max _ dim }
2186     ) ;
2187     \node
2188     [
2189         node~contents = { } ,
2190         fit = ( @@~south~west ) ( @@~north~east ) ,
2191         inner~sep = \c_zero_dim ,
2192         name = nm - \int_use:N \g_@@_env_int - \@@_i: - \@@_j: ,
2193         alias =
2194         \str_if_empty:NF \l_@@_name_str
2195         { \l_@@_name_str - \@@_i: - \@@_j: }
2196     ]
2197     ;
2198 }

```

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 optional argument between `<` and `>` (for TeX instructions put before the math mode of the label)

```

2199 \NewDocumentCommand \@@_Block: { m D < > { } m }
2200 { \@@_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.

```

2201 \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).

```
2202 \cs_new:Npn \@@_Block_ii:nnnn #1 #2 #3 #4
2203 {
```

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.

```
2204   \int_compare:nNt { #1 } = 1
2205   {
2206     \begin { tikzpicture } [ remember-picture , baseline ]
2207       \coordinate
2208         ( nm - \int_use:N \g_@@_env_int
2209           - Block
2210           - \int_use:N \c@iRow
2211           - \int_use:N \c@jCol ) ;
2212     \end { tikzpicture }
2213   }
2214   \bool_gset_true:N \g_@@_extra_nodes_bool
```

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

```
2215   \tl_gput_left:Nx \g_@@_code_after_tl
2216   {
2217     \@@_Block_iii:nnnnn
2218     { \int_use:N \c@iRow }
2219     { \int_use:N \c@jCol }
2220     { \int_eval:n { \c@iRow + #1 - 1 } }
2221     { \int_eval:n { \c@jCol + #2 - 1 } }
2222     \exp_not:n { { #3 $ #4 $ } }
2223   }
2224 }
```

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

```
2225 \cs_new_protected:Npn \@@_Block_iii:nnnnn #1 #2 #3 #4 #5
2226 {
2227   \bool_if:nTF
2228   {
2229     \int_compare_p:nNn { #3 } > \c@iRow
2230     || \int_compare_p:nNn { #4 } > \c@jCol
2231   }
2232   { \msg_error:nnnn { nicematrix } { Block-too-large } { #1 } { #2 } }
2233   {
```

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.

```
2234   \int_compare:nNtTF { #1 } = { #3 }
2235   {
2236     \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.

```
2237     \tikz@parse@node \pgfutil@firstofone (Block-#1-#2)
2238     \dim_set:Nn \l_tmpa_dim \pgf@y
2239     \node
2240     [
2241       fit = ( #1 - #2 - medium . north-west )
2242             ( #3 - #4 - medium . south-east ) ,
2243       inner-sep = 0 pt ,
2244     ]
2245     (#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`.

```

2246         \tikz@parse@node \pgfutil@firstofone (#1-#2)
2247         \path (\pgf@x,\l_tmpa_dim) node [ anchor = base ] { #5 } ;
2248     \end { tikzpicture }
2249 }

```

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.

```

2250     {
2251     \begin { tikzpicture }
2252     \node
2253     [
2254         fit = ( #1 - #2 - medium . north-west )
2255             ( #3 - #4 - medium . south-east ) ,
2256         inner~sep = 0 pt ,
2257     ]

```

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

```

2258         (#1-#2) { } ;
2259     \node at (#1-#2.center) { #5 } ;
2260     \end { tikzpicture }
2261 }
2262 }
2263 }

```

17.17 How to draw the dotted lines transparently

```

2264 \cs_set_protected:Npn \@@_renew_matrix:
2265 {
2266     \RenewDocumentEnvironment { pmatrix } { } {
2267         { \pNiceMatrix }
2268         { \endpNiceMatrix }
2269     \RenewDocumentEnvironment { vmatrix } { } {
2270         { \vNiceMatrix }
2271         { \endvNiceMatrix }
2272     \RenewDocumentEnvironment { Vmatrix } { } {
2273         { \VNiceMatrix }
2274         { \endVNiceMatrix }
2275     \RenewDocumentEnvironment { bmatrix } { } {
2276         { \bNiceMatrix }
2277         { \endbNiceMatrix }
2278     \RenewDocumentEnvironment { Bmatrix } { } {
2279         { \BNiceMatrix }
2280         { \endBNiceMatrix }
2281     }

```

17.18 Automatic arrays

```

2282 \cs_new_protected:Npn \@@_set_size:n #1-#2 \q_stop
2283 {
2284     \int_set:Nn \l_@@_nb_rows_int { #1 }
2285     \int_set:Nn \l_@@_nb_cols_int { #2 }
2286 }
2287 \NewDocumentCommand \AutoNiceMatrixWithDelims { m m O { } m O { } m ! O { } }
2288 {
2289     \int_zero_new:N \l_@@_nb_rows_int
2290     \int_zero_new:N \l_@@_nb_cols_int
2291     \@@_set_size:n #4 \q_stop
2292     \begin { NiceArrayWithDelims } { #1 } { #2 }
2293         { * { \l_@@_nb_cols_int } { C } } [ #3 , #5 , #7 ]
2294     \int_compare:nNnT \l_@@_first_row_int = \c_zero_int
2295     {

```

```

2296     \int_compare:nNnT \l_@@_first_col_int = \c_zero_int { & }
2297     \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
2298     \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
2299   }
2300   \prg_replicate:nn \l_@@_nb_rows_int
2301   {
2302     \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`).

```

2303     \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { { } #6 & } #6
2304     \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
2305   }
2306   \int_compare:nNnT \l_@@_last_row_int > { -2 }
2307   {
2308     \int_compare:nNnT \l_@@_first_col_int = \c_zero_int { & }
2309     \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
2310     \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
2311   }
2312   \end { NiceArrayWithDelims }
2313 }
2314 \cs_set_protected:Npn \@@_define_com:nnn #1 #2 #3
2315 {
2316   \cs_set_protected:cpn { #1 AutoNiceMatrix }
2317   {
2318     \str_gset:Nx \g_@@_type_env_str
2319     { command ~ \c_backslash_str #1 AutoNiceMatrix }
2320     \AutoNiceMatrixWithDelims { #2 } { #3 }
2321   }
2322 }
2323 \@@_define_com:nnn p ( )
2324 \@@_define_com:nnn b [ ]
2325 \@@_define_com:nnn v | |
2326 \@@_define_com:nnn V \ | \ |
2327 \@@_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.

```

2328 \keys_define:nn { NiceMatrix / Package }
2329 {
2330   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
2331   renew-dots .value_forbidden:n = true ,
2332   renew-matrix .code:n = \@@_renew_matrix: ,
2333   renew-matrix .value_forbidden:n = true ,
2334   transparent .meta:n = { renew-dots , renew-matrix } ,
2335   transparent .value_forbidden:n = true ,
2336   obsolete-environments .code:n =
2337     \@@_msg_redirect_name:nn { Obsolete-environment } { none }
2338 }
2339 \ProcessKeysOptions { NiceMatrix / Package }

```

17.20 Error messages of the package

```

2340 \@@_msg_new:nn { unknown~cell~for~line~in~code~after }
2341 {
2342   Your~command~\token_to_str:N\line\{#1\}\{#2\}~in~the~'code~after'~

```

```

2343   can't-be-executed-because-a-Tikz-node-doesn't-exist.\\
2344   If-you-go-on-this-command-will-be-ignored.
2345 }

2346 \@@_msg_new:nn { last-col-non-empty-for-NiceArray }
2347 {
2348   In-the-\g_@@_type_env_str,~you-must-use-the-option~
2349   'last-col'~without-value.\\
2350   However,~you-can-go-on-for-this-time~
2351   (the-value~'\l_keys_value_tl'~will-be-ignored).
2352 }

2353 \@@_msg_new:nn { last-col-empty-for-NiceMatrix }
2354 {
2355   In-the-\g_@@_type_env_str, you-can't-use-the-option~
2356   'last-col'~without-value.~You-must-give-the-number-of-that-last-column.\\
2357   If-you-go-on-this-option-will-be-ignored.
2358 }

2359 \@@_msg_new:nn { Block-too-large }
2360 {
2361   You-try-to-draw-a-block-in-the-cell~#1-#2-of-your-matrix-but-the-matrix-is~
2362   too-small-for-that-block.\\
2363   If-you-go-on,~this-command-will-be-ignored.
2364 }

2365 \@@_msg_new:nn { Impossible-line }
2366 {
2367   A-dotted-line-can't-be-drawn-because-you-have-not-put~
2368   all-the-ampersands-required-on-the-row~#1.\\
2369   If-you-go-on,~this-dotted-line-will-be-ignored.
2370 }

2371 \@@_msg_new:nn { Wrong-last-row }
2372 {
2373   You-have-used~'last-row=\int_use:N \l_@@_last_row_int'~but-your~
2374   \g_@@_type_env_str\ seems-to-have~\int_use:N \c@iRow \ rows.~
2375   If-you-go-on,~the-value-of~\int_use:N \c@iRow \ will-be-used-for~
2376   last-row.~You-can-avoid-this-problem-by-using~'last-row'~
2377   without-value~(more-compilations-might-be-necessary).
2378 }

2379 \@@_msg_new:nn { Yet-in-env }
2380 {
2381   Environments~\{NiceArray\}~(or~\{NiceMatrix\},~etc.)~can't-be-nested.\\
2382   This-error-is-fatal.
2383 }

2384 \@@_msg_new:nn { Outside-math-mode }
2385 {
2386   The~\g_@@_type_env_str\ can-be-used-only-in-math-mode~
2387   (and-not-in~\token_to_str:N \vcenter).\\
2388   This-error-is-fatal.
2389 }

2390 \@@_msg_new:nn { Option-Transparent-suppressed }
2391 {
2392   The-option~'Transparent'~has-been-renamed~'transparent'.\\
2393   However,~you-can-go-on-for-this-time.
2394 }

2395 \@@_msg_new:nn { Option-RenewMatrix-suppressed }
2396 {
2397   The-option~'RenewMatrix'~has-been-renamed~'renew-matrix'.\\
2398   However,~you-can-go-on-for-this-time.
2399 }

2400 \@@_msg_new:nn { Bad-value-for-letter-for-dotted-lines }
2401 {

```

```

2402 The~value~of~key~'\tl_use:N\l_keys_key_tl'~must~be~of~length~1.\\
2403 If~you~go~on,~it~will~be~ignored.
2404 }
2405 \@@_msg_new:nnn { Unknown~key~for~NiceMatrixOptions }
2406 {
2407 The~key~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~command~
2408 \token_to_str:N \NiceMatrixOptions. \\
2409 If~you~go~on,~it~will~be~ignored. \\
2410 For~a~list~of~the~available~keys,~type~H~<return>.
2411 }
2412 {
2413 The~available~options~are~(in~alphabetic~order):~
2414 allow~duplicate~names,~
2415 code~for~first~col,~
2416 code~for~first~row,~
2417 code~for~last~col,~
2418 code~for~last~row,~
2419 exterior~arraycolsep,~
2420 hlines,~
2421 left~margin,~
2422 letter~for~dotted~lines,~
2423 nullify~dots,~
2424 parallelize~diags,~
2425 renew~dots,~
2426 renew~matrix,~
2427 right~margin,~
2428 small~
2429 and~transparent
2430 }
2431 \@@_msg_new:nnn { Unknown~option~for~NiceArray }
2432 {
2433 The~option~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~environment~
2434 \{NiceArray\}. \\
2435 If~you~go~on,~it~will~be~ignored. \\
2436 For~a~list~of~the~available~options,~type~H~<return>.
2437 }
2438 {
2439 The~available~options~are~(in~alphabetic~order):~
2440 b,~
2441 c,~
2442 code~after,~
2443 code~for~first~col,~
2444 code~for~first~row,~
2445 code~for~last~col,~
2446 code~for~last~row,~
2447 columns~width,~
2448 create~extra~nodes,~
2449 extra~left~margin,~
2450 extra~right~margin,~
2451 first~col,~
2452 first~row,~
2453 hlines,~
2454 last~col,~
2455 last~row,~
2456 left~margin,~
2457 name,~
2458 nullify~dots,~
2459 parallelize~diags,~
2460 renew~dots,~
2461 right~margin,~
2462 small~
2463 and~t.
2464 }

```

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).

```

2465 \@@_msg_new:nnn { Unknown-option-for-NiceMatrix }
2466 {
2467   The-option~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~
2468   \g_@@_type_env_str. \\
2469   If-you-go-on,~it~will~be~ignored. \\
2470   For-a-list-of-the-available-options,~type~H~<return>.
2471 }
2472 {
2473   The~available~options~are~(in~alphabetic~order):~
2474   code-after,~
2475   code-for-first-col,~
2476   code-for-first-row,~
2477   code-for-last-col,~
2478   code-for-last-row,~
2479   columns-width,~
2480   create-extra-nodes,~
2481   extra-left-margin,~
2482   extra-right-margin,~
2483   first-col,~
2484   first-row,~
2485   hlines,~
2486   last-col,~
2487   last-row,~
2488   left-margin,~
2489   name,~
2490   nullify-dots,~
2491   parallelize-diags,~
2492   renew-dots,~
2493   right-margin~
2494   and~small.
2495 }

```

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

```

2496 \@@_msg_new:nnn { Duplicate-name }
2497 {
2498   The-name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
2499   the~same~environment~name~twice.~You~can~go~on,~but,~
2500   maybe,~you~will~have~incorrect~results~especially~
2501   if~you~use~'columns-width=auto'.~If~you~use~nicematrix~inside~some~
2502   environments~of~amsmath,~this~error~may~be~an~artefact.~In~this~case,~
2503   use~the~option~'allow-duplicate-names'.\\
2504   For-a-list-of-the-names~already~used,~type~H~<return>. \\
2505 }
2506 {
2507   The~names~already~defined~in~this~document~are:~
2508   \seq_use:Nnnn \g_@@_names_seq { ,~ } { ,~ } { ~and~ }.
2509 }
2510 \@@_msg_new:nn { Option-auto-for-columns-width }
2511 {
2512   You~can't~give~the~value~'auto'~to~the~option~'columns-width'~here.~
2513   If~you~go~on,~the~option~will~be~ignored.
2514 }
2515 \@@_msg_new:nn { Zero-row }
2516 {
2517   There~is~a~problem.~Maybe~your~\g_@@_type_env_str\ is~empty.~
2518   Maybe~you~have~used~l,~c~and~r~instead~of~L,~C~and~R~in~the~preamble~
2519   of~your~environment. \\
2520   If~you~go~on,~the~result~may~be~incorrect.
2521 }
2522 \@@_msg_new:nn { Use-of-~:~in-first-position }

```



```

2523 {
2524     You~can't~use~the~column~specifier~'\l_@@_letter_for_dotted_lines_str'~in~the~
2525     first~position~of~the~preamble~of~the~\g_@@_type_env_str. \\\
2526     If~you~go~on,~this~dotted~line~will~be~ignored.
2527 }

```

17.21 Obsolete environments

```

2528 \@@_msg_new:nn { Obsolete~environment }
2529 {
2530     The~environment~\{\@currentenv\}~is~obsolete.~We~should~use~#1~instead.~
2531     However,~you~can~go~on~for~this~time.~
2532     If~you~don't~want~to~see~this~error~again,~you~should~load~'nicematrix'~
2533     with~the~option~'obsolete~environments'.
2534 }
2535 \NewDocumentEnvironment { pNiceArrayC } { }
2536 {
2537     \@@_error:nn { Obsolete~environment }
2538     { the~option~'last-col' }
2539     \int_set:Nn \l_@@_last_col_int \c_zero_dim
2540     \pNiceArray
2541 }
2542 { \endpNiceArray }
2543 \NewDocumentEnvironment { bNiceArrayC } { }
2544 {
2545     \@@_error:nn { Obsolete~environment }
2546     { the~option~'last-col' }
2547     \int_set:Nn \l_@@_last_col_int \c_zero_dim
2548     \bNiceArray
2549 }
2550 { \endbNiceArray }
2551 \NewDocumentEnvironment { BNiceArrayC } { }
2552 {
2553     \@@_error:nn { Obsolete~environment }
2554     { the~option~'last-col' }
2555     \int_set:Nn \l_@@_last_col_int \c_zero_dim
2556     \BNiceArray
2557 }
2558 { \endBNiceArray }
2559 \NewDocumentEnvironment { vNiceArrayC } { }
2560 {
2561     \@@_error:nn { Obsolete~environment }
2562     { the~option~'last-col' }
2563     \int_set:Nn \l_@@_last_col_int \c_zero_dim
2564     \vNiceArray
2565 }
2566 { \endvNiceArray }
2567 \NewDocumentEnvironment { VNiceArrayC } { }
2568 {
2569     \@@_error:nn { Obsolete~environment }
2570     { the~option~'last-col' }
2571     \int_set:Nn \l_@@_last_col_int \c_zero_dim
2572     \VNiceArray
2573 }
2574 { \endVNiceArray }
2575 \NewDocumentEnvironment { pNiceArrayRC } { }
2576 {
2577     \@@_error:nn { Obsolete~environment }
2578     { the~options~'last-col'~and~'first-row' }
2579     \int_set:Nn \l_@@_last_col_int \c_zero_dim
2580     \int_set:Nn \l_@@_first_row_int \c_zero_int
2581     \pNiceArray

```

```

2582 }
2583 { \endpNiceArray }
2584 \NewDocumentEnvironment { bNiceArrayRC } { }
2585 {
2586   \@@_error:nn { Obsolete-environment }
2587   { the~options~'last-col'~and~'first-row' }
2588   \int_set:Nn \l_@@_last_col_int \c_zero_dim
2589   \int_set:Nn \l_@@_first_row_int \c_zero_int
2590   \bNiceArray
2591 }
2592 { \endbNiceArray }
2593 \NewDocumentEnvironment { BNiceArrayRC } { }
2594 {
2595   \@@_error:nn { Obsolete-environment }
2596   { the~options~'last-col'~and~'first-row' }
2597   \int_set:Nn \l_@@_last_col_int \c_zero_dim
2598   \int_set:Nn \l_@@_first_row_int \c_zero_int
2599   \BNiceArray
2600 }
2601 { \endBNiceArray }
2602 \NewDocumentEnvironment { vNiceArrayRC } { }
2603 {
2604   \@@_error:nn { Obsolete-environment }
2605   { the~options~'last-col'~and~'first-row' }
2606   \bool_set_true:N \l_@@_last_col_bool
2607   \int_set:Nn \l_@@_first_row_int \c_zero_int
2608   \vNiceArray
2609 }
2610 { \endvNiceArray }
2611 \NewDocumentEnvironment { VNiceArrayRC } { }
2612 {
2613   \@@_error:nn { Obsolete-environment }
2614   { the~options~'last-col'~and~'first-row' }
2615   \int_set:Nn \l_@@_last_col_int \c_zero_dim
2616   \int_set:Nn \l_@@_first_row_int \c_zero_int
2617   \VNiceArray
2618 }
2619 { \endVNiceArray }
2620 \NewDocumentEnvironment { NiceArrayCwithDelims } { }
2621 {
2622   \@@_error:nn { Obsolete-environment }
2623   { the~option~'last-col' }
2624   \int_set:Nn \l_@@_last_col_int \c_zero_dim
2625   \NiceArrayWithDelims
2626 }
2627 { \endNiceArrayWithDelims }
2628 \NewDocumentEnvironment { NiceArrayRCwithDelims } { }
2629 {
2630   \@@_error:nn { Obsolete-environment }
2631   { the~options~'last-col'~and~'first-row' }
2632   \int_set:Nn \l_@@_last_col_int \c_zero_dim
2633   \int_set:Nn \l_@@_first_row_int \c_zero_int
2634   \NiceArrayWithDelims
2635 }
2636 { \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 StackExchange³³, Tikz externalization is now deactivated in the environments of the extension `nicematrix`.³⁴

Changes between version 2.1 and 2.1.2

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

³³cf. tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package

³⁴Before this version, there was an error when using `nicematrix` with Tikz externalization. In any case, it's not possible to externalize the Tikz elements constructed by `nicematrix` because they use the options `overlay` and `remember picture`.

Changes between version 2.1.2 and 2.1.3

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

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

Changes between version 2.2 and 2.2.1

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

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

Changes between version 2.2.1 and 2.3

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

Option `hlines`.

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

Changes between version 2.3 and 3.0

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

Composition of exterior rows and columns 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 StackExchange³⁵, 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.

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).

³⁵cf. tex.stackexchange.com/questions/510841/nicematrix-and-tikz-external-optimize

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.

Index

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

Symbols	
<code>\&</code>	2224
<code>\%</code>	2289,
	2295, 2301, 2336, 2342, 2349, 2355, 2361,
	2374, 2380, 2385, 2390, 2395, 2401, 2402,
	2427, 2428, 2461, 2462, 2496, 2497, 2512, 2518
<code>\{</code>	984, 2320, 2335, 2374, 2427, 2523
<code>\}</code>	984, 2320, 2335, 2374, 2427, 2523
<code>\ </code>	1000, 2319
<code>\sqcup</code>	2367, 2368, 2379, 2510
A	
<code>\array</code>	438
<code>\arraycolsep</code>	165, 167, 169, 470,
	619, 620, 679, 682, 690, 694, 719, 726, 758,
	796, 798, 812, 881, 919, 1871, 1874, 1937, 1938
<code>\arrayrulewidth</code>	452, 453
<code>\arraystretch</code>	469
<code>\AtBeginDocument</code>	64, 92
<code>\AutoNiceMatrixWithDelims</code>	2278, 2311
B	
<code>\begin</code>	1009,
	1057, 1300, 1323, 1338, 1655, 1817, 1897,
	1907, 1916, 1929, 2035, 2121, 2243, 2283, 2315
<code>\bgroup</code>	354
<code>\Block</code>	517
<code>\BNiceArray</code>	2549, 2592
<code>\bNiceArray</code>	2541, 2583
<code>\BNiceMatrix</code>	2270
<code>\bNiceMatrix</code>	2267
bool commands:	
<code>\bool_do_until:Nn</code>	1171, 1231
<code>\bool_gset_eq:NN</code>	472
<code>\bool_gset_false:N</code>	562
<code>\bool_gset_true:N</code>	550, 888, 1857, 2212
<code>\bool_if:NTF</code>	36, 100, 292, 414,
	430, 446, 467, 475, 518, 560, 579, 588, 591,
	617, 648, 650, 657, 659, 735, 761, 809, 823,
	894, 1054, 1097, 1115, 1124, 1131, 1195,
	1223, 1255, 1284, 1292, 1318, 1330, 1355,
	1362, 1377, 1392, 1394, 1406, 1410, 1425,
	1440, 1442, 1454, 1458, 1463, 1471, 1476,
	1482, 1486, 1501, 1505, 1510, 1514, 1543,
	1547, 1552, 1556, 1600, 1602, 1613, 1635,
	1636, 1637, 1682, 1688, 1694, 1700, 1706,
	1731, 1802, 1834, 1839, 1846, 1886, 2009, 2019
<code>\bool_if:nTF</code>	545,
	670, 683, 700, 824, 895, 1093, 1291, 1681,
	1687, 1693, 1699, 1705, 1783, 1810, 1863, 2236
<code>\bool_new:N</code>	11, 27, 54, 55, 62, 63, 87,
	90, 91, 132, 133, 135, 136, 137, 139, 140, 1994
<code>\bool_set:Nn</code>	1467, 1480
<code>\bool_set_false:N</code>	1030, 1044,
	1129, 1130, 1170, 1175, 1230, 1235, 1353,
	1404, 1452, 1499, 1541, 1751, 1752, 1808, 1809
<code>\bool_set_true:N</code>	12,
	29, 32, 68, 95, 134, 179, 230, 580, 597, 957,
	1052, 1179, 1185, 1191, 1199, 1203, 1227,
	1239, 1245, 1251, 1259, 1264, 1288, 1758,
	1766, 1772, 1780, 1892, 1893, 1999, 2001, 2599
<code>\l_tmpa_bool</code>	1030, 1044, 1052, 1054, 1467, 1486
<code>\l_tmpb_bool</code>	1480, 1482
box commands:	
<code>\box_clear_new:N</code>	643
<code>\box_dp:N</code> ..	312, 324, 374, 486, 492, 496, 806
<code>\box_ht:N</code> ..	314, 319, 322, 488, 490, 494, 805
<code>\box_move_down:nn</code>	375, 776
<code>\box_move_up:nn</code>	389, 767
<code>\box_set_dp:Nn</code>	806
<code>\box_set_ht:Nn</code>	805
<code>\box_use:N</code>	355, 373, 389, 873, 951
<code>\box_use_drop:N</code>	784, 797, 807
<code>\box_wd:N</code>	331, 379, 632, 640, 846, 917
<code>\l_tmpa_box</code>	290, 312, 314, 319, 322,
	324, 331, 355, 364, 373, 625, 632, 633, 640,
	787, 805, 806, 807, 821, 846, 873, 892, 917, 951
<code>\l_tmpb_box</code>	372, 374, 379, 389
C	
<code>\Cdots</code>	509
<code>\cdots</code>	521, 1669
Cdots internal commands:	
<code>_nm_Cdots</code>	509, 521, 1685
cdots internal commands:	
<code>_nm_cdots</code>	1669, 1688
char commands:	
<code>\char_set_catcode_letter:N</code>	2224
<code>\cleaders</code>	1841
<code>\coordinate</code>	383,
	388, 681, 698, 709, 721, 2144, 2147, 2176, 2181
<code>\crrcr</code>	676
cs commands:	
<code>\cs_generate_variant:Nn</code>	358, 1315, 2031, 2032
<code>\cs_gset:Npn</code>	1102, 1109, 2024, 2133
<code>\cs_gset:Npx</code>	1675

<code>\cs_gset_eq:NN</code>	113, 479	<code>\dim_set:Nn</code> 74, 76, 180, 235, 358, 374, 470, 746, 752, 1133, 1134, 1519, 1521, 1561, 1563, 1582, 1623, 1628, 1840, 1860, 2013, 2041, 2048, 2060, 2064, 2069, 2073, 2085, 2098
<code>\cs_if_exist:NTF</code>	460, 463, 581, 584, 600, 607, 1031, 1045, 1090, 1160, 1161, 1320, 1332, 1364, 1379, 1412, 1427, 1858, 1894, 1926, 2011, 2054	<code>\dim_set_eq:NN</code>
<code>\cs_if_exist_p:N</code>	1812, 1814	624, 635, 1859, 2039, 2046, 2093, 2108
<code>\cs_if_free:NTF</code>	1035, 1215, 1276, 1351, 1402, 1450, 1497, 1539	<code>\dim_sub:Nn</code>
<code>\cs_new:Npn</code>	441, 1715, 1726, 1789, 1853, 2200, 2201, 2225	2111
<code>\cs_new_protected:Nn</code>	280, 300, 326, 359, 1078, 1087, 1158, 1163, 1294, 1316, 1349, 1358, 1400, 1448, 1495, 1537, 1579, 1673, 1709, 1749, 1800, 1881, 2033, 2138, 2173	<code>\dim_use:N</code>
<code>\cs_new_protected:Npn</code>	18, 19, 20, 21, 22, 23, 24, 25, 56, 116, 307, 417, 428, 443, 458, 1003, 1791, 1945, 2168, 2210, 2234, 2273	1588, 1589, 1592, 1593, 2026, 2088, 2089, 2101, 2103, 2145, 2146, 2148, 2149, 2178, 2179, 2183, 2184
<code>\cs_set:Npn</code>	435, 469, 473, 497, 1165, 1205, 1266, 1787, 1837, 2120, 2170, 2171	<code>\dim_zero:N</code>
<code>\cs_set_eq:NN</code>	104, 432, 433, 434, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 520, 521, 522, 523, 524, 525, 526, 533, 534, 535, 537, 1151, 1668, 1669, 1670, 1671, 1672, 1714, 1835, 1993	749, 755, 2007
<code>\cs_set_protected:Npn</code>	69, 98, 415, 590, 2255, 2305, 2307	<code>\dim_zero_new:N</code>
		1119, 1120, 1121, 1122, 1581, 1804, 1805, 1806, 1807, 1888, 1889, 1890, 1891, 2038, 2040, 2045, 2047
		<code>\c_max_dim</code>
		1904, 2039, 2041, 2046, 2048
		<code>\g_tmpa_dim</code>
		689, 693, 696, 707, 718, 1066, 1071, 2124, 2134
		<code>\l_tmpa_dim</code>
		374, 375, 389, 746, 749, 768, 793, 805, 1623, 1662, 1859, 1860, 1874
		<code>\g_tmpb_dim</code>
		1068, 1071, 2128, 2134
		<code>\l_tmpb_dim</code> 752, 755, 778, 800, 806, 1628, 1663
		<code>\c_zero_dim</code>
		336, 337, 407, 548, 624, 635, 673, 853, 854, 931, 932, 1598, 2154, 2190, 2532, 2540, 2548, 2556, 2564, 2572, 2581, 2590, 2608, 2617, 2625
		<code>\dots</code>
		525
D		E
<code>\Ddots</code>	511	<code>\egroup</code>
<code>\ddots</code>	523, 1671	356
Ddots internal commands:		else commands:
<code>__nm_Ddots</code>	511, 523, 1697	<code>\else:</code>
ddots internal commands:		58
<code>__nm_ddots</code>	1671, 1700	<code>\end</code>
<code>\DeclareOption</code>	12, 13	1020, 1069, 1313, 1327, 1345, 1665, 1825, 1903, 1912, 1925, 1936, 2119, 2129, 2252, 2303, 2314
dim commands:		<code>\endarray</code>
<code>\dim_abs:n</code>	1071	728
<code>\dim_add:Nn</code>	2114	<code>\endBNiceArray</code>
<code>\dim_compare:nNnTF</code>	1070, 1598	2551, 2594
<code>\dim_compare_p:nNn</code>	548, 673, 1468, 1481	<code>\endbNiceArray</code>
<code>\dim_eval:n</code>	2134	2543, 2585
<code>\dim_gadd:Nn</code>	1396, 1397, 1639, 1647, 1662, 1663, 1937, 1938	<code>\endBNiceMatrix</code>
<code>\dim_gset:Nn</code>	311, 313, 318, 321, 323, 330, 486, 488, 490, 492, 494, 496, 619, 620, 632, 640, 689, 693, 842, 913, 1066, 1068, 1305, 1306, 1311, 1312, 1326, 1344, 1484, 1525, 1567, 1820, 1821, 1823, 1824, 1900, 1901, 1902, 1904, 1910, 1919, 1920, 1923, 1924, 1932, 1935, 2124, 2128	2271
<code>\dim_gset_eq:NN</code>	303, 1393, 1395, 1441, 1443, 1489	<code>\endbNiceMatrix</code>
<code>\dim_gzero:N</code>	304, 305	2268
<code>\dim_gzero_new:N</code>	485, 487, 489, 491, 493, 495, 589, 615, 616, 1296, 1297, 1298, 1299, 2000	<code>\endNiceArrayWithDelims</code>
<code>\dim_max:nn</code>	312, 314, 319, 322, 324, 331, 844, 915, 1486, 1911, 2032, 2070, 2074	962, 970, 978, 986, 994, 1002, 2620, 2629
<code>\dim_min:nn</code>	1486, 2031, 2061, 2065	<code>\endpNiceArray</code>
<code>\dim_new:N</code>	52, 73, 75, 141, 142, 143, 144, 145, 146	2535, 2576
<code>\dim_ratio:nn</code>	1529, 1571, 1605, 1609, 1616, 1620, 1626, 1631, 1642, 1650	<code>\endpNiceMatrix</code>
		2259
		<code>\endVNiceArray</code>
		2567, 2612
		<code>\endvNiceArray</code>
		2559, 2603
		<code>\endVNiceMatrix</code>
		2265
		<code>\endvNiceMatrix</code>
		2262
		<code>\everycr</code>
		483, 499
		exp commands:
		<code>\exp_after:wN</code>
		120
		<code>\exp_args:NV</code>
		540, 667
		<code>\exp_not:n</code>
		2220
		<code>\ExplSyntaxOff</code>
		1113, 2028, 2136
		<code>\ExplSyntaxOn</code>
		1099, 2021, 2130
		F
		<code>\fi</code>
		189
		fi commands:
		<code>\fi:</code>
		60
		fp commands:
		<code>\fp_to_dim:n</code>
		1584

G

group commands:

 \group_begin: 102, 623, 1089, 2223

 \group_end: 107, 641, 1154, 2233

H

\halign 501, 503

hbox commands:

 \hbox:n 43, 45, 47, 385, 794, 2205

 \hbox_overlap_left:n 848

 \hbox_overlap_right:n 920, 1861

 \hbox_set:Nn 372, 625, 633, 787

 \hbox_set:Nw 290, 364, 663, 821, 892

 \hbox_set_end: 329, 370, 732, 840, 911

 \hbox_to_wd:nn 379, 1842, 1872

\Hdotsfor 515

\hdotsfor 526

\hdottedline 513

\hfil 381, 537

\hfill 1850

\hrule 452

\Hspace 514

\hspace 1712

\hss 537, 1847

I

\ialign 473, 497

\iddots 512

\iddots 38, 524, 1672

ldots internal commands:

 _nm_iddots 512, 524, 1703

iddots internal commands:

 _nm_iddots 1672, 1706

if commands:

 \if_mode_math: 58

\ifstandalone 584

int commands:

 \int_add:Nn 1173, 1174, 1257, 1258

 \int_compare:nNnTF

 283, 285, 293, 294, 296, 309, 316,

 448, 450, 551, 595, 645, 654, 677, 733, 737,

 744, 750, 756, 763, 772, 1013, 1048, 1080,

 1095, 1176, 1178, 1182, 1184, 1188, 1190,

 1236, 1238, 1242, 1244, 1248, 1250, 1517,

 1559, 1718, 1755, 1769, 1883, 1947, 1949,

 1951, 1953, 1955, 1960, 1962, 1968, 1970,

 1976, 1978, 1980, 1985, 1987, 2203, 2227,

 2285, 2287, 2289, 2293, 2295, 2297, 2299, 2301

 \int_compare:nTF 241

 \int_compare_p:nNn 826,

 829, 831, 897, 900, 902, 1869, 2238, 2239

 \int_eval:n 1015, 1721,

 1790, 2089, 2093, 2104, 2108, 2183, 2218, 2219

 \int_gadd:Nn 1724

 \int_gdecr:N 1093

 \int_gincr:N 282, 302, 587, 706, 717, 889, 2006

 \int_gset:Nn 288, 530, 561, 699, 890

 \int_gset_eq:NN

 553, 740, 1092, 1094, 1160, 1161

 \int_gsub:Nn 1096

 \int_gzero:N 445

 \int_gzero_new:N 462, 465, 531, 532, 559

 \int_incr:N 1516, 1558

 \int_max:nn 289, 891

 \int_new:N 50, 51, 79, 80, 81, 83, 85, 88

 \int_set:Nn 82, 84, 86,

 89, 256, 602, 609, 1166, 1167, 1168, 1169,

 1604, 1608, 1615, 1619, 1633, 1753, 1754,

 1757, 1761, 1763, 1765, 1771, 1775, 1777,

 1779, 2081, 2082, 2275, 2276, 2532, 2540,

 2548, 2556, 2564, 2572, 2573, 2581, 2582,

 2590, 2591, 2600, 2608, 2609, 2617, 2625, 2626

 \int_set_eq:NN 461, 464

 \int_step_inline:nn 1905

 \int_step_inline:nnn 1656, 1786

 \int_step_variable:nNn 2083, 2096

 \int_step_variable:nnNn

 2036, 2043, 2050, 2052, 2140, 2142

 \int_sub:Nn 1197, 1198, 1233, 1234

 \int_use:N 343, 344, 345, 350, 351,

 397, 398, 399, 404, 405, 423, 424, 555, 600,

 603, 681, 698, 711, 712, 723, 724, 862, 863,

 869, 940, 941, 942, 947, 948, 1032, 1038,

 1039, 1040, 1046, 1049, 1061, 1062, 1063,

 1102, 1103, 1110, 1148, 1208, 1209, 1218,

 1219, 1220, 1226, 1269, 1270, 1279, 1280,

 1281, 1287, 1302, 1303, 1304, 1308, 1309,

 1310, 1321, 1325, 1334, 1335, 1341, 1342,

 1367, 1368, 1369, 1382, 1383, 1384, 1415,

 1416, 1417, 1430, 1431, 1432, 1676, 1677,

 1721, 1742, 1743, 1812, 1814, 1858, 1859,

 1895, 1922, 1927, 1934, 2011, 2014, 2025,

 2055, 2058, 2068, 2115, 2123, 2126, 2127,

 2133, 2155, 2191, 2216, 2217, 2366, 2367, 2368

 \int_zero:N 193, 194, 266, 271, 274, 275

 \int_zero_new:N

 1117, 1118, 1125, 1126, 1127, 1128, 2280, 2281

 \c_one_int 241, 283, 285,

 316, 1096, 1354, 1405, 1453, 1500, 1517, 1559

 \g_tmpa_int 699, 706, 712, 717, 724

 \l_tmpa_int 1604, 1608, 1615,

 1619, 1643, 1651, 1656, 1761, 1762, 1775, 1776

 \l_tmpb_int 1633, 1645, 1653

 \c_zero_int 293, 294, 309,

 645, 744, 756, 763, 826, 897, 1080, 1354,

 1405, 1453, 1869, 1883, 1947, 1949, 1951,

 1953, 1960, 1968, 1976, 1985, 2285, 2287,

 2293, 2299, 2573, 2582, 2591, 2600, 2609, 2626

iow commands:

 \iow_now:Nn 1099, 1100,

 1107, 1113, 2021, 2022, 2028, 2130, 2131, 2136

K

\kern 47

keys commands:

 \keys_define:nn

 147, 175, 198, 220, 252, 259, 269, 1995, 2321

 \l_keys_key_tl 2395, 2400, 2426, 2460

 \keys_set:nn 251, 592, 593, 1008, 2008

 \l_keys_value_tl 2344, 2491

L

\Ldots 508

\ldots 520, 1668

Ldots internal commands:

 _nm_Ldots 508, 520, 525, 1679

ldots internal commands:

 _nm_ldots 1668, 1682

\left 628, 637, 790
\line 1151, 2335
\lineskip 747, 753

M

\makebox 373
math commands:
\c_math_toggle_token
..... 291, 328, 627, 629, 636, 638, 666,
729, 789, 803, 822, 839, 893, 910, 1845, 1848
\mathinner 40
\mkern 42, 44, 46, 47
msg commands:
\msg_error:nn 18
\msg_error:nnn 19, 1225, 1286
\msg_error:nnnn 20, 2241
\msg_fatal:nn 21, 22
\msg_new:nnn 23
\msg_new:nnnn 24
\msg_redirect_name:nnn 26
\msg_warning:nn 37
\multicolumn .. 516, 1714, 1728, 1734, 1746, 2205
\myfiledate 8
\myfileversion 9

N

\newcolumntype 361, 505, 506, 507, 540
\NewDocumentCommand 250,
1679, 1685, 1691, 1697, 1703, 1733, 1737, 2278
\NewDocumentEnvironment
..... 570, 955, 963, 971, 979, 987,
995, 1005, 2004, 2528, 2536, 2544, 2552,
2560, 2568, 2577, 2586, 2595, 2604, 2613, 2621
\NewExpandableDocumentCommand 2198
\NiceArrayWithDelims
... 960, 968, 976, 984, 992, 1000, 2618, 2627
\NiceMatrixOptions 250, 2401
nm internal commands:
__nm_actualization_for_first_and_
last_row: 307, 332, 841, 912
__nm_actually_draw_Ldots: 1355, 1358, 1785
__nm_adapt_S_column: 98, 113, 577
__nm_add_to_empty_cells:
... 1673, 1683, 1689, 1695, 1701, 1707, 1711
__nm_adjust_with_col_nodes:
..... 1316, 1391, 1439
__nm_after_array: 814, 1078
__nm_after_array_i: 1081, 1087
__nm_array: 428, 667
\l_nm_auto_columns_width_bool
..... 137, 179, 547, 672, 685, 2001
__nm_begin_of_row: 286, 300, 820
__nm_Block: 517, 2198
\l_nm_block_auto_columns_width_bool ..
..... 588, 686, 1994, 1999, 2009, 2019
__nm_Block_i 2199, 2200
__nm_Block_ii:nnnn 2200, 2201
__nm_Block_iii:nnnn 2208, 2210
__nm_Block_iv:nnnnn 2215, 2234
\g_nm_Cdots_lines_tl 563, 1140
__nm_Cell: 123, 280, 365, 505, 506, 507
\g_nm_code_after_tl
..... 78, 191, 554, 1152, 1153, 2213
\l_nm_code_for_first_col_tl 149, 834

\l_nm_code_for_first_row_tl 153, 294
\l_nm_code_for_last_col_tl 151, 905
\l_nm_code_for_last_row_tl 155, 297
\g_nm_col_total_int 288, 289,
532, 701, 702, 890, 891, 1092, 2043, 2053, 2142
\c_nm_colortbl_loaded_bool ... 63, 68, 475
\l_nm_columns_width_dim
..... 52, 180, 235, 548, 673, 694, 2007, 2013
__nm_create_extra_nodes:
..... 1124, 1291, 1292, 1784, 2033, 2120
__nm_create_nodes: 2080, 2118, 2138
\l_nm_ddots_int 1117, 1516, 1517
\g_nm_Ddots_lines_tl 566, 1138
__nm_define_com:nnn
..... 2305, 2316, 2317, 2318, 2319, 2320
__nm_define_env:n
... 1003, 1022, 1023, 1024, 1025, 1026, 1027
\l_nm_delta_x_one_dim ... 1119, 1519, 1529
\l_nm_delta_x_two_dim ... 1121, 1561, 1571
\l_nm_delta_y_one_dim ... 1120, 1521, 1529
\l_nm_delta_y_two_dim ... 1122, 1563, 1571
__nm_dotfill: 1835, 1837, 1877
__nm_double_int_eval:n .. 1789, 1796, 1797
\g_nm_dp_ante_last_row_dim
..... 303, 491, 492, 779
\g_nm_dp_last_row_dim
..... 303, 304, 323, 324, 495, 496, 753, 779
\g_nm_dp_row_zero_dim 311, 312, 485, 486, 747
\c_nm_draft_bool
..... 11, 12, 36, 414, 1731, 1802, 1834, 1886
__nm_draw_Cdots:nn 1400
__nm_draw_Ddots:nn 1495
__nm_draw_Hdotsfor:nnn 1741, 1749
__nm_draw_Iddots:nn 1537
__nm_draw_Ldots:nn 1349
__nm_draw_tikz_line:
1398, 1444, 1491, 1533, 1575, 1579, 1826, 1941
__nm_draw_Vdots:nn 1448
__nm_end_Cell: . 125, 326, 369, 505, 506, 507
\g_nm_env_int
..... 50, 343, 397, 587, 600, 603, 681,
698, 711, 723, 862, 940, 1038, 1051, 1061,
1102, 1148, 1218, 1279, 1302, 1308, 1321,
1325, 1334, 1341, 1367, 1382, 1415, 1430,
1677, 1812, 1814, 1858, 1859, 1895, 1927,
2055, 2058, 2068, 2123, 2126, 2133, 2155, 2191
__nm_error:n . 18, 224, 228, 234, 243, 246,
255, 257, 265, 267, 273, 278, 739, 1083, 1884
__nm_error:nn
..... 19, 186, 2530, 2538, 2546, 2554,
2562, 2570, 2579, 2588, 2597, 2606, 2615, 2623
__nm_error:nnn 20, 1829
__nm_everycr: 441, 480, 483
__nm_everycr_i: 442, 443
\l_nm_exterior_arraycolsep_bool
..... 132, 231, 650, 659, 1867
\l_nm_extra_left_margin_dim
..... 145, 170, 665, 878
\g_nm_extra_nodes_bool
..... 140, 472, 550, 1124, 1857, 2212
\l_nm_extra_nodes_bool 139, 163, 472
\l_nm_extra_right_margin_dim
..... 146, 171, 731, 926

_nm_extract_coords:	2168, 2175	\l_nm_inter_dots_dim	73, 74, 1134, 1605, 1609,
_nm_fatal:n	21, 59, 579		1616, 1620, 1626, 1631, 1643, 1651, 1840, 1843
_nm_fatal:nn	22	_nm_j:	2043, 2045,
\l_nm_final_i_int			2046, 2047, 2048, 2053, 2055, 2059, 2062,
....	1127, 1168, 1173, 1176, 1197, 1202,		2064, 2065, 2068, 2071, 2073, 2074, 2096,
	1208, 1219, 1226, 1309, 1383, 1431, 1754, 1776		2098, 2102, 2104, 2108, 2109, 2142, 2145,
\l_nm_final_j_int			2148, 2155, 2158, 2171, 2178, 2183, 2191, 2194
....	1128, 1169, 1174, 1182, 1188, 1198, 1202,	\l_nm_l_dim	1581, 1582, 1598, 1605, 1609,
	1209, 1220, 1310, 1384, 1432, 1771, 1777, 1779		1616, 1620, 1626, 1631, 1643, 1644, 1651, 1652
\l_nm_final_open_bool		\l_nm_last_col_bool	2599
....	1130, 1175, 1179, 1185, 1191,	\g_nm_last_col_found_bool	
	1195, 1223, 1292, 1330, 1377, 1394, 1425,		90, 562, 700, 809, 888, 1093
	1442, 1463, 1476, 1510, 1552, 1602, 1613,	\l_nm_last_col_int	
	1636, 1637, 1752, 1772, 1780, 1783, 1809, 1893	88, 89, 256, 266, 274, 654, 1013,
_nm_find_extremities_of_line:nnnn ..			1015, 2289, 2295, 2301, 2532, 2540, 2548,
....	1163, 1354, 1405, 1453, 1500, 1542		2556, 2564, 2572, 2581, 2590, 2608, 2617, 2625
\l_nm_first_col_int	83,	\l_nm_last_row_int	85,
	84, 193, 271, 285, 645, 677, 756, 1869,		86, 195, 276, 296, 450, 595, 602, 609, 733,
	1947, 2043, 2053, 2082, 2142, 2287, 2293, 2299		737, 740, 750, 772, 829, 831, 900, 902,
\l_nm_first_row_int	81, 82, 194,		1095, 1955, 1962, 1970, 1980, 1987, 2297, 2366
	275, 530, 744, 763, 1951, 2036, 2050, 2081,	\l_nm_last_row_without_value_bool ..	
	2140, 2285, 2573, 2582, 2591, 2600, 2609, 2626	87, 597, 735, 1097
_nm_gobble_ampersands:n	2206, 2225, 2230	\g_nm_last_vdotted_col_int	
_nm_Hdotsfor:	515, 526, 1726	551, 553, 559, 561
_nm_Hdotsfor_i	1729, 1733, 1737	\g_nm_Ldots_lines_tl	564, 1141
\g_nm_Hdotsfor_lines_tl ..	568, 1136, 1739	\g_nm_left_delim_dim ..	615, 619, 632, 876
_nm_hdottedline:	513, 1853	\l_nm_left_margin_dim	
\l_nm_hlines_bool	135, 158, 446	141, 164, 664, 877, 1875, 2113
_nm_Hspace:	514, 1709	\l_nm_letter_for_dotted_lines_str ..	
\g_nm_ht_last_row_dim	242, 248, 249, 539, 540, 2517
....	305, 321, 322, 493, 494, 753	_nm_line:nn	1151, 1791
\g_nm_ht_row_one_dim ..	318, 319, 489, 490, 768	_nm_line_i:nn	1795, 1800
\g_nm_ht_row_zero_dim		\g_nm_max_cell_width_dim	
....	313, 314, 487, 488, 747, 768	330, 331, 589, 690, 2000, 2026
_nm_i:	2036, 2038,	_nm_msg_new:nn	23, 34,
	2039, 2040, 2041, 2050, 2055, 2059, 2060,		2333, 2339, 2346, 2352, 2358, 2364, 2372,
	2061, 2062, 2068, 2069, 2070, 2071, 2083,		2377, 2383, 2388, 2393, 2503, 2508, 2515, 2521
	2085, 2088, 2089, 2093, 2094, 2140, 2146,	_nm_msg_new:nnn ..	24, 2398, 2424, 2458, 2489
	2149, 2155, 2158, 2170, 2179, 2184, 2191, 2194	_nm_msg_redirect_name:nn ..	25, 237, 2330
\l_nm_iddots_int	1118, 1558, 1559	_nm_multicolumn:nnn	516, 1715
\g_nm_iddots_lines_tl	567, 1139	\g_nm_multicolumn_cells_seq	
_nm_if_not_empty_cell:nn	1028	528, 1720, 2062, 2071, 2164
_nm_if_not_empty_cell:nnTF		\g_nm_multicolumn_sizes_seq ..	529, 1722, 2165
....	1202, 1262, 1762, 1776	\l_nm_name_str	138, 188,
\l_nm_impossible_line_bool			347, 349, 401, 403, 598, 607, 610, 866, 868,
....	62, 1227, 1288, 1353, 1355,		944, 946, 1105, 1109, 2157, 2158, 2193, 2194
	1404, 1406, 1452, 1454, 1499, 1501, 1541, 1543	\g_nm_names_seq	53, 185, 187, 2501
\l_nm_in_env_bool	54, 579, 580	\l_nm_nb_cols_int	
\l_nm_initial_i_int	2276, 2281, 2284, 2288, 2294, 2300
....	1125, 1166, 1233, 1236, 1257, 1263,	\l_nm_nb_rows_int	2275, 2280, 2291
	1269, 1280, 1287, 1303, 1368, 1416, 1753, 1762	\l_nm_NiceArray_bool	
\l_nm_initial_j_int	55, 591, 617, 648, 657, 761, 957, 1865
....	1126, 1167, 1234, 1242, 1248, 1258, 1263,	\g_nm_NiceMatrixBlock_int	
	1270, 1281, 1304, 1369, 1417, 1757, 1763, 1765	51, 2006, 2011, 2014, 2025
\l_nm_initial_open_bool		_nm_node_for_multicolumn:nn ..	2166, 2173
....	1129, 1235, 1239,	\l_nm_nullify_dots_bool	
	1245, 1251, 1255, 1284, 1291, 1318, 1362,	136, 162, 1682, 1688, 1694, 1700, 1706
	1392, 1410, 1440, 1458, 1471, 1505, 1547,	_nm_old_multicolumn	1714, 1717
	1600, 1635, 1751, 1758, 1766, 1783, 1808, 1892	\l_nm_parallelize_diags_bool	
_nm_instruction_of_type:n	133, 134, 159, 1115, 1514, 1556
....	415, 417, 1681, 1687, 1693, 1699, 1705	\l_nm_pos_env_str	
		130, 131, 261, 262, 263, 439, 765, 774

`_nm_pre_array:` 458, 614
`\c_nm_preamble_first_col_tl` 646, 816
`\c_nm_preamble_last_col_tl` 655, 884
`\l_nm_radius_dim` 75, 76, 1133, 1660
`\l_nm_renew_dots_bool` . 160, 230, 518, 2323
`_nm_renew_matrix:` 222, 225, 229, 2255, 2325
`_nm_renew_NC@rewrite@S:` 116, 560
`_nm_renewcolumntype:nn` 359, 536, 537
`_nm_restore_iRow_jCol:` . 1084, 1156, 1158
`_nm_retrieve_coords:nn`
1294, 1315, 1360, 1408, 1456, 1469, 1503, 1545
`\c_nm_revtex_bool` 27, 29, 32, 430
`\g_nm_right_delim_dim` .. 616, 620, 640, 924
`\l_nm_right_margin_dim`
..... 142, 166, 730, 925, 1875, 2116
`\g_nm_row_total_int`
..... 531, 1094, 1103, 1110, 2036, 2050, 2140
`\l_nm_save_iRow_int` 79, 461, 1160
`\l_nm_save_jCol_int` 80, 464, 1161
`_nm_set_size:n` 2273, 2282
`\c_nm_siunitx_loaded_bool` 91, 95, 100, 560
`\l_nm_small_bool`
... 157, 292, 467, 823, 894, 1131, 1839, 1846
`\l_nm_stop_loop_bool`
1170, 1171, 1199, 1203, 1230, 1231, 1259, 1264
`\c_nm_table_collect_begin_tl` 108, 110, 123
`\c_nm_table_print_tl` 111, 112, 125
`_nm_test_if_math_mode:`
..... 56, 578, 967, 975, 983, 991, 999
`\l_nm_the_array_box` 643, 663, 784, 797
`\g_nm_type_env_str` 77,
572, 574, 958, 959, 965, 966, 973, 974,
981, 982, 989, 990, 997, 998, 1007, 1155,
2309, 2341, 2348, 2367, 2379, 2461, 2510, 2518
`\g_nm_Vdots_lines_tl` 565, 1137
`_nm_vdottedline:n` 555, 1881
`_nm_vline:` 590, 1945
`_nm_vline_i:`
..... 69, 1956, 1963, 1971, 1981, 1988, 1993
`\g_nm_width_first_col_dim` 144, 759, 842, 845
`\g_nm_width_last_col_dim` 143, 811, 913, 916
`\g_nm_x_final_dim`
..... 1298, 1311, 1344, 1468, 1481, 1487,
1489, 1520, 1528, 1562, 1570, 1588, 1625,
1641, 1806, 1823, 1890, 1901, 1923, 1935, 1938
`\g_nm_x_initial_dim` .. 1296, 1305, 1326,
1468, 1481, 1484, 1487, 1489, 1520, 1528,
1562, 1570, 1589, 1625, 1639, 1641, 1659,
1662, 1804, 1820, 1888, 1900, 1919, 1932, 1937
`\g_nm_y_final_dim` 1299, 1312, 1393, 1395,
1397, 1441, 1443, 1522, 1525, 1564, 1567,
1592, 1630, 1649, 1807, 1824, 1891, 1902, 1924
`\g_nm_y_initial_dim` 1297, 1306,
1393, 1395, 1396, 1441, 1443, 1522, 1527,
1564, 1569, 1593, 1630, 1647, 1649, 1659,
1663, 1805, 1821, 1889, 1904, 1910, 1911, 1920
`\noalign` 442, 479, 1855
`\node` .. 340, 394, 857, 935, 2150, 2186, 2244, 2251
`\normalbaselines` 466
`\nulldelimiterspace` 624, 635

O

`\omit` 677, 678, 705, 716

P

`\path` 1818
peek commands:
`\peek_charcode_remove_ignore_spaces:NTF`
..... 2229
`\pgfpathcircle` 1658
`\pgfpoint` 1659
`\pgfpointanchor` 1065, 1067
`\pgfusepath` 1661
`\phantom` 1682, 1688, 1694, 1700, 1706
`\pNiceArray` 2533, 2574
`\pNiceMatrix` 2258
prg commands:
`\prg_do_nothing:` 104, 113, 479, 1835
`\prg_replicate:nn`
701, 702, 1734, 1746, 2288, 2291, 2294, 2300
`\prg_return_false:` 1042, 1055, 1072
`\prg_return_true:` 1033, 1073
`\prg_set_conditional:Npnn` 1028
`\ProcessKeysOptions` 2332
`\ProcessOptions` 14
`\ProvideDocumentCommand` 38
`\ProvidesExplPackage` 6

Q

quark commands:
`\q_stop` 1789,
1796, 1797, 2168, 2175, 2199, 2200, 2273, 2282

R

`\raise` 43, 45, 47
`\relax` 14, 534, 535
`\renewcommand` 118
`\RenewDocumentEnvironment`
..... 2257, 2260, 2263, 2266, 2269
`\RequirePackage` 2, 4, 5, 15, 16, 17
`\right` 628, 637, 802

S

`\scriptstyle` 292, 823, 894, 1846
seq commands:
`\seq_gclear_new:N` 528, 529
`\seq_gput_left:Nn` 187, 1720, 1722
`\seq_if_in:NnTF` 185, 2062, 2071
`\seq_mapthread_function:NNN` 2163
`\seq_new:N` 53
`\seq_use:Nnnn` 2501
skip commands:
`\skip_horizontal:N` .. 679, 696, 707, 718, 726
`\skip_horizontal:n` 544,
664, 665, 682, 719, 730, 731, 758, 759,
796, 798, 811, 812, 874, 881, 919, 922, 1871
`\skip_vertical:n` 453, 793, 800
`\c_zero_skip` 484, 500
str commands:
`\c_backslash_str` 2310
`\c_colon_str` 249
`\str_gclear:N` 1155
`\str_gset:Nn`
574, 959, 966, 974, 982, 990, 998, 1007, 2309
`\str_if_empty:NTF`
..... 347, 401, 572, 598, 866, 944,
958, 965, 973, 981, 989, 997, 1105, 2157, 2193
`\str_if_eq:nnTF` 178, 233, 765, 774

<code>\str_new:N</code>	77, 130, 138, 248	<code>\tikz@library@external@loaded</code> ...	581, 1090
<code>\str_set:Nn</code>	131, 184, 242, 261, 262, 263	<code>\tikz@parse@node</code>	1301,
<code>\str_set_eq:NN</code>	188, 249		1307, 1324, 1339, 1819, 1822, 1898, 1908,
<code>\l_tmpa_str</code>	184, 185, 187, 188		1917, 1921, 1930, 1933, 2057, 2067, 2122, 2125
T			
<code>\tabskip</code>	484, 500	tex commands:	
TeX and L ^A T _E X 2 _ε commands:			
<code>\@acol</code>	434	<code>\tex_the:D</code>	122
<code>\@acoll</code>	432	<code>\theiRow</code>	460, 1160
<code>\@acolr</code>	433	<code>\thejCol</code>	463, 1161
<code>\@addtopreamble</code>	590	<code>\tikz</code> 333, 382, 387, 393, 680, 697, 708, 720, 850, 928	
<code>\@array@array</code>	436	<code>\tikzset</code>	583, 585, 1091, 1142, 2079, 2117
<code>\@arrayacol</code>	432, 433, 434	tl commands:	
<code>\@arrayrule</code>	590	<code>\tl_const:Nn</code>	816, 884
<code>\@arstrutbox</code>	486, 488, 490, 492, 494, 496	<code>\tl_count:n</code>	241
<code>\@currentenv</code>	2523	<code>\tl_gclear:N</code>	1153
<code>\@halignto</code>	435	<code>\tl_gclear_new:N</code> 563, 564, 565, 566, 567, 568	
<code>\@height</code>	452	<code>\tl_gput_left:Nn</code>	2213
<code>\@ifclassloaded</code>	28, 31	<code>\tl_gput_right:Nn</code>	419, 554, 1739
<code>\@ifnextchar</code>	533	<code>\tl_gset:Nn</code>	106, 110, 112
<code>\@ifpackageloaded</code>	66, 94	<code>\tl_if_empty:nTF</code>	254, 264, 272
<code>\@mainaux</code>	1099, 1100,	<code>\tl_item:Nn</code>	109, 110, 112
	1107, 1113, 2021, 2022, 2028, 2130, 2131, 2136	<code>\tl_new:N</code>	78, 108, 111
<code>\@temptokena</code>	103, 106, 120, 122	<code>\tl_put_left:Nn</code>	646, 651
<code>\c@iRow</code>	293, 296, 302,	<code>\tl_put_right:Nn</code>	655, 660
	309, 316, 344, 350, 398, 404, 423, 448, 450,	<code>\tl_set:Nn</code>	109, 644, 1058
	461, 462, 530, 737, 740, 826, 831, 863, 869,	<code>\tl_set_rescan:Nnn</code>	538
	897, 902, 941, 947, 1080, 1094, 1096, 1160,	<code>\tl_use:N</code>	2395, 2400, 2426, 2460
	1176, 1676, 1721, 1742, 1922, 1934, 1953,	<code>\g_tmpa_tl</code>	106, 109, 112
	1955, 1960, 1962, 1968, 1970, 1978, 1980,	<code>\l_tmpa_tl</code>	109, 110,
	1985, 1987, 2083, 2216, 2218, 2238, 2367, 2368		644, 646, 651, 655, 660, 667, 1058, 1065, 1067
<code>\c@jCol</code>	282,	token commands:	
	283, 289, 294, 345, 351, 399, 405, 424, 445,	<code>\token_to_str:N</code>	2335, 2380, 2401
	464, 465, 551, 553, 555, 889, 891, 942,		
	948, 1092, 1093, 1161, 1188, 1248, 1335,	U	
	1342, 1676, 1721, 1724, 1743, 1769, 1905,	<code>\unless</code>	183
	1949, 1976, 2096, 2115, 2127, 2217, 2219, 2239	use commands:	
<code>\c@MaxMatrixCols</code>	1014	<code>\use:N</code>	422, 603, 610, 1049, 2014
<code>\CT@arc@</code>	69	<code>\use:n</code>	1793
<code>\CT@everycr</code>	477	<code>\usetikzlibrary</code>	3
<code>\CT@row@color</code>	479	V	
<code>\ifmeasuring@</code>	183	<code>\vbox</code>	47
<code>\NC@find</code>	104, 127	vbox commands:	
<code>\NC@find@W</code>	535	<code>\vbox:n</code>	377
<code>\NC@find@w</code>	534	<code>\vcenter</code>	628, 637, 791, 2380
<code>\NC@rewrite@S</code>	105, 118	<code>\Vdots</code>	510
<code>\new@ifnextchar</code>	533	<code>\vdots</code>	522, 1670
<code>\p@</code>	43, 45, 47	Vdots internal commands:	
<code>\pgf@x</code>	1066, 1068, 1305,	<code>__nm_Vdots</code>	510, 522, 1691
	1311, 1326, 1344, 1820, 1823, 1900, 1901,	vdots internal commands:	
	1919, 1923, 1932, 1935, 2065, 2074, 2124, 2128	<code>__nm_vdots</code>	1670, 1694
<code>\pgf@y</code>	1306, 1312,	<code>\vline</code>	69, 1993
	1821, 1824, 1902, 1911, 1920, 1924, 2061, 2070	<code>\VNiceArray</code>	2565, 2610
<code>\pgfutil@firstofone</code>	1301,	<code>\vNiceArray</code>	2557, 2601
	1307, 1324, 1339, 1819, 1822, 1898, 1908,	<code>\VNiceMatrix</code>	2264
	1917, 1921, 1930, 1933, 2057, 2067, 2122, 2125	<code>\vNiceMatrix</code>	2261

Contents

1	Presentation	1
2	The environments of this extension	2
3	The continuous dotted lines	2
3.1	The option nullify-dots	3
3.2	The command <code>\Hdotsfor</code>	4
3.3	How to generate the continuous dotted lines transparently	5
4	The Tikz nodes created by nicematrix	5
5	The code-after	6
6	The environment <code>\NiceArray</code>	7
7	The exterior rows and columns	7
8	The dotted lines to separate rows or columns	9
9	The width of the columns	10
10	Block matrices	11
11	The option small	11
12	The counters <code>iRow</code> and <code>jCol</code>	12
13	The option <code>hlines</code>	12
14	Utilisation of the column type <code>S</code> of <code>siunitx</code>	13
15	Technical remarks	13
15.1	Intersections of dotted lines	13
15.2	Diagonal lines	13
15.3	The “empty” cells	14
15.4	The option <code>exterior-arraycolsep</code>	15
15.5	The class option <code>draft</code>	15
15.6	A technical problem with the argument of <code>\\</code>	15
15.7	Obsolete environments	15
16	Examples	16
16.1	Dotted lines	16
16.2	Width of the columns	18
16.3	How to highlight cells of the matrix	19
16.4	Direct utilisation of the Tikz nodes	21

17	Implementation	22
17.1	Declaration of the package and extensions loaded	23
17.2	Technical definitions	23
17.2.1	Variables for the exterior rows and columns	26
17.2.2	The column S of siunitx	27
17.3	The options	28
17.4	Important code used by <code>{NiceArrayWithDelims}</code>	33
17.5	The environment <code>{NiceArrayWithDelims}</code>	40
17.6	The environment <code>{NiceMatrix}</code> and its variants	48
17.7	How to know whether a cell is “empty”	49
17.8	After the construction of the array	50
17.9	The actual instructions for drawing the dotted line with Tikz	61
17.10	User commands available in the new environments	63
17.11	The command <code>\line</code> accessible in code-after	65
17.12	The commands to draw dotted lines to separate columns and rows	66
17.13	The vertical rules	69
17.14	The environment <code>{NiceMatrixBlock}</code>	70
17.15	The extra nodes	71
17.16	Block matrices	74
17.17	How to draw the dotted lines transparently	76
17.18	Automatic arrays	76
17.19	We process the options	77
17.20	Error messages of the package	77
17.21	Obsolete environments	81
18	History	83
	Index	86