

The package `nicematrix`*

F. Pantigny
 fpantigny@wanadoo.fr

October 29, 2019

Abstract

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

1 Presentation

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

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

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

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

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 3.6 of `nicematrix`, at the date of 2019/10/29.

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>	$\begin{bmatrix} a_1 & \cdots & \cdots & \cdots & a_1 \\ \vdots & & & & \vdots \\ \vdots & a_2 & \cdots & \cdots & a_2 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ a_1 & a_2 & \cdots & \cdots & a_n \end{bmatrix}$
<code>\Vdots & & \Cdots & & \Vdots \\</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>	$\begin{bmatrix} 0 & \cdots & 0 \\ \vdots & & \vdots \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$
<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>	$\begin{bmatrix} 0 & \cdots & \cdots & 0 \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix}$
<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.

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

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

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

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

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

3.2 The command `\Hdotsfor`

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

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

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

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

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

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

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

3.3 How to generate the continuous dotted lines transparently

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

- The option `renew-dots`

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

- The option `renew-matrix`

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

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

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

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

4 The Tikz nodes created by `nicematrix`

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

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

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

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

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

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

In fact, the package `nicematrix` can create “extra nodes”. 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.

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

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

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

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

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

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

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

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

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

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

5 The code-after

The option `code-after` may be used to give some code that will be excuted after the construction of the matrix (and, hence, after the construction of all the Tikz nodes).

In the `code-after`, the Tikz nodes should be accessed by a name of the form $i-j$ (without the prefix of the name of the environment).

Moreover, a special command, called `\line` is available to draw directly dotted lines between nodes.

```
\begin{pNiceMatrix}[code-after = \line{1-1}{3-3}]
0 & 0 & 0 \\
0 & & 0 \\
0 & 0 & 0
\end{pNiceMatrix}
```

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

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

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

6 The environment `{NiceArray}`

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

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

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

```


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


```

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 `\newcolumnstype` of `array`. This definition overrides an eventual previous definition. In fact, the column types `w` and `W` are also redefined.

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

```

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

```

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

We have several remarks to do.

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

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

```

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

```

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

Remarks

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

8 The dotted lines to separate rows or columns

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

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

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

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

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

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}
```

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

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

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

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

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

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

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

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

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

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

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

Several compilations may be necessary to achieve the job.

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

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

10 Block matrices

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

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

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

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

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`).

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

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

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 represents 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{\alpha{jCol}} ,
     code-for-first-col = \mathbf{\arabic{iRow}} ]
& & & & \\
& 1 & 2 & 3 & 4 \\
& 5 & 6 & 7 & 8 \\
& 9 & 10 & 11 & 12
\end{pNiceMatrix}

```

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

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`.

13 The option `hlines`

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

```

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

```

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

14 Utilisation of the column type `S` of `siunitx`

If the package `siunitx` is loaded (before or after `nicematrix`), it's possible to use the `S` column type of `siunitx` in the environments of `nicematrix`. The implementation doesn't use explicitly any private macro of `siunitx`.

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).

```

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

```

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

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

15 Technical remarks

15.1 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.

```

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

```

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

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:

Example with parallelization (default):

```

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

```

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

```

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

```

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

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`.

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

The same example without parallelization:

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

15.3 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).

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@{}}\dots\end{array}`.

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

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

<pre>\begin{pNiceMatrix} a & \frac{AB}{2mm} \noalign{\kern2mm} b & c \end{pNiceMatrix}</pre>	$\begin{pmatrix} a & \frac{A}{B} \\ b & c \end{pmatrix}$
--	--

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

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

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

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

16 Examples

16.1 Dotted lines

A tridiagonal matrix:

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.

`$\begin{pNiceMatrix}[nullify-dots]`

```
a      & b      & 0      &      & \Cdots & 0      & \\
b      & a      & b      & \Ddots &      & \Vdots & \\
0      & b      & a      & \Ddots &      &      & \\
      & \Ddots & \Ddots & \Ddots &      & 0      & \\
\Vdots &      &      &      &      & b      & \\
0      & \Cdots &      & 0      & b      & a      & \\
\end{pNiceMatrix}$
```

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

A permutation matrix:

`$\begin{pNiceMatrix}`

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

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

An example with `\Iddots`:

`$\begin{pNiceMatrix}`

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

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

An example with `\multicolumn`:

`\begin{BNiceMatrix}[nullify-dots]`

```
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\Cdots & & \multicolumn{6}{C}{10 \text{ other rows}} & \Cdots \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\end{BNiceMatrix}
```

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

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}
```

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

An example for the resultant of two polynomials:

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

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

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

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

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

16.2 Width of the columns

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

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

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

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]
\begin{tikzpicture}
\foreach \i in {1,...,4}
\foreach \j in {1,...,4}
\draw[fit=(\i-\j)] (0,0) rectangle (1,1);
\end{tikzpicture}
\end{pNiceArray}

```

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

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

```

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

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

```

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

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

```

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

```

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

```

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

```

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

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

```

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

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

```

We obtain the following matrix.

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

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

```

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

```

We obtain the following matrix.

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

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

```

\begin{pNiceArray}{>{\strut}CCC}%
[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}
&

```

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

```

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

```

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

```

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

```

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

```

\begin{bNiceArray}{CC>{\strut}CCC}
& & & \\
& & \vdots & \\
\cdots & & c_{ij} & \\
\\
\\

```

```

\end{bNiceArray}
\end{array}$

\end{NiceMatrixBlock}

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

```

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

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

```

1 <@@=nm>
2 \RequirePackage{tikz}
3 \usetikzlibrary{fit}
4 \RequirePackage{expl3}[2019/07/01]

```

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

```

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

```

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

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

```

11 \bool_new:N \c__nm_draft_bool
12 \DeclareOption { draft } { \bool_set_true:N \c__nm_draft_bool }
13 \DeclareOption* { }
14 \ProcessOptions \relax

```

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

We load `array` and `amsmath`.

```

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

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

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

```

17.2 Technical definitions

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

```

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

```

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

```

34 \__nm_msg_new:nn { Draft~mode }
35 { The~compilation~is~in~draft~mode:~the~dotted~lines~won't~be~drawn. }

36 \bool_if:NT \c__nm_draft_bool
37 { \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.

```

38 \ProvideDocumentCommand \iddots { }
39 {
40   \mathinner
41   {
42     \mkern 1 mu
43     \raise \p@ \hbox:n { . }
44     \mkern 2 mu
45     \raise 4 \p@ \hbox:n { . }
46     \mkern 2 mu
47     \raise 7 \p@ \vbox { \kern 7 pt \hbox:n { . } } \mkern 1 mu
48   }
49 }

```

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.

```
50 \int_new:N \g__nm_env_int
```

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

```
51 \int_new:N \g__nm_NiceMatrixBlock_int
```

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

```
52 \dim_new:N \l__nm_columns_width_dim
```

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

```
53 \seq_new:N \g__nm_names_seq
```

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

```
54 \bool_new:N \l__nm_in_env_bool
```

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

```
55 \bool_new:N \l__nm_NiceArray_bool
```

```
56 \cs_new_protected:Npn \__nm_test_if_math_mode:
57 {
58   \if_mode_math: \else:
59     \__nm_fatal:n { Outside-math-mode }
60   \fi:
61 }
```

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

```
62 \bool_new:N \l__nm_impossible_line_bool
```

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

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

```

63 \bool_new:N \c__nm_colortbl_loaded_bool
64 \AtBeginDocument
65 {
66   \@ifpackageloaded { colortbl }
67   {
68     \bool_set_true:N \c__nm_colortbl_loaded_bool
69     \cs_set_protected:Npn \__nm_vline_i: { { \CT@arc@ \vline } }
70   }
71   { }
72 }

```

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.

```

73 \dim_new:N \l__nm_inter_dots_dim
74 \dim_set:Nn \l__nm_inter_dots_dim { 0.45 em }

```

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

```

75 \dim_new:N \l__nm_radius_dim
76 \dim_set:Nn \l__nm_radius_dim { 0.53 pt }

```

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

```

77 \str_new:N \g__nm_type_env_str
78 \tl_new:N \g__nm_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.

```

79 \int_new:N \l__nm_save_iRow_int
80 \int_new:N \l__nm_save_jCol_int

```

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

17.2.1 Variables for the exterior rows and columns

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

- **First row**

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

```

81 \int_new:N \l__nm_first_row_int
82 \int_set:Nn \l__nm_first_row_int 1

```

- **First column**

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

```

83 \int_new:N \l__nm_first_col_int
84 \int_set:Nn \l__nm_first_col_int 1

```

- **Last row**

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

```
85 \int_new:N \l__nm_last_row_int
86 \int_set:Nn \l__nm_last_row_int { -2 }
```

If, in an environment like `{pNiceArray}`, the option `last-row` is used without value, we will globally raise the following flag. It will be used to know if we have, after the construction of the array, to write in the `aux` file the number of the “last row”.²²

```
87 \bool_new:N \l__nm_last_row_without_value_bool
```

- **Last column**

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

```
88 \int_new:N \l__nm_last_col_int
89 \int_set:Nn \l__nm_last_col_int { -1 }
```

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

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

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

```
90 \bool_new:N \g__nm_last_col_found_bool
```

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

17.2.2 The column `S` of `siunitx`

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

```
91 \bool_new:N \c__nm_siunitx_loaded_bool
92 \AtBeginDocument
93 {
94   \ifpackageloaded { siunitx }
95     { \bool_set_true:N \c__nm_siunitx_loaded_bool }
96     { }
97 }
```

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:

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.

```

\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:`.

```

98 \cs_set_protected:Npn \_nm_adapt_S_column:
99 {

```

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

```

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

```

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

```

104     \cs_set_eq:NN \NC@find \prg_do_nothing:
105     \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`).

```

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

```

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`).

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).

```

113     \cs_gset_eq:NN \__nm_adapt_S_column: \prg_do_nothing:
114 }
115 }
```

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).

```

116 \cs_new_protected:Npn \__nm_renew_NC@rewrite@S:
117 {
118   \renewcommand*{\NC@rewrite@S}[1] []
119   {
120     \@temptokena \exp_after:wN
121     {
122       \tex_the:D \@temptokena
123       > { \__nm_Cell: \c_@@_table_collect_begin_tl S {##1} }
124       c
125       < { \c_@@_table_print_tl \__nm_end_Cell: }
126     }
127     \NC@find
128   }
129 }
```

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.

```

130 \str_new:N \l__nm_pos_env_str
131 \str_set:Nn \l__nm_pos_env_str c
```

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

```

132 \bool_new:N \l__nm_exterior_arraycolsep_bool
```

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

```

133 \bool_new:N \l__nm_parallelize_diags_bool
134 \bool_set_true:N \l__nm_parallelize_diags_bool
```

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

```

135 \bool_new:N \l__nm_hlines_bool
```

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

```

136 \bool_new:N \l__nm_nullify_dots_bool
```

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

```

137 \bool_new:N \l__nm_auto_columns_width_bool
```

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

```
138 \str_new:N \l__nm_name_str
```

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

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

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

```
141 \dim_new:N \l__nm_left_margin_dim
142 \dim_new:N \l__nm_right_margin_dim
143 \dim_new:N \g__nm_width_last_col_dim
144 \dim_new:N \g__nm_width_first_col_dim
```

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

```
145 \dim_new:N \l__nm_extra_left_margin_dim
146 \dim_new:N \l__nm_extra_right_margin_dim
```

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

```
147 \keys_define:nn { NiceMatrix / Global }
148 {
149   code-for-first-col .tl_set:N = \l__nm_code_for_first_col_tl ,
150   code-for-first-col .value_required:n = true ,
151   code-for-last-col .tl_set:N = \l__nm_code_for_last_col_tl ,
152   code-for-last-col .value_required:n = true ,
153   code-for-first-row .tl_set:N = \l__nm_code_for_first_row_tl ,
154   code-for-first-row .value_required:n = true ,
155   code-for-last-row .tl_set:N = \l__nm_code_for_last_row_tl ,
156   code-for-last-row .value_required:n = true ,
157   small .bool_set:N = \l__nm_small_bool ,
158   hlines .bool_set:N = \l__nm_hlines_bool ,
159   parallelize-diags .bool_set:N = \l__nm_parallelize_diags_bool ,
```

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

```
160   renew-dots .bool_set:N = \l__nm_renew_dots_bool ,
161   renew-dots .value_forbidden:n = true ,
162   nullify-dots .bool_set:N = \l__nm_nullify_dots_bool ,
```

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

```
163   create-extra-nodes .bool_set:N = \l__nm_extra_nodes_bool ,
164   left-margin .dim_set:N = \l__nm_left_margin_dim ,
165   left-margin .default:n = \arraycolsep ,
166   right-margin .dim_set:N = \l__nm_right_margin_dim ,
167   right-margin .default:n = \arraycolsep ,
168   margin .meta:n = { left-margin = #1 , right-margin = #1 } ,
169   margin .default:n = \arraycolsep ,
170   extra-left-margin .dim_set:N = \l__nm_extra_left_margin_dim ,
171   extra-right-margin .dim_set:N = \l__nm_extra_right_margin_dim ,
172   extra-margin .meta:n =
173     { extra-left-margin = #1 , extra-right-margin = #1 } ,
174 }
```

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

```

175 \keys_define:nn { NiceMatrix / Env }
176 {
177   columns-width .code:n =
178     \str_if_eq:nnTF { #1 } { auto }
179     { \bool_set_true:N \l__nm_auto_columns_width_bool }
180     { \dim_set:Nn \l__nm_columns_width_dim { #1 } } ,
181   columns-width .value_required:n = true ,
182   name .code:n =
183     \unless \ifmeasuring@
184       \str_set:Nn \l_tmpa_str { #1 }
185       \seq_if_in:NVTf \g__nm_names_seq \l_tmpa_str
186       { \__nm_error:nn { Duplicate~name } { #1 } }
187       { \seq_gput_left:Nv \g__nm_names_seq \l_tmpa_str }
188       \str_set_eq:NN \l__nm_name_str \l_tmpa_str
189     \fi ,
190   name .value_required:n = true ,
191   code-after .tl_gset:N = \g__nm_code_after_tl ,
192   code-after .value_required:n = true ,
193   first-col .code:n = \int_zero:N \l__nm_first_col_int ,
194   first-row .code:n = \int_zero:N \l__nm_first_row_int ,
195   last-row .int_set:N = \l__nm_last_row_int ,
196   last-row .default:n = -1 ,
197 }

```

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

```

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

```

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

```

220 \keys_define:nn { NiceMatrix / NiceMatrixOptions }
221 {

```

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

```

222   renew-matrix .code:n = \__nm_renew_matrix: ,
223   renew-matrix .value_forbidden:n = true ,
224   RenewMatrix .code:n = \__nm_error:n { Option~RenewMatrix~suppressed }
225   \__nm_renew_matrix: ,

```

```

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

```

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

```

231 exterior-arraycolsep .bool_set:N = \l__nm_exterior_arraycolsep_bool ,

```

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

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

```

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

```

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

```

236 allow-duplicate-names .code:n =
237 \__nm_msg_redirect_name:nn { Duplicate-name } { none } ,
238 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`).

```

239 letter-for-dotted-lines .code:n =
240 {
241 \int_compare:nTF { \tl_count:n { #1 } = \c_one_int }
242 { \str_set:Nx \l__nm_letter_for_dotted_lines_str { #1 } }
243 { \__nm_error:n { Bad-value-for-letter-for-dotted-lines } }
244 } ,
245 letter-for-dotted-lines .value_required:n = true ,

246 unknown .code:n = \__nm_error:n { Unknown-key-for-NiceMatrixOptions }
247 }

248 \str_new:N \l__nm_letter_for_dotted_lines_str
249 \str_set_eq:NN \l__nm_letter_for_dotted_lines_str \c_colon_str

```

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

```

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

```

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

```

252 \keys_define:nn { NiceMatrix / NiceMatrix }
253 {
254 last-col .code:n = \tl_if_empty:nTF {#1}
255 { \__nm_error:n { last-col-empty-for-NiceMatrix } }
256 { \int_set:Nn \l__nm_last_col_int { #1 } } ,
257 unknown .code:n = \__nm_error:n { Unknown-option-for-NiceMatrix }
258 }

```

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

```
259 \keys_define:nn { NiceMatrix / NiceArray }
260 {
```

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

```
261 c .code:n = \str_set:Nn \l__nm_pos_env_str c ,
262 t .code:n = \str_set:Nn \l__nm_pos_env_str t ,
263 b .code:n = \str_set:Nn \l__nm_pos_env_str b ,
```

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

```
264 last-col .code:n = \tl_if_empty:nF {#1}
265 { \__nm_error:n { last-col-non-empty-for-NiceArray } }
266 \int_zero:N \l__nm_last_col_int ,
267 unknown .code:n = \__nm_error:n { Unknown-option-for-NiceArray }
268 }
269 \keys_define:nn { NiceMatrix / pNiceArray }
270 {
271 first-col .code:n = \int_zero:N \l__nm_first_col_int ,
272 last-col .code:n = \tl_if_empty:nF {#1}
273 { \__nm_error:n { last-col-non-empty-for-NiceArray } }
274 \int_zero:N \l__nm_last_col_int ,
275 first-row .code:n = \int_zero:N \l__nm_first_row_int ,
276 last-row .int_set:N = \l__nm_last_row_int ,
277 last-row .default:n = -1 ,
278 unknown .code:n = \__nm_error:n { Unknown-option-for-NiceMatrix }
279 }
```

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}).

```
280 \cs_new_protected:Nn \__nm_Cell:
281 {
```

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

```
282 \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.

```
283 \int_compare:nNnT \c@jCol = \c_one_int
284 {
285 \int_compare:nNnT \l__nm_first_col_int = \c_one_int
286 \__nm_begin_of_row:
287 }
288 \int_gset:Nn \g__nm_col_total_int
289 { \int_max:nn \g__nm_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).

```
290 \hbox_set:Nw \l_tmpa_box
291 \c_math_toggle_token
292 \bool_if:NT \l__nm_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).

```

293 \int_compare:nNnTF \c@iRow = \c_zero_int
294 { \int_compare:nNnT \c@jCol > \c_zero_int \l__nm_code_for_first_row_tl }
295 {
296   \int_compare:nNnT \c@iRow = \l__nm_last_row_int
297   \l__nm_code_for_last_row_tl
298 }
299 }

```

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.

```

300 \cs_new_protected:Nn \__nm_begin_of_row:
301 {
302   \int_gincr:N \c@iRow
303   \dim_gset_eq:NN \g__nm_dp_ante_last_row_dim \g__nm_dp_last_row_dim
304   \dim_gzero:N \g__nm_dp_last_row_dim
305   \dim_gzero:N \g__nm_ht_last_row_dim
306 }

```

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.

```

307 \cs_new_protected:Npn \__nm_actualization_for_first_and_last_row:
308 {
309   \int_compare:nNnT \c@iRow = \c_zero_int
310   {
311     \dim_gset:Nn \g__nm_dp_row_zero_dim
312     { \dim_max:nn \g__nm_dp_row_zero_dim { \box_dp:N \l_tmpa_box } }
313     \dim_gset:Nn \g__nm_ht_row_zero_dim
314     { \dim_max:nn \g__nm_ht_row_zero_dim { \box_ht:N \l_tmpa_box } }
315   }
316   \int_compare:nNnT \c@iRow = \c_one_int
317   {
318     \dim_gset:Nn \g__nm_ht_row_one_dim
319     { \dim_max:nn \g__nm_ht_row_one_dim { \box_ht:N \l_tmpa_box } }
320   }
321   \dim_gset:Nn \g__nm_ht_last_row_dim
322   { \dim_max:nn \g__nm_ht_last_row_dim { \box_ht:N \l_tmpa_box } }
323   \dim_gset:Nn \g__nm_dp_last_row_dim
324   { \dim_max:nn \g__nm_dp_last_row_dim { \box_dp:N \l_tmpa_box } }
325 }

326 \cs_new_protected:Nn \__nm_end_Cell:
327 {
328   \c_math_toggle_token
329   \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”).

```

330 \dim_gset:Nn \g__nm_max_cell_width_dim
331 { \dim_max:nn \g__nm_max_cell_width_dim { \box_wd:N \l_tmpa_box } }

```

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

```

332 \__nm_actualization_for_first_and_last_row:

```

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

```

333 \tikz
334 [
335     remember~picture ,
336     inner~sep = \c_zero_dim ,
337     minimum~width = \c_zero_dim ,
338     baseline
339 ]
340 \node
341 [
342     anchor = base ,
343     name = nm - \int_use:N \g__nm_env_int -
344             \int_use:N \c@iRow -
345             \int_use:N \c@jCol ,
346     alias =
347         \str_if_empty:NF \l__nm_name_str
348         {
349             \l__nm_name_str -
350             \int_use:N \c@iRow -
351             \int_use:N \c@jCol
352         }
353 ]
354 \bgroup
355 \box_use:N \l_tmpa_box
356 \egroup ;
357 }
358 \cs_generate_variant:Nn \dim_set:Nn { N x }

```

In the environment `{NiceArrayWithDelims}`, we will have to redefine the column types `w` and `W`. These definitions are rather long because we have to construct the `w`-nodes in these columns. The redefinition of these two column types are very close and that's why we use a macro `\@@_renewcolumnntype:nn`. The first argument is the type of the column (`w` or `W`) and the second argument is a code inserted at a special place and which is the only difference between the two definitions.

```

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

```

```

388             \coordinate ( __nm~south~west ) ;
389             \box_move_up:nn \l_tmpa_dim { \box_use:N \l_tmpb_box }
390         }
391     }
392 }

```

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

```

393     \tikz [ remember~picture , overlay ]
394     \node
395     [
396         node~contents = { } ,
397         name = nm - \int_use:N \g__nm_env_int -
398                 \int_use:N \c@iRow -
399                 \int_use:N \c@jCol - w,
400         alias =
401             \str_if_empty:NF \l__nm_name_str
402             {
403                 \l__nm_name_str -
404                 \int_use:N \c@iRow -
405                 \int_use:N \c@jCol - w
406             } ,
407         inner~sep = \c_zero_dim ,
408         fit = ( __nm~south~west ) ( __nm~north~east )
409     ]
410 ;
411 }
412 }
413 }

```

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.

```

414 \bool_if:NTF \c__nm_draft_bool
415 { \cs_set_protected:Npn \__nm_instruction_of_type:n #1 { } }
416 {
417     \cs_new_protected:Npn \__nm_instruction_of_type:n #1
418     {

```

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).

```

419     \tl_gput_right:cx
420     { g__nm_ #1 _ lines _ tl }
421     {
422         \use:c { __nm _ draw _ #1 : nn }
423         { \int_use:N \c@iRow }

```

```

424         { \int_use:N \c@jCol }
425     }
426 }
427 }

```

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`.

```

428 \cs_new_protected:Npn \__nm_array:
429 {
430     \bool_if:NTF \c__nm_revtex_bool
431     {
432         \cs_set_eq:NN \@acol1 \@arrayacol
433         \cs_set_eq:NN \@acolr \@arrayacol
434         \cs_set_eq:NN \@acol \@arrayacol
435         \cs_set:Npn \@halignto { }
436         \@array@array
437     }
438     \array

```

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

```

439     [ \l__nm_pos_env_str ]
440 }

```

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

```

441 \cs_new:Npn \__nm_everycr:
442 { \noalign { \__nm_everycr_i: } }
443 \cs_new_protected:Npn \__nm_everycr_i:
444 {
445     \int_gzero:N \c@jCol
446     \bool_if:NT \l__nm_hlines_bool
447     {

```

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.

```

448         \int_compare:nNnT \c@iRow > { -1 }
449         {
450             \int_compare:nNnF \c@iRow = \l__nm_last_row_int
451             {
452                 \hrule \@height \arrayrulewidth
453                 \skip_vertical:n { - \arrayrulewidth }
454             }
455         }
456     }
457 }

```

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

```

458 \cs_new_protected:Npn \__nm_pre_array:
459 {
460     \cs_if_exist:NT \theiRow
461     { \int_set_eq:NN \l__nm_save_iRow_int \c@iRow }
462     \int_gzero_new:N \c@iRow
463     \cs_if_exist:NT \thejCol
464     { \int_set_eq:NN \l__nm_save_jCol_int \c@jCol }
465     \int_gzero_new:N \c@jCol
466     \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}`).

```

467 \bool_if:NT \l__nm_small_bool
468 {
469     \cs_set:Npn \arraystretch { 0.47 }
470     \dim_set:Nn \arraycolsep { 1.45 pt }
471 }

```

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`).

```

472 \bool_gset_eq:NN \g__nm_extra_nodes_bool \l__nm_extra_nodes_bool

```

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

```

473 \cs_set:Npn \ialign
474 {
475     \bool_if:NTF \c__nm_colortbl_loaded_bool
476     {
477         \CT@everycr
478         {
479             \noalign { \cs_gset_eq:NN \CT@row@color \prg_do_nothing: }
480             \__nm_everycr:
481         }
482     }
483     { \everycr { \__nm_everycr: } }
484     \tabskip = \c_zero_skip

```

The box `\@arstrutbox` is a box constructed in the beginning of the environment `{array}`. The construction of that box takes into account the current values of `\arraystretch`²⁴ and `\extrarowheight` (of `array`). That box is inserted (via `\@arstrut`) in the beginning of each row of the array. That's why we use the dimensions of that box to initialize the variables which will be the dimensions of the potential first and last row of the environment. This initialization must be done after the creation of `\@arstrutbox` and that's why we do it in the `\ialign`.

```

485 \dim_gzero_new:N \g__nm_dp_row_zero_dim
486 \dim_gset:Nn \g__nm_dp_row_zero_dim { \box_dp:N \@arstrutbox }
487 \dim_gzero_new:N \g__nm_ht_row_zero_dim
488 \dim_gset:Nn \g__nm_ht_row_zero_dim { \box_ht:N \@arstrutbox }
489 \dim_gzero_new:N \g__nm_ht_row_one_dim
490 \dim_gset:Nn \g__nm_ht_row_one_dim { \box_ht:N \@arstrutbox }
491 \dim_gzero_new:N \g__nm_dp_ante_last_row_dim
492 \dim_gset:Nn \g__nm_dp_ante_last_row_dim { \box_dp:N \@arstrutbox }
493 \dim_gzero_new:N \g__nm_ht_last_row_dim
494 \dim_gset:Nn \g__nm_ht_last_row_dim { \box_ht:N \@arstrutbox }
495 \dim_gzero_new:N \g__nm_dp_last_row_dim
496 \dim_gset:Nn \g__nm_dp_last_row_dim { \box_dp:N \@arstrutbox }

```

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

```

497 \cs_set:Npn \ialign
498 {
499     \everycr { }
500     \tabskip = \c_zero_skip
501     \halign
502 }

```

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

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

```

503     \halign
504 }

```

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}.

```

505 \newcolumntype L { > \_nm_Cell: l < \_nm_end_Cell: }
506 \newcolumntype C { > \_nm_Cell: c < \_nm_end_Cell: }
507 \newcolumntype R { > \_nm_Cell: r < \_nm_end_Cell: }

508 \cs_set_eq:NN \Ldots \_nm_Ldots
509 \cs_set_eq:NN \Cdots \_nm_Cdots
510 \cs_set_eq:NN \Vdots \_nm_Vdots
511 \cs_set_eq:NN \Ddots \_nm_Ddots
512 \cs_set_eq:NN \Iddots \_nm_Iddots
513 \cs_set_eq:NN \hdottedline \_nm_hdottedline:
514 \cs_set_eq:NN \Hspace \_nm_Hspace:
515 \cs_set_eq:NN \Hdotsfor \_nm_Hdotsfor:
516 \cs_set_eq:NN \multicolumn \_nm_multicolumn:nnn
517 \cs_set_eq:NN \Block \_nm_Block:
518 \bool_if:NT \l__nm_renew_dots_bool
519 {
520     \cs_set_eq:NN \ldots \_nm_Ldots
521     \cs_set_eq:NN \cdots \_nm_Cdots
522     \cs_set_eq:NN \vdots \_nm_Vdots
523     \cs_set_eq:NN \ddots \_nm_Ddots
524     \cs_set_eq:NN \iddots \_nm_Iddots
525     \cs_set_eq:NN \dots \_nm_Ldots
526     \cs_set_eq:NN \hdotsfor \_nm_Hdotsfor:
527 }

```

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

```

528 \seq_gclear_new:N \g__nm_multicolumn_cells_seq
529 \seq_gclear_new:N \g__nm_multicolumn_sizes_seq

```

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

```

530 \int_gset:Nn \c@iRow { \l__nm_first_row_int - 1 }

```

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

```

531 \int_gzero_new:N \g__nm_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.

```

532 \int_gzero_new:N \g__nm_col_total_int
533 \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:`.

```

534 \cs_set_eq:NN \NC@find@w \relax
535 \cs_set_eq:NN \NC@find@W \relax
536 \_nm_renewcolumntype:nn w { }
537 \_nm_renewcolumntype:nn W { \cs_set_eq:NN \hss \hfil }

```

By default, the letter used to specify a dotted line in the preamble of an environment of `nicematrix` (for example in {pNiceArray}) is the letter `.`. However, this letter is used by some extensions, for example `arydshln`. That’s why it’s possible to change the letter used by `nicematrix` with the option `letter-for-dotted-lines` which changes the value of `\l_@@_letter_for_dotted_lines_str`. 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).

```

538 \tl_set_rescan:Nno
539 \l__nm_letter_for_dotted_lines_str { } \l__nm_letter_for_dotted_lines_str
540 \exp_args:NV \newcolumntype \l__nm_letter_for_dotted_lines_str
541 {
542   !
543   {
544     \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.

```

545 \bool_if:nF
546 {
547   \l__nm_auto_columns_width_bool
548   || \dim_compare_p:nNn \l__nm_columns_width_dim > \c_zero_dim
549 }
550 { \bool_gset_true:N \g__nm_extra_nodes_bool }

```

Consider the following code:

```

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

```

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

```

551 \int_compare:nNnT \c@jCol > \g__nm_last_vdotted_col_int
552 {
553   \int_gset_eq:NN \g__nm_last_vdotted_col_int \c@jCol
554   \tl_gput_right:Nx \g__nm_code_after_tl

```

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

```

555 { \__nm_vdottedline:n { \int_use:N \c@jCol } }
556 }
557 }
558 }
559 \int_gzero_new:N \g__nm_last_vdotted_col_int
560 \bool_if:NT \c__nm_siunitx_loaded_bool \__nm_renew_NC@rewrite@S:
561 \int_gset:Nn \g__nm_last_vdotted_col_int { -1 }
562 \bool_gset_false:N \g__nm_last_col_found_bool

```

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

```

563 \tl_gclear_new:N \g__nm_Cdots_lines_tl
564 \tl_gclear_new:N \g__nm_Ldots_lines_tl
565 \tl_gclear_new:N \g__nm_Vdots_lines_tl
566 \tl_gclear_new:N \g__nm_Ddots_lines_tl
567 \tl_gclear_new:N \g__nm_Iddots_lines_tl
568 \tl_gclear_new:N \g__nm_Hdotsfor_lines_tl
569 }

```

17.5 The environment {NiceArrayWithDelims}

```

570 \NewDocumentEnvironment { NiceArrayWithDelims } { m m O { } m ! O { } }
571 {
572   \str_if_empty:NT \g__nm_type_env_str
573   { \str_gset:Nn \g__nm_type_env_str { NiceArrayWithDelims } }
574   \__nm_adapt_S_column:
575   \__nm_test_if_math_mode:
576   \bool_if:NT \l__nm_in_env_bool { \__nm_fatal:n { Yet-in~env } }
577   \bool_set_true:N \l__nm_in_env_bool

```

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

```

578   \cs_if_exist:NT \tikz@library@external@loaded
579   {
580     \tikzset { external / export = false }
581     \cs_if_exist:NT \ifstandalone
582     { \tikzset { external / optimize = false } }
583   }

```

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

```

584   \int_gincr:N \g__nm_env_int
585   \bool_if:NF \l__nm_block_auto_columns_width_bool
586   { \dim_gzero_new:N \g__nm_max_cell_width_dim }

```

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

```

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

```

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

```

588   \bool_if:NTF \l__nm_NiceArray_bool
589   { \keys_set:nn { NiceMatrix / NiceArray } }
590   { \keys_set:nn { NiceMatrix / pNiceArray } }
591   { #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).

```

592   \int_compare:nNnT \l__nm_last_row_int = { -1 }
593   {
594     \bool_set_true:N \l__nm_last_row_without_value_bool

```

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

```

595   \str_if_empty:NTF \l__nm_name_str
596   {
597     \cs_if_exist:cT { __nm_last_row_ \int_use:N \g__nm_env_int }
598     {
599       \int_set:Nn \l__nm_last_row_int
600       { \use:c { __nm_last_row_ \int_use:N \g__nm_env_int } }
601     }
602   }
603   {
604     \cs_if_exist:cT { __nm_last_row_ \l__nm_name_str }
605     {
606       \int_set:Nn \l__nm_last_row_int
607       { \use:c { __nm_last_row_ \l__nm_name_str } }
608     }
609   }
610 }

```

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

```

611   \__nm_pre_array:

```

We compute the width of the two delimiters.

```

612   \dim_gzero_new:N \g__nm_left_delim_dim
613   \dim_gzero_new:N \g__nm_right_delim_dim
614   \bool_if:NTF \l__nm_NiceArray_bool

```



```

615 {
616   \dim_gset:Nn \g__nm_left_delim_dim { 2 \arraycolsep }
617   \dim_gset:Nn \g__nm_right_delim_dim { 2 \arraycolsep }
618 }
619 {
620   \group_begin:
621   \dim_set_eq:NN \nulldelimiterspace \c_zero_dim
622   \hbox_set:Nn \l_tmpa_box
623   {
624     \c_math_toggle_token
625     \left #1 \vcenter to 3 cm { } \right.
626     \c_math_toggle_token
627   }
628   \dim_gset:Nn \g__nm_left_delim_dim { \box_wd:N \l_tmpa_box }
629   \hbox_set:Nn \l_tmpa_box
630   {
631     \dim_set_eq:NN \nulldelimiterspace \c_zero_dim
632     \c_math_toggle_token
633     \left. \vcenter to 3 cm { } \right #2
634     \c_math_toggle_token
635   }
636   \dim_gset:Nn \g__nm_right_delim_dim { \box_wd:N \l_tmpa_box }
637   \group_end:
638 }
639 }

```

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

```

640   \box_clear_new:N \l__nm_the_array_box

```

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

```

641   \tl_set:Nn \l_tmpa_tl { #4 }
642   \int_compare:nNnTF \l__nm_first_col_int = \c_zero_int
643   { \tl_put_left:NV \l_tmpa_tl \c__nm_preamble_first_col_tl }
644   {
645     \bool_if:NT \l__nm_NiceArray_bool
646     {
647       \bool_if:NF \l__nm_exterior_arraycolsep_bool
648       { \tl_put_left:Nn \l_tmpa_tl { @ { } } }
649     }
650   }
651   \int_compare:nNnTF \l__nm_last_col_int > { -1 }
652   { \tl_put_right:NV \l_tmpa_tl \c__nm_preamble_last_col_tl }
653   {
654     \bool_if:NT \l__nm_NiceArray_bool
655     {
656       \bool_if:NF \l__nm_exterior_arraycolsep_bool
657       { \tl_put_right:Nn \l_tmpa_tl { @ { } } }
658     }
659   }

```

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).

```

660   \hbox_set:Nw \l__nm_the_array_box
661   \skip_horizontal:n \l__nm_left_margin_dim
662   \skip_horizontal:n \l__nm_extra_left_margin_dim
663   \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`).

```

664   \exp_args:NV \__nm_array: \l_tmpa_tl
665 }

```

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`).

```

666 {
667   \bool_if:nT
668   {
669     \l__nm_auto_columns_width_bool
670     || \dim_compare_p:nNn \l__nm_columns_width_dim > \c_zero_dim
671   }
672   {
673     \crrc
674     \int_compare:nNnT \l__nm_first_col_int = 0 { \omit & }
675     \omit

```

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

```

676   \skip_horizontal:N \arraycolsep
677   \tikz [ remember~picture , overlay ]
678     \coordinate [ name = nm - \int_use:N \g__nm_env_int - col - 0 ] ;
679   \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.

```

680   \bool_if:nTF
681   {
682     \l__nm_auto_columns_width_bool
683     && ! \l__nm_block_auto_columns_width_bool
684   }
685   {
686     \dim_gset:Nn \g_tmpa_dim
687       { \g__nm_max_cell_width_dim + 2 \arraycolsep }
688   }
689   {
690     \dim_gset:Nn \g_tmpa_dim
691       { \l__nm_columns_width_dim + 2 \arraycolsep }
692   }
693   \skip_horizontal:N \g_tmpa_dim
694   \tikz [ remember~picture , overlay ]
695     \coordinate [ name = nm - \int_use:N \g__nm_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.

```

696   \int_gset:Nn \g_tmpa_int 1
697   \bool_if:nTF \g__nm_last_col_found_bool
698   { \prg_replicate:nn { \g__nm_col_total_int - 3 } }
699   { \prg_replicate:nn { \g__nm_col_total_int - 2 } }
700   {
701     &
702     \omit

```

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

```

703     \int_gincr:N \g_tmpa_int
704     \skip_horizontal:N \g_tmpa_dim

```

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

```

705     \tikz [ remember~picture , overlay ]
706       \coordinate
707       [
708         name = nm - \int_use:N \g__nm_env_int -
709           col - \int_use:N \g_tmpa_int
710       ] ;

```

```
711     }
```

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

```
712     &
713     \omit
714     \int_gincr:N \g_tmpa_int
715     \skip_horizontal:N \g_tmpa_dim
716     \skip_horizontal:n { - \arraycolsep }
717     \tikz [ remember~picture , overlay ]
718     \coordinate
719     [
720         name = nm - \int_use:N \g_nm_env_int -
721         col - \int_use:N \g_tmpa_int
722     ] ;
723     \skip_horizontal:N \arraycolsep
724 }
725 \endarray
726 \c_math_toggle_token
727 \skip_horizontal:n \l_nm_right_margin_dim
728 \skip_horizontal:n \l_nm_extra_right_margin_dim
729 \hbox_set_end:

730 \int_compare:nNnT \l_nm_last_row_int > { -2 }
731 {
732     \bool_if:NF \l_nm_last_row_without_value_bool
733     {
734         \int_compare:nNnF \l_nm_last_row_int = \c@iRow
735         {
736             \__nm_error:n { Wrong-last-row }
737             \int_gset_eq:NN \l_nm_last_row_int \c@iRow
738         }
739     }
740 }
```

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).

```
741 \int_compare:nNnTF \l_nm_first_row_int = \c_zero_int
742 {
743     \dim_set:Nn \l_tmpa_dim
744     { \g_nm_dp_row_zero_dim + \lineskip + \g_nm_ht_row_zero_dim }
745 }
746 { \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

```
747 \int_compare:nNnTF \l_nm_last_row_int > { -2 }
748 {
749     \dim_set:Nn \l_tmpb_dim
750     { \g_nm_ht_last_row_dim + \lineskip + \g_nm_dp_last_row_dim }
751 }
752 { \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. 45).

```
753 \int_compare:nNnT \l_nm_first_col_int = \c_zero_int
754 {
755     \skip_horizontal:n \arraycolsep
756     \skip_horizontal:n \g_nm_width_first_col_dim
757 }
```

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`).

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}`.

```

758   \bool_if:NTF \l__nm_NiceArray_bool
759   {
760     \int_compare:nNnT \l__nm_first_row_int = \c_zero_int
761     {
762       \str_if_eq:VnTF \l__nm_pos_env_str { t }
763       {
764         \box_move_up:nn
765         { \l_tmpa_dim - \g__nm_ht_row_zero_dim + \g__nm_ht_row_one_dim }
766       }
767     }
768     {
769       \int_compare:nNnT \l__nm_last_row_int > 0
770       {
771         \str_if_eq:VnTF \l__nm_pos_env_str { b }
772         {
773           \box_move_down:nn
774           {
775             \l_tmpb_dim
776             - \g__nm_dp_last_row_dim + \g__nm_dp_ante_last_row_dim
777           }
778         }
779       }
780     }
781     { \box_use_drop:N \l__nm_the_array_box }
782   }

```

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

```

783   {
784     \hbox_set:Nn \l_tmpa_box
785     {
786       \c_math_toggle_token
787       \left #1
788       \vcenter
789       {

```

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

```

790       \skip_vertical:n { - \l_tmpa_dim }
791       \hbox:n
792       {
793         \skip_horizontal:n { - \arraycolsep }
794         \box_use_drop:N \l__nm_the_array_box
795         \skip_horizontal:n { - \arraycolsep }
796       }

```

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

```

797       \skip_vertical:n { - \l_tmpb_dim }
798     }
799     \right #2
800     \c_math_toggle_token
801   }
802   \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }
803   \box_set_dp:Nn \l_tmpa_box { \box_dp:N \l_tmpa_box + \l_tmpb_dim }
804   \box_use_drop:N \l_tmpa_box
805 }

```

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. 46).

```

806   \bool_if:NT \g__nm_last_col_found_bool
807   {
808     \skip_horizontal:n \g__nm_width_last_col_dim
809     \skip_horizontal:n \arraycolsep
810   }
811   \__nm_after_array:

```

812 }

This is the end of the environment {NiceArrayWithDelims}.

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

```
813 \tl_const:Nn \c__nm_preamble_first_col_tl
814 {
815   >
816   {
817     \__nm_begin_of_row:
```

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

```
818     \hbox_set:Nw \l_tmpa_box
819     \c_math_toggle_token
820     \bool_if:NT \l__nm_small_bool \scriptstyle
821     \int_compare:nNnT \c@iRow > \c_zero_int \l__nm_code_for_first_col_tl
822   }
823   1
824   <
825   {
826     \c_math_toggle_token
827     \hbox_set_end:
828     \__nm_actualization_for_first_and_last_row:
```

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

```
829     \dim_gset:Nn \g__nm_width_first_col_dim
830     {
831       \dim_max:nn
832       \g__nm_width_first_col_dim
833       { \box_wd:N \l_tmpa_box }
834     }
```

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

```
835     \hbox_overlap_left:n
836     {
837       \tikz
838       [
839         remember~picture ,
840         inner~sep = \c_zero_dim ,
841         minimum~width = \c_zero_dim ,
842         baseline
843       ]
844       \node
845       [
846         anchor = base ,
847         name =
848           nm -
849           \int_use:N \g__nm_env_int -
850           \int_use:N \c@iRow -
851           0 ,
852         alias =
853           \str_if_empty:NF \l__nm_name_str
854           {
855             \l__nm_name_str -
856             \int_use:N \c@iRow -
857             0
858           }
859       ]
860       { \box_use:N \l_tmpa_box } ;
861     \skip_horizontal:n
862     {
863       \g__nm_left_delim_dim +
864       \l__nm_left_margin_dim +
```

```

865         \l__nm_extra_left_margin_dim
866     }
867 }
868 \skip_horizontal:n { - 2 \arraycolsep }
869 }
870 }

```

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

```

871 \tl_const:Nn \c__nm_preamble_last_col_tl
872 {
873     >
874     {

```

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

```

875     \bool_gset_true:N \g__nm_last_col_found_bool
876     \int_gincr:N \c@jCol
877     \int_gset:Nn \g__nm_col_total_int
878     { \int_max:nn \g__nm_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.

```

879     \hbox_set:Nw \l_tmpa_box
880     \c_math_toggle_token
881     \bool_if:NT \l__nm_small_bool \scriptstyle
882     \l__nm_code_for_last_col_tl
883 }
884 1
885 <
886 {
887     \c_math_toggle_token
888     \hbox_set_end:
889     \__nm_actualization_for_first_and_last_row:

```

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

```

890     \dim_gset:Nn \g__nm_width_last_col_dim
891     {
892         \dim_max:nn
893         \g__nm_width_last_col_dim
894         { \box_wd:N \l_tmpa_box }
895     }
896     \skip_horizontal:n { - 2 \arraycolsep }

```

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

```

897     \hbox_overlap_right:n
898     {
899         \skip_horizontal:n
900         {
901             \g__nm_right_delim_dim +
902             \l__nm_right_margin_dim +
903             \l__nm_extra_right_margin_dim
904         }
905         \tikz
906         [
907             remember-picture ,
908             inner~sep = \c_zero_dim ,
909             minimum~width = \c_zero_dim ,
910             baseline
911         ]
912         \node
913         [
914             anchor = base ,
915             name =
916                 nm -
917                 \int_use:N \g__nm_env_int -
918                 \int_use:N \c@iRow -
919                 \int_use:N \c@jCol ,

```

```

920         alias =
921         \str_if_empty:NF \l__nm_name_str
922         {
923             \l__nm_name_str -
924             \int_use:N \c@iRow -
925             \int_use:N \c@jCol
926         }
927     ]
928     { \box_use:N \l_tmpa_box } ;
929 }
930 }
931 }

```

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.

```

932 \NewDocumentEnvironment { NiceArray } { }
933 {
934     \bool_set_true:N \l__nm_NiceArray_bool
935     \str_if_empty:NT \g__nm_type_env_str
936     { \str_gset:Nn \g__nm_type_env_str { NiceArray } }

```

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

```

937     \NiceArrayWithDelims . .
938 }
939 { \endNiceArrayWithDelims }

```

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

```

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

948 \NewDocumentEnvironment { bNiceArray } { }
949 {
950     \str_if_empty:NT \g__nm_type_env_str
951     { \str_gset:Nn \g__nm_type_env_str { NiceArray } }
952     \__nm_test_if_math_mode:
953     \NiceArrayWithDelims [ ]
954 }
955 { \endNiceArrayWithDelims }

956 \NewDocumentEnvironment { BNiceArray } { }
957 {
958     \str_if_empty:NT \g__nm_type_env_str
959     { \str_gset:Nn \g__nm_type_env_str { BNiceArray } }
960     \__nm_test_if_math_mode:
961     \NiceArrayWithDelims \{ \}
962 }
963 { \endNiceArrayWithDelims }

964 \NewDocumentEnvironment { vNiceArray } { }
965 {
966     \str_if_empty:NT \g__nm_type_env_str
967     { \str_gset:Nn \g__nm_type_env_str { vNiceArray } }
968     \__nm_test_if_math_mode:
969     \NiceArrayWithDelims | |

```

```

970 }
971 { \endNiceArrayWithDelims }
972 \NewDocumentEnvironment { VNiceArray } { }
973 {
974   \str_if_empty:NT \g__nm_type_env_str
975     { \str_gset:Nn \g__nm_type_env_str { VNiceArray } }
976   \__nm_test_if_math_mode:
977   \NiceArrayWithDelims || ||
978 }
979 { \endNiceArrayWithDelims }

```

17.6 The environment `{NiceMatrix}` and its variants

```

980 \cs_new_protected:Npn \__nm_define_env:n #1
981 {
982   \NewDocumentEnvironment { #1 NiceMatrix } { ! 0 { } }
983   {
984     \str_gset:Nn \g__nm_type_env_str { #1 NiceMatrix }
985     \keys_set:nn { NiceMatrix / NiceMatrix } { ##1 }
986     \begin { #1 NiceArray }
987       {
988         *
989         {
990           \int_compare:nNnTF \l__nm_last_col_int = { -1 }
991             \c@MaxMatrixCols
992             { \int_eval:n { \l__nm_last_col_int - 1 } }
993         }
994         C
995       }
996     }
997     { \end { #1 NiceArray } }
998   }
999   \__nm_define_env:n { }
1000   \__nm_define_env:n p
1001   \__nm_define_env:n b
1002   \__nm_define_env:n B
1003   \__nm_define_env:n v
1004   \__nm_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.

```

1005 \prg_set_conditional:Npnn \__nm_if_not_empty_cell:nn #1 #2 { T , TF }
1006 {

```

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.

```

1007   \bool_set_false:N \l_tmpa_bool
1008   \cs_if_exist:cTF
1009     { __nm _ dotted _ \int_use:N #1 - \int_use:N #2 }
1010     \prg_return_true:
1011   {

```


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.

```

1012     \cs_if_free:cTF
1013     {
1014         pgf@sh@ns@nm -
1015         \int_use:N \g__nm_env_int -
1016         \int_use:N #1 -
1017         \int_use:N #2
1018     }
1019     { \prg_return_false: }
1020     {

```

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.

```

1021     \bool_set_false:N \l_tmpa_bool
1022     \cs_if_exist:cT
1023     { __nm _ empty _ \int_use:N #1 - \int_use:N #2 }
1024     {
1025         \int_compare:nNnT
1026         { \use:c { __nm _ empty _ \int_use:N #1 - \int_use:N #2 } }
1027         =
1028         \g__nm_env_int
1029         { \bool_set_true:N \l_tmpa_bool }
1030     }
1031     \bool_if:NTF \l_tmpa_bool
1032     \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.

```

1033     {
1034         \begin { pgfpicture }

```

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

```

1035     \tl_set:Nx \l_tmpa_tl
1036     {
1037         nm -
1038         \int_use:N \g__nm_env_int -
1039         \int_use:N #1 -
1040         \int_use:N #2
1041     }
1042     \pgfpointanchor \l_tmpa_tl { east }
1043     \dim_gset:Nn \g_tmpa_dim \pgf@x
1044     \pgfpointanchor \l_tmpa_tl { west }
1045     \dim_gset:Nn \g_tmpb_dim \pgf@x
1046     \end { pgfpicture }
1047     \dim_compare:nNnTF
1048     { \dim_abs:n { \g_tmpb_dim - \g_tmpa_dim } } < { 0.5 pt }
1049     \prg_return_false:
1050     \prg_return_true:
1051     }
1052 }
1053 }
1054 }

```

17.8 After the construction of the array

```

1055 \cs_new_protected:Nn \__nm_after_array:
1056 {
1057   \int_compare:nNnTF \c@iRow > \c_zero_int
1058     \__nm_after_array_i:
1059     {
1060       \__nm_error:n { Zero~row }
1061       \__nm_restore_iRow_jCol:
1062     }
1063 }

```

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

```

1064 \cs_new_protected:Nn \__nm_after_array_i:
1065 {
1066   \group_begin:
1067   \cs_if_exist:NT \tikz@library@external@loaded
1068     { \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

```

1069   \int_gset_eq:NN \c@jCol \g_nm_col_total_int
1070   \bool_if:nT \g_nm_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.

```

1071   \int_gset_eq:NN \g_nm_row_total_int \c@iRow
1072   \int_compare:nNnT \l_nm_last_row_int > { -1 }
1073     { \int_gsub:Nn \c@iRow \c_one_int }

```

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

```

1074   \bool_if:NT \l_nm_last_row_without_value_bool
1075     {
1076       \iow_now:Nn \@mainaux \ExplSyntaxOn
1077       \iow_now:Nx \@mainaux
1078       {
1079         \cs_gset:cpn { __nm_last_row_ \int_use:N \g_nm_env_int }
1080         { \int_use:N \g_nm_row_total_int }
1081       }

```

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.

```

1082       \str_if_empty:NF \l_nm_name_str
1083       {
1084         \iow_now:Nx \@mainaux
1085         {
1086           \cs_gset:cpn { __nm_last_row_ \l_nm_name_str }
1087           { \int_use:N \g_nm_row_total_int }
1088         }
1089       }
1090       \iow_now:Nn \@mainaux \ExplSyntaxOff
1091     }

```

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.

```

1092   \bool_if:NT \l_nm_parallelize_diags_bool
1093     {
1094       \int_zero_new:N \l_nm_ddots_int
1095       \int_zero_new:N \l_nm_iddots_int

```

The dimensions `\l_@@_delta_x_one_dim` and `\l_@@_delta_y_one_dim` will contain the Δ_x and Δ_y of the first `\Ddots` diagonal. We have to store these values in order to draw the others `\Ddots`

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

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

diagonals parallel to the first one. Similarly Δ_x and Δ_y of the first Δ diagonal.

```

1096 \dim_zero_new:N \l__nm_delta_x_one_dim
1097 \dim_zero_new:N \l__nm_delta_y_one_dim
1098 \dim_zero_new:N \l__nm_delta_x_two_dim
1099 \dim_zero_new:N \l__nm_delta_y_two_dim
1100 }

```

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).

```

1101 \bool_if:NT \g__nm_extra_nodes_bool \__nm_create_extra_nodes:
1102 \int_zero_new:N \l__nm_initial_i_int
1103 \int_zero_new:N \l__nm_initial_j_int
1104 \int_zero_new:N \l__nm_final_i_int
1105 \int_zero_new:N \l__nm_final_j_int
1106 \bool_set_false:N \l__nm_initial_open_bool
1107 \bool_set_false:N \l__nm_final_open_bool

```

If the option `small` is used, the values Δ_x and Δ_y (used to draw the dotted lines) are changed.

```

1108 \bool_if:NT \l__nm_small_bool
1109 {
1110   \dim_set:Nn \l__nm_radius_dim { 0.37 pt }
1111   \dim_set:Nn \l__nm_inter_dots_dim { 0.25 em }
1112 }

```

Now, we really draw the lines. The code to draw the lines has been constructed in the token lists Δ , etc.

```

1113 \g__nm_Hdotsfor_lines_tl
1114 \g__nm_Vdots_lines_tl
1115 \g__nm_Ddots_lines_tl
1116 \g__nm_Iddots_lines_tl
1117 \g__nm_Cdots_lines_tl
1118 \g__nm_Ldots_lines_tl

```

Now, the code-after.

```

1119 \tikzset
1120 {
1121   every-picture / .style =
1122   {
1123     overlay ,
1124     remember-picture ,
1125     name-prefix = nm - \int_use:N \g__nm_env_int -
1126   }
1127 }
1128 \cs_set_eq:NN \line \__nm_line:nn
1129 \g__nm_code_after_tl
1130 \tl_gclear:N \g__nm_code_after_tl
1131 \group_end:
1132 \str_gclear:N \g__nm_type_env_str
1133 \__nm_restore_iRow_jCol:
1134 }
1135 \cs_new_protected:Nn \__nm_restore_iRow_jCol:
1136 {
1137   \cs_if_exist:NT \theiRow { \int_gset_eq:NN \c@iRow \l__nm_save_iRow_int }
1138   \cs_if_exist:NT \thejCol { \int_gset_eq:NN \c@jCol \l__nm_save_jCol_int }
1139 }

```

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

its right.

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

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

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

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

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

```
1140 \cs_new_protected:Nn \__nm_find_extremities_of_line:nnnn
1141 {
```

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

```
1142 \cs_set:cpn { __nm _ dotted _ #1 - #2 } { }
```

Initialization of variables.

```
1143 \int_set:Nn \l__nm_initial_i_int { #1 }
1144 \int_set:Nn \l__nm_initial_j_int { #2 }
1145 \int_set:Nn \l__nm_final_i_int { #1 }
1146 \int_set:Nn \l__nm_final_j_int { #2 }
```

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

```
1147 \bool_set_false:N \l__nm_stop_loop_bool
1148 \bool_do_until:Nn \l__nm_stop_loop_bool
1149 {
1150   \int_add:Nn \l__nm_final_i_int { #3 }
1151   \int_add:Nn \l__nm_final_j_int { #4 }
```

We test if we are still in the matrix.

```
1152   \bool_set_false:N \l__nm_final_open_bool
1153   \int_compare:nNnTF \l__nm_final_i_int > \c@iRow
1154   {
1155     \int_compare:nNnT { #3 } = 1
1156     { \bool_set_true:N \l__nm_final_open_bool }
1157   }
1158   {
1159     \int_compare:nNnTF \l__nm_final_j_int < 1
1160     {
1161       \int_compare:nNnT { #4 } = { -1 }
1162       { \bool_set_true:N \l__nm_final_open_bool }
```

```

1163     }
1164     {
1165         \int_compare:nNtT \l__nm_final_j_int > \c@jCol
1166         {
1167             \int_compare:nNtT { #4 } = 1
1168             { \bool_set_true:N \l__nm_final_open_bool }
1169         }
1170     }
1171 }
1172 \bool_if:NTF \l__nm_final_open_bool

```

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

```

1173 {
1174     \int_sub:Nn \l__nm_final_i_int { #3 }
1175     \int_sub:Nn \l__nm_final_j_int { #4 }
1176     \bool_set_true:N \l__nm_stop_loop_bool
1177 }

```

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`.

```

1178 {
1179     \__nm_if_not_empty_cell:nnTF \l__nm_final_i_int \l__nm_final_j_int
1180     { \bool_set_true:N \l__nm_stop_loop_bool }

```

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

```

1181 {
1182     \cs_set:cpn
1183     {
1184         __nm _ dotted _
1185         \int_use:N \l__nm_final_i_int -
1186         \int_use:N \l__nm_final_j_int
1187     }
1188     { }
1189 }
1190 }
1191 }

```

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).

```

1192 \cs_if_free:cT
1193 {
1194     pgf@sh@ns@nm -
1195     \int_use:N \g__nm_env_int -
1196     \int_use:N \l__nm_final_i_int -
1197     \int_use:N \l__nm_final_j_int
1198 }
1199 {
1200     \bool_if:NF \l__nm_final_open_bool
1201     {
1202         \msg_error:nnx { nicematrix } { Impossible~line }
1203         { \int_use:N \l__nm_final_i_int }
1204         \bool_set_true:N \l__nm_impossible_line_bool
1205     }
1206 }

```

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

```

1207 \bool_set_false:N \l__nm_stop_loop_bool
1208 \bool_do_until:Nn \l__nm_stop_loop_bool
1209 {
1210   \int_sub:Nn \l__nm_initial_i_int { #3 }
1211   \int_sub:Nn \l__nm_initial_j_int { #4 }
1212   \bool_set_false:N \l__nm_initial_open_bool
1213   \int_compare:nNnTF \l__nm_initial_i_int < 1
1214   {
1215     \int_compare:nNnT { #3 } = 1
1216     { \bool_set_true:N \l__nm_initial_open_bool }
1217   }
1218   {
1219     \int_compare:nNnTF \l__nm_initial_j_int < 1
1220     {
1221       \int_compare:nNnT { #4 } = 1
1222       { \bool_set_true:N \l__nm_initial_open_bool }
1223     }
1224     {
1225       \int_compare:nNnT \l__nm_initial_j_int > \c@jCol
1226       {
1227         \int_compare:nNnT { #4 } = { -1 }
1228         { \bool_set_true:N \l__nm_initial_open_bool }
1229       }
1230     }
1231   }
1232   \bool_if:NTF \l__nm_initial_open_bool
1233   {
1234     \int_add:Nn \l__nm_initial_i_int { #3 }
1235     \int_add:Nn \l__nm_initial_j_int { #4 }
1236     \bool_set_true:N \l__nm_stop_loop_bool
1237   }
1238   {
1239     \__nm_if_not_empty_cell:nnTF
1240     \l__nm_initial_i_int \l__nm_initial_j_int
1241     { \bool_set_true:N \l__nm_stop_loop_bool }
1242     {
1243       \cs_set:cpn
1244       {
1245         __nm _ dotted _
1246         \int_use:N \l__nm_initial_i_int -
1247         \int_use:N \l__nm_initial_j_int
1248       }
1249       { }
1250     }
1251   }
1252 }

```

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).

```

1253 \cs_if_free:cT
1254 {
1255   pgf@sh@ns@nm -
1256   \int_use:N \g__nm_env_int -
1257   \int_use:N \l__nm_initial_i_int -
1258   \int_use:N \l__nm_initial_j_int
1259 }
1260 {
1261   \bool_if:NF \l__nm_initial_open_bool
1262   {
1263     \msg_error:nnx { nicematrix } { Impossible~line }
1264     { \int_use:N \l__nm_initial_i_int }
1265     \bool_set_true:N \l__nm_impossible_line_bool

```

```

1266     }
1267 }

```

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.

```

1268 \bool_if:nT \l__nm_initial_open_bool \__nm_create_extra_nodes:
1269 \bool_if:NT \l__nm_final_open_bool \__nm_create_extra_nodes:
1270 }

```

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

The two arguments of the command `\@@_retrieve_coords:nn` are the suffix and the anchor that must be used for the two nodes.

The coordinates are stored in `\g__@@_x_initial_dim`, `\g__@@_y_initial_dim`, `\g__@@_x_final_dim`, `\g__@@_y_final_dim`. These variables are global for technical reasons: we have to do an affectation in an environment `{tikzpicture}`.

```

1271 \cs_new_protected:Nn \__nm_retrieve_coords:nn
1272 {
1273   \dim_gzero_new:N \g__nm_x_initial_dim
1274   \dim_gzero_new:N \g__nm_y_initial_dim
1275   \dim_gzero_new:N \g__nm_x_final_dim
1276   \dim_gzero_new:N \g__nm_y_final_dim
1277   \begin { tikzpicture } [ remember-picture ]
1278     \tikz@parse@node \pgfutil@firstofone
1279     ( nm - \int_use:N \g__nm_env_int -
1280       \int_use:N \l__nm_initial_i_int -
1281       \int_use:N \l__nm_initial_j_int #1 )
1282     \dim_gset:Nn \g__nm_x_initial_dim \pgf@x
1283     \dim_gset:Nn \g__nm_y_initial_dim \pgf@y
1284     \tikz@parse@node \pgfutil@firstofone
1285     ( nm - \int_use:N \g__nm_env_int -
1286       \int_use:N \l__nm_final_i_int -
1287       \int_use:N \l__nm_final_j_int #2 )
1288     \dim_gset:Nn \g__nm_x_final_dim \pgf@x
1289     \dim_gset:Nn \g__nm_y_final_dim \pgf@y
1290   \end { tikzpicture }
1291 }
1292 \cs_generate_variant:Nn \__nm_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`).

```

1293 \cs_new_protected:Nn \__nm_adjust_with_col_nodes:
1294 {
1295   \bool_if:NT \l__nm_initial_open_bool
1296   {
1297     \cs_if_exist:cT
1298     { pgf@sh@ns@nm - \int_use:N \g__nm_env_int - col - 0 }
1299     {
1300       \begin { tikzpicture } [ remember-picture ]
1301         \tikz@parse@node \pgfutil@firstofone
1302         ( nm - \int_use:N \g__nm_env_int - col - 0 )
1303         \dim_gset:Nn \g__nm_x_initial_dim \pgf@x
1304       \end { tikzpicture }

```

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.

```

1305     }
1306   }
1307   \bool_if:NT \l__nm_final_open_bool
1308   {
1309     \cs_if_exist:cT
1310     {
1311       pgf@sh@ns@nm - \int_use:N \g__nm_env_int - col -
1312       \int_use:N \c@jCol
1313     }
1314     {
1315       \begin { tikzpicture } [ remember~picture ]
1316       \tikz@parse@node \pgfutil@firstofone
1317       (
1318         nm - \int_use:N \g__nm_env_int - col -
1319         \int_use:N \c@jCol
1320       )
1321       \dim_gset:Nn \g__nm_x_final_dim \pgf@x
1322       \end { tikzpicture }
1323     }
1324   }
1325 }

1326 \cs_new_protected:Nn \__nm_draw_Ldots:nn
1327 {
1328   \cs_if_free:cT { __nm _ dotted _ #1 - #2 }
1329   {
1330     \bool_set_false:N \l__nm_impossible_line_bool
1331     \__nm_find_extremities_of_line:nnnn { #1 } { #2 } \c_zero_int \c_one_int
1332     \bool_if:NF \l__nm_impossible_line_bool \__nm_actually_draw_Ldots:
1333   }
1334 }

```

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`.

```

1335 \cs_new_protected:Nn \__nm_actually_draw_Ldots:
1336 {
1337   \__nm_retrieve_coords:xx
1338   {
1339     \bool_if:NTF \l__nm_initial_open_bool
1340     {

```

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

```

1341     \cs_if_exist:cTF
1342     {
1343       pgf@sh@ns@nm
1344       - \int_use:N \g__nm_env_int
1345       - \int_use:N \l__nm_initial_i_int
1346       - \int_use:N \l__nm_initial_j_int - w
1347     }
1348     { - w.base~west }
1349     { - medium.base~west }
1350   }
1351   { .base~east }
1352 }
1353 {
1354   \bool_if:NTF \l__nm_final_open_bool
1355   {
1356     \cs_if_exist:cTF
1357     {
1358       pgf@sh@ns@nm
1359       - \int_use:N \g__nm_env_int

```



```

1360         - \int_use:N \l__nm_final_i_int
1361         - \int_use:N \l__nm_final_j_int - w
1362     }
1363     { - w.base~east }
1364     { - medium.base~east }
1365 }
1366 { .base~west }
1367 }
1368 \__nm_adjust_with_col_nodes:
1369 \bool_if:NT \l__nm_initial_open_bool
1370 { \dim_gset_eq:NN \g__nm_y_initial_dim \g__nm_y_final_dim }
1371 \bool_if:NT \l__nm_final_open_bool
1372 { \dim_gset_eq:NN \g__nm_y_final_dim \g__nm_y_initial_dim }

```

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

```

1373 \dim_gadd:Nn \g__nm_y_initial_dim { 0.53 pt }
1374 \dim_gadd:Nn \g__nm_y_final_dim { 0.53 pt }
1375 \__nm_draw_tikz_line:
1376 }

```

```

1377 \cs_new_protected:Nn \__nm_draw_Cdots:nn
1378 {
1379     \cs_if_free:cT { __nm _ dotted _ #1 - #2 }
1380     {
1381         \bool_set_false:N \l__nm_impossible_line_bool
1382         \__nm_find_extremities_of_line:nnnn { #1 } { #2 } \c_zero_int \c_one_int
1383         \bool_if:NF \l__nm_impossible_line_bool
1384         {
1385             \__nm_retrieve_coords:xx
1386             {
1387                 \bool_if:NTF \l__nm_initial_open_bool
1388                 {
1389                     \cs_if_exist:cTF
1390                     {
1391                         pgf@sh@ns@nm
1392                         - \int_use:N \g__nm_env_int
1393                         - \int_use:N \l__nm_initial_i_int
1394                         - \int_use:N \l__nm_initial_j_int - w
1395                     }
1396                     { - w.mid~west }
1397                     { - medium.mid~west }
1398                 }
1399                 { .mid~east }
1400             }
1401             {
1402                 \bool_if:NTF \l__nm_final_open_bool
1403                 {
1404                     \cs_if_exist:cTF
1405                     {
1406                         pgf@sh@ns@nm
1407                         - \int_use:N \g__nm_env_int
1408                         - \int_use:N \l__nm_final_i_int
1409                         - \int_use:N \l__nm_final_j_int - w
1410                     }
1411                     { - w.mid~east }
1412                     { - medium.mid~east }
1413                 }
1414                 { .mid~west }
1415             }
1416             \__nm_adjust_with_col_nodes:
1417             \bool_if:NT \l__nm_initial_open_bool
1418             { \dim_gset_eq:NN \g__nm_y_initial_dim \g__nm_y_final_dim }

```

```

1419         \bool_if:NT \l__nm_final_open_bool
1420         { \dim_gset_eq:NN \g__nm_y_final_dim \g__nm_y_initial_dim }
1421         \__nm_draw_tikz_line:
1422     }
1423 }
1424 }

```

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?).

```

1425 \cs_new_protected:Nn \__nm_draw_Vdots:nn
1426 {
1427     \cs_if_free:cT { __nm _ dotted _ #1 - #2 }
1428     {
1429         \bool_set_false:N \l__nm_impossible_line_bool
1430         \__nm_find_extremities_of_line:nnnn { #1 } { #2 } \c_one_int \c_zero_int
1431         \bool_if:NF \l__nm_impossible_line_bool
1432         {
1433             \__nm_retrieve_coords:xx
1434             {
1435                 \bool_if:NTF \l__nm_initial_open_bool
1436                 { - medium.north-west }
1437                 { .south-west }
1438             }
1439             {
1440                 \bool_if:NTF \l__nm_final_open_bool
1441                 { - medium.south-west }
1442                 { .north-west }
1443             }
1444         }
1445     }
1446 }

```

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

```

1444 \bool_set:Nn \l_tmpa_bool
1445 { \dim_compare_p:nNn \g__nm_x_initial_dim = \g__nm_x_final_dim }
1446 \__nm_retrieve_coords:xx
1447 {
1448     \bool_if:NTF \l__nm_initial_open_bool
1449     { - medium.north }
1450     { .south }
1451 }
1452 {
1453     \bool_if:NTF \l__nm_final_open_bool
1454     { - medium.south }
1455     { .north }
1456 }

```

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

```

1457 \bool_set:Nn \l_tmppb_bool
1458 { \dim_compare_p:nNn \g__nm_x_initial_dim = \g__nm_x_final_dim }
1459 \bool_if:NF \l_tmppb_bool
1460 {
1461     \dim_gset:Nn \g__nm_x_initial_dim
1462     {
1463         \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
1464         \g__nm_x_initial_dim \g__nm_x_final_dim
1465     }
1466     \dim_gset_eq:NN \g__nm_x_final_dim \g__nm_x_initial_dim
1467 }
1468 \__nm_draw_tikz_line:
1469 }
1470 }
1471 }

```

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.

```

1472 \cs_new_protected:Nn \__nm_draw_Ddots:nn
1473 {
1474   \cs_if_free:cT { __nm _ dotted _ #1 - #2 }
1475   {
1476     \bool_set_false:N \l__nm_impossible_line_bool
1477     \__nm_find_extremities_of_line:nnnn { #1 } { #2 } \c_one_int \c_one_int
1478     \bool_if:NF \l__nm_impossible_line_bool
1479     {
1480       \__nm_retrieve_coords:xx
1481       {
1482         \bool_if:NTF \l__nm_initial_open_bool
1483         { - medium.north-west }
1484         { .south-east }
1485       }
1486       {
1487         \bool_if:NTF \l__nm_final_open_bool
1488         { - medium.south-east }
1489         { .north-west }
1490       }
1491     }

```

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.

```

1491     \bool_if:NT \l__nm_parallelize_diags_bool
1492     {
1493       \int_incr:N \l__nm_ddots_int

```

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

```

1494       \int_compare:nNnTF \l__nm_ddots_int = \c_one_int

```

If the diagonal line is the first one, we have no adjustment of the line to do but we store the Δ_x and the Δ_y of the line because these values will be used to draw the others diagonal lines parallels to the first one.

```

1495       {
1496         \dim_set:Nn \l__nm_delta_x_one_dim
1497         { \g__nm_x_final_dim - \g__nm_x_initial_dim }
1498         \dim_set:Nn \l__nm_delta_y_one_dim
1499         { \g__nm_y_final_dim - \g__nm_y_initial_dim }
1500       }

```

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`.

```

1501       {
1502         \dim_gset:Nn \g__nm_y_final_dim
1503         {
1504           \g__nm_y_initial_dim +
1505           ( \g__nm_x_final_dim - \g__nm_x_initial_dim ) *
1506           \dim_ratio:nn \l__nm_delta_y_one_dim \l__nm_delta_x_one_dim
1507         }
1508       }
1509     }

```

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

```

1510     \__nm_draw_tikz_line:
1511   }
1512 }
1513 }

```

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

```

1514 \cs_new_protected:Nn \__nm_draw_Iddots:nn
1515 {
1516   \cs_if_free:cT { __nm _ dotted _ #1 - #2 }

```

```

1517 {
1518   \bool_set_false:N \l__nm_impossible_line_bool
1519   \__nm_find_extremities_of_line:nnnn { #1 } { #2 } 1 { -1 }
1520   \bool_if:NF \l__nm_impossible_line_bool
1521   {
1522     \__nm_retrieve_coords:xx
1523     {
1524       \bool_if:NTF \l__nm_initial_open_bool
1525       { - medium.north~east }
1526       { .south~west }
1527     }
1528     {
1529       \bool_if:NTF \l__nm_final_open_bool
1530       { - medium.south~west }
1531       { .north~east }
1532     }
1533     \bool_if:NT \l__nm_parallelize_diags_bool
1534     {
1535       \int_incr:N \l__nm_iddots_int
1536       \int_compare:nNnTF \l__nm_iddots_int = \c_one_int
1537       {
1538         \dim_set:Nn \l__nm_delta_x_two_dim
1539         { \g__nm_x_final_dim - \g__nm_x_initial_dim }
1540         \dim_set:Nn \l__nm_delta_y_two_dim
1541         { \g__nm_y_final_dim - \g__nm_y_initial_dim }
1542       }
1543       {
1544         \dim_gset:Nn \g__nm_y_final_dim
1545         {
1546           \g__nm_y_initial_dim +
1547           ( \g__nm_x_final_dim - \g__nm_x_initial_dim ) *
1548           \dim_ratio:nn \l__nm_delta_y_two_dim \l__nm_delta_x_two_dim
1549         }
1550       }
1551     }
1552     \__nm_draw_tikz_line:
1553   }
1554 }
1555 }

```

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`.

```

1556 \cs_new_protected:Nn \__nm_draw_tikz_line:
1557 {

```

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.

```

1558   \dim_zero_new:N \l__nm_l_dim
1559   \dim_set:Nn \l__nm_l_dim
1560   {
1561     \fp_to_dim:n
1562     {
1563       sqrt
1564       (
1565         ( \dim_use:N \g__nm_x_final_dim
1566         - \dim_use:N \g__nm_x_initial_dim
1567         ) ^ 2
1568         +

```

```

1569         ( \dim_use:N \g__nm_y_final_dim
1570         - \dim_use:N \g__nm_y_initial_dim
1571         ) ^ 2
1572     )
1573 }
1574 }

```

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

```

1575 \dim_compare:nNnF \l__nm_l_dim = \c_zero_dim

```

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

```

1576 {
1577     \bool_if:NTF \l__nm_initial_open_bool
1578     {
1579         \bool_if:NTF \l__nm_final_open_bool
1580         {
1581             \int_set:Nn \l_tmpa_int
1582             { \dim_ratio:nn \l__nm_l_dim \l__nm_inter_dots_dim }
1583         }
1584         {
1585             \int_set:Nn \l_tmpa_int
1586             { \dim_ratio:nn { \l__nm_l_dim - 0.3 em } \l__nm_inter_dots_dim }
1587         }
1588     }
1589     {
1590         \bool_if:NTF \l__nm_final_open_bool
1591         {
1592             \int_set:Nn \l_tmpa_int
1593             { \dim_ratio:nn { \l__nm_l_dim - 0.3 em } \l__nm_inter_dots_dim }
1594         }
1595         {
1596             \int_set:Nn \l_tmpa_int
1597             { \dim_ratio:nn { \l__nm_l_dim - 0.6 em } \l__nm_inter_dots_dim }
1598         }
1599     }

```

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

```

1600 \dim_set:Nn \l_tmpa_dim
1601 {
1602     ( \g__nm_x_final_dim - \g__nm_x_initial_dim ) *
1603     \dim_ratio:nn \l__nm_inter_dots_dim \l__nm_l_dim
1604 }
1605 \dim_set:Nn \l_tmpb_dim
1606 {
1607     ( \g__nm_y_final_dim - \g__nm_y_initial_dim ) *
1608     \dim_ratio:nn \l__nm_inter_dots_dim \l__nm_l_dim
1609 }

```

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`.

```

1610 \int_set:Nn \l_tmpb_int
1611 {
1612     \bool_if:NTF \l__nm_initial_open_bool
1613     { \bool_if:NTF \l__nm_final_open_bool 1 0 }
1614     { \bool_if:NTF \l__nm_final_open_bool 2 1 }
1615 }

```

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.

```

1616 \dim_gadd:Nn \g__nm_x_initial_dim

```

```

1617     {
1618         ( \g__nm_x_final_dim - \g__nm_x_initial_dim ) *
1619         \dim_ratio:nn
1620         { \l__nm_l_dim - \l__nm_inter_dots_dim * \l_tmpa_int }
1621         { \l__nm_l_dim * 2 }
1622         * \l_tmpb_int
1623     }

```

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

```

1624     \dim_gadd:Nn \g__nm_y_initial_dim
1625     {
1626         ( \g__nm_y_final_dim - \g__nm_y_initial_dim ) *
1627         \dim_ratio:nn
1628         { \l__nm_l_dim - \l__nm_inter_dots_dim * \l_tmpa_int }
1629         { \l__nm_l_dim * 2 } *
1630         \l_tmpb_int
1631     }
1632     \begin { tikzpicture } [ overlay ]
1633     \int_step_inline:nnn 0 \l_tmpa_int
1634     {
1635         \pgfpathcircle
1636         { \pgfpoint { \g__nm_x_initial_dim } { \g__nm_y_initial_dim } }
1637         { \l__nm_radius_dim }
1638         \pgfusepath { fill }
1639         \dim_gadd:Nn \g__nm_x_initial_dim \l_tmpa_dim
1640         \dim_gadd:Nn \g__nm_y_initial_dim \l_tmpb_dim
1641     }
1642     \end { tikzpicture }
1643 }
1644 }

```

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).

```

1645 \cs_set_eq:NN \__nm_ldots \ldots
1646 \cs_set_eq:NN \__nm_cdots \cdots
1647 \cs_set_eq:NN \__nm_vdots \vdots
1648 \cs_set_eq:NN \__nm_ddots \ddots
1649 \cs_set_eq:NN \__nm_iddots \iddots

```

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

```

1650 \cs_new_protected:Nn \__nm_add_to_empty_cells:
1651 {
1652     \cs_gset:cpx
1653     { __nm _ empty _ \int_use:N \c@iRow - \int_use:N \c@jCol }
1654     { \int_use:N \g__nm_env_int }
1655 }

```

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.

```

1656 \NewDocumentCommand \__nm_Ldots { s }
1657 {
1658     \bool_if:nF { #1 } { \__nm_instruction_of_type:n { Ldots } }
1659     \bool_if:NF \l__nm_nullify_dots_bool { \phantom \__nm_ldots }
1660     \__nm_add_to_empty_cells:
1661 }

```

```

1662 \NewDocumentCommand \__nm_Cdots { s }
1663 {
1664   \bool_if:nF { #1 } { \__nm_instruction_of_type:n { Cdots } }
1665   \bool_if:NF \l__nm_nullify_dots_bool { \phantom \__nm_cdots }
1666   \__nm_add_to_empty_cells:
1667 }

1668 \NewDocumentCommand \__nm_Vdots { s }
1669 {
1670   \bool_if:nF { #1 } { \__nm_instruction_of_type:n { Vdots } }
1671   \bool_if:NF \l__nm_nullify_dots_bool { \phantom \__nm_vdots }
1672   \__nm_add_to_empty_cells:
1673 }

1674 \NewDocumentCommand \__nm_Ddots { s }
1675 {
1676   \bool_if:nF { #1 } { \__nm_instruction_of_type:n { Ddots } }
1677   \bool_if:NF \l__nm_nullify_dots_bool { \phantom \__nm_ddots }
1678   \__nm_add_to_empty_cells:
1679 }

1680 \NewDocumentCommand \__nm_Iddots { s }
1681 {
1682   \bool_if:nF { #1 } { \__nm_instruction_of_type:n { Iddots } }
1683   \bool_if:NF \l__nm_nullify_dots_bool { \phantom \__nm_iddots }
1684   \__nm_add_to_empty_cells:
1685 }

```

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

```

1686 \cs_new_protected:Nn \__nm_Hspace:
1687 {
1688   \__nm_add_to_empty_cells:
1689   \hspace
1690 }

```

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

```

1691 \cs_set_eq:NN \__nm_old_multicolumn \multicolumn
1692 \cs_new:Npn \__nm_multicolumn:nnn #1 #2 #3
1693 {
1694   \__nm_old_multicolumn { #1 } { #2 } { #3 }
1695   \int_compare:nNnT #1 > 1
1696   {
1697     \seq_gput_left:Nx \g__nm_multicolumn_cells_seq
1698     { \int_eval:n \c@iRow - \int_use:N \c@jCol }
1699     \seq_gput_left:Nn \g__nm_multicolumn_sizes_seq { #1 }
1700   }
1701   \int_gadd:Nn \c@jCol { #1 - 1 }
1702 }

```

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`.

```

1703 \cs_new:Npn \__nm_Hdotsfor:
1704 {
1705   \multicolumn { 1 } { C } { }
1706   \__nm_Hdotsfor_i
1707 }

```

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:`).

```

1708 \bool_if:NTF \c_nm_draft_bool
1709 {
1710   \NewDocumentCommand \_nm_Hdotsfor_i { 0 { } m }
1711     { \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { C } { } } }
1712 }
1713 {
1714   \NewDocumentCommand \_nm_Hdotsfor_i { 0 { } m }
1715     {
1716       \tl_gput_right:Nx \g__nm_Hdotsfor_lines_tl
1717       {
1718         \_nm_draw_Hdotsfor:nnn
1719         { \int_use:N \c@iRow }
1720         { \int_use:N \c@jCol }
1721         { #2 }
1722       }
1723       \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { C } { } }
1724     }
1725 }

1726 \cs_new_protected:Nn \_nm_draw_Hdotsfor:nnn
1727 {
1728   \bool_set_false:N \l_nm_initial_open_bool
1729   \bool_set_false:N \l_nm_final_open_bool

```

For the row, it's easy.

```

1730 \int_set:Nn \l_nm_initial_i_int { #1 }
1731 \int_set:Nn \l_nm_final_i_int { #1 }

```

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

```

1732 \int_compare:nNnTF #2 = 1
1733 {
1734   \int_set:Nn \l_nm_initial_j_int 1
1735   \bool_set_true:N \l_nm_initial_open_bool
1736 }
1737 {
1738   \int_set:Nn \l_tmpa_int { #2 - 1 }
1739   \_nm_if_not_empty_cell:nnTF \l_nm_initial_i_int \l_tmpa_int
1740   { \int_set:Nn \l_nm_initial_j_int { #2 - 1 } }
1741   {
1742     \int_set:Nn \l_nm_initial_j_int {#2}
1743     \bool_set_true:N \l_nm_initial_open_bool
1744   }
1745 }
1746 \int_compare:nNnTF { #2 + #3 - 1 } = \c@jCol
1747 {
1748   \int_set:Nn \l_nm_final_j_int { #2 + #3 - 1 }
1749   \bool_set_true:N \l_nm_final_open_bool
1750 }
1751 {
1752   \int_set:Nn \l_tmpa_int { #2 + #3 }
1753   \_nm_if_not_empty_cell:nnTF \l_nm_final_i_int \l_tmpa_int
1754   { \int_set:Nn \l_nm_final_j_int { #2 + #3 } }
1755   {
1756     \int_set:Nn \l_nm_final_j_int { #2 + #3 - 1 }
1757     \bool_set_true:N \l_nm_final_open_bool
1758   }
1759 }
1760 \bool_if:nT { \l_nm_initial_open_bool || \l_nm_final_open_bool }
1761   \_nm_create_extra_nodes:
1762   \_nm_actually_draw_Ldots:

```


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

```

1763 \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
1764 { \cs_set:cpn { __nm _ dotted _ #1 - ##1 } { } }
1765 }

```

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.

```

1766 \cs_new_protected:Nn \__nm_line:nn
1767 {
1768   \bool_if:NF \c__nm_draft_bool
1769   {
1770     \dim_zero_new:N \g__nm_x_initial_dim
1771     \dim_zero_new:N \g__nm_y_initial_dim
1772     \dim_zero_new:N \g__nm_x_final_dim
1773     \dim_zero_new:N \g__nm_y_final_dim
1774     \bool_set_false:N \l__nm_initial_open_bool
1775     \bool_set_false:N \l__nm_final_open_bool
1776     \bool_if:nTF
1777     {
1778       \cs_if_exist_p:c { pgf@sh@ns@nm - \int_use:N \g__nm_env_int - #1 }
1779       &&
1780       \cs_if_exist_p:c { pgf@sh@ns@nm - \int_use:N \g__nm_env_int - #2 }
1781     }
1782     {
1783       \begin { tikzpicture }
1784         \path~( #1 )~---~( #2 )~node[at~start]~( i )~{}~node[at~end]~( f )~{} ;
1785         \tikz@parse@node \pgfutil@firstofone ( i )
1786         \dim_gset:Nn \g__nm_x_initial_dim \pgf@x
1787         \dim_gset:Nn \g__nm_y_initial_dim \pgf@y
1788         \tikz@parse@node \pgfutil@firstofone ( f )
1789         \dim_gset:Nn \g__nm_x_final_dim \pgf@x
1790         \dim_gset:Nn \g__nm_y_final_dim \pgf@y
1791       \end { tikzpicture }
1792       \__nm_draw_tikz_line:
1793     }
1794     {
1795       \__nm_error:nnn { unknown~cell~for~line~in~code~after }
1796       { #1 } { #2 }
1797     }
1798   }
1799 }

```

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

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

```
1800 \bool_if:NTF \c_nm_draft_bool
1801   { \cs_set_eq:NN \_nm_dotfill: \prg_do_nothing: }
1802   {
1803     \cs_set:Npn \_nm_dotfill:
1804     {
```

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.

```
1805     \bool_if:NT \l_nm_small_bool
1806     { \dim_set:Nn \l_nm_inter_dots_dim { 0.25 em } }
1807     \cleaders
1808     \hbox_to_wd:nn
1809     { \l_nm_inter_dots_dim }
1810     {
1811       \c_math_toggle_token
1812       \bool_if:NT \l_nm_small_bool \scriptstyle
1813       \hss . \hss
1814       \c_math_toggle_token
1815     }
1816     \hfill
1817   }
1818 }
```

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

```
1819 \cs_new:Npn \_nm_hdottedline:
1820 {
1821   \noalign
1822   {
1823     \bool_gset_true:N \g_nm_extra_nodes_bool
1824     \cs_if_exist:cTF { \_nm_width_ \int_use:N \g_nm_env_int }
1825     { \dim_set_eq:Nc \l_tmpa_dim { \_nm_width_ \int_use:N \g_nm_env_int } }
1826     { \dim_set:Nn \l_tmpa_dim { 5 mm } }
1827     \hbox_overlap_right:n
1828     {
1829       \bool_if:nT
1830       {
1831         \l_nm_NiceArray_bool
1832         &&
1833         ! \l_nm_exterior_arraycolsep_bool
1834         &&
1835         \int_compare_p:nNn \l_nm_first_col_int > \c_zero_int
1836       }
1837       { \skip_horizontal:n { - \arraycolsep } }
1838       \hbox_to_wd:nn
1839       {
1840         \l_tmpa_dim + 2 \arraycolsep
1841         - \l_nm_left_margin_dim - \l_nm_right_margin_dim
1842       }
1843       \_nm_dotfill:
1844     }
1845   }
1846 }
```

```
1847 \cs_new_protected:Nn \_nm_vdottedline:n
1848 {
```

³¹In fact, almost the same position because of the width of the line: the width of a dotted line is not the same as the width of a line created by `\hline`.

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

```

1849 \int_compare:nNnTF #1 = \c_zero_int
1850 { \__nm_error:n { Use-of~:-in-first-position } }
1851 {
1852   \bool_if:NF \c__nm_draft_bool
1853   {
1854     \dim_zero_new:N \g__nm_x_initial_dim
1855     \dim_zero_new:N \g__nm_y_initial_dim
1856     \dim_zero_new:N \g__nm_x_final_dim
1857     \dim_zero_new:N \g__nm_y_final_dim
1858     \bool_set_true:N \l__nm_initial_open_bool
1859     \bool_set_true:N \l__nm_final_open_bool

```

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

```

1860 \cs_if_exist:cTF
1861 { pgf@sh@ns@nm -\int_use:N \g__nm_env_int - col - #1 }
1862 {
1863   \begin { tikzpicture } [ remember-picture ]
1864     \tikz@parse@node\pgfutil@firstofone
1865     ( col - #1 )
1866     \dim_gset:Nn \g__nm_x_initial_dim \pgf@x
1867     \dim_gset:Nn \g__nm_x_final_dim \pgf@x
1868     \dim_gset:Nn \g__nm_y_final_dim \pgf@y
1869   \end { tikzpicture }
1870   \dim_gset:Nn \g__nm_y_initial_dim { - \c_max_dim }
1871   \int_step_inline:nn \c@jCol
1872   {
1873     \begin { tikzpicture } [ remember-picture ]
1874       \tikz@parse@node\pgfutil@firstofone
1875       ( 1 - ##1 . north-east )
1876       \dim_gset:Nn \g__nm_y_initial_dim
1877       { \dim_max:nn \g__nm_y_initial_dim \pgf@y }
1878     \end { tikzpicture }
1879   }
1880 }

```

If not, we use the “large node”.

```

1881 {
1882   \begin { tikzpicture } [ remember-picture ]
1883     \tikz@parse@node\pgfutil@firstofone
1884     ( 1 - #1 - large .north-east )
1885     \dim_gset:Nn \g__nm_x_initial_dim \pgf@x
1886     \dim_gset:Nn \g__nm_y_initial_dim \pgf@y
1887     \tikz@parse@node\pgfutil@firstofone
1888     ( \int_use:N \c@iRow - #1 - large .south-east )
1889     \dim_gset:Nn \g__nm_x_final_dim \pgf@x
1890     \dim_gset:Nn \g__nm_y_final_dim \pgf@y
1891   \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).

```

1892 \cs_if_exist:cTF
1893 { pgf@sh@ns@nm -\int_use:N \g__nm_env_int - 1 - #1 - w }
1894 {
1895   \begin { tikzpicture } [ remember-picture ]
1896     \tikz@parse@node\pgfutil@firstofone
1897     ( 1 - #1 - w .north-east )
1898     \dim_gset:Nn \g__nm_x_initial_dim \pgf@x
1899     \tikz@parse@node\pgfutil@firstofone
1900     ( \int_use:N \c@iRow - #1 - w .south-east )
1901     \dim_gset:Nn \g__nm_x_final_dim \pgf@x
1902   \end { tikzpicture }
1903   \dim_gadd:Nn \g__nm_x_initial_dim \arraycolsep

```

```

1904         \dim_gadd:Nn \g__nm_x_final_dim \arraycolsep
1905     }
1906 }
1907 \__nm_draw_tikz_line:
1908 }
1909 }
1910 }

```

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 `\newcolumnntype`:

```

\newcolumnntype { | }
{ ! { \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).

```

1911 \cs_new_protected:Npn \__nm_vline:
1912 {
1913     \int_compare:nNnTF \l__nm_first_col_int = \c_zero_int
1914     {
1915         \int_compare:nNnTF \c@jCol = \c_zero_int
1916         {
1917             \int_compare:nNnTF \l__nm_first_row_int = \c_zero_int
1918             {
1919                 \int_compare:nNnF \c@iRow = \c_zero_int
1920                 {
1921                     \int_compare:nNnF \c@iRow = \l__nm_last_row_int
1922                     \__nm_vline_i:
1923                 }
1924             }
1925             {
1926                 \int_compare:nNnF \c@iRow = \c_zero_int
1927                 {
1928                     \int_compare:nNnF \c@iRow = \l__nm_last_row_int
1929                     \__nm_vline_i:
1930                 }
1931             }
1932         }
1933         {
1934             \int_compare:nNnF \c@iRow = \c_zero_int
1935             {
1936                 \int_compare:nNnF \c@iRow = \l__nm_last_row_int
1937                 \__nm_vline_i:
1938             }
1939         }
1940     }
1941     {
1942         \int_compare:nNnTF \c@jCol = \c_zero_int
1943         {
1944             \int_compare:nNnF \c@iRow = { -1 }
1945             {
1946                 \int_compare:nNnF \c@iRow = { \l__nm_last_row_int - 1 }
1947                 \__nm_vline_i:

```

```

1948         }
1949     }
1950     {
1951         \int_compare:nNf \c@iRow = \c_zero_int
1952         {
1953             \int_compare:nNf \c@iRow = \l__nm_last_row_int
1954             \__nm_vline_i:
1955         }
1956     }
1957 }
1958 }

```

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

```

1959 \cs_set_eq:NN \__nm_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.

```

1960 \bool_new:N \l__nm_block_auto_columns_width_bool

```

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

```

1961 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
1962 {
1963     auto-columns-width .code:n =
1964     {
1965         \bool_set_true:N \l__nm_block_auto_columns_width_bool
1966         \dim_gzero_new:N \g__nm_max_cell_width_dim
1967         \bool_set_true:N \l__nm_auto_columns_width_bool
1968     }
1969 }

1970 \NewDocumentEnvironment { NiceMatrixBlock } { ! 0 { } }
1971 {
1972     \int_gincr:N \g__nm_NiceMatrixBlock_int
1973     \dim_zero:N \l__nm_columns_width_dim
1974     \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
1975     \bool_if:NT \l__nm_block_auto_columns_width_bool
1976     {
1977         \cs_if_exist:cT { __nm_max_cell_width_ \int_use:N \g__nm_NiceMatrixBlock_int }
1978         {
1979             \dim_set:Nx \l__nm_columns_width_dim
1980             { \use:c { __nm_max_cell_width_ \int_use:N \g__nm_NiceMatrixBlock_int } }
1981         }
1982     }
1983 }

```

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`).

```

1984 {
1985     \bool_if:NT \l__nm_block_auto_columns_width_bool
1986     {
1987         \iow_now:Nn \@mainaux \ExplSyntaxOn
1988         \iow_now:Nx \@mainaux
1989         {
1990             \cs_gset:cpn
1991             { __nm_max_cell_width_ \int_use:N \g__nm_NiceMatrixBlock_int }

```

```

1992         { \dim_use:N \g__nm_max_cell_width_dim }
1993     }
1994     \iow_now:Nn \@mainaux \ExplSyntaxOff
1995 }
1996 }

```

17.15 The extra nodes

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

```

1997 \cs_generate_variant:Nn \dim_min:nn { v n }
1998 \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`.

```

1999 \cs_new_protected:Nn \__nm_create_extra_nodes:
2000 {
2001     \begin { tikzpicture } [ remember-picture , overlay ]
2002     \int_step_variable:nnNn \l__nm_first_row_int \g__nm_row_total_int \__nm_i:
2003     {
2004         \dim_zero_new:c { l__nm_row\___nm_i: _min_dim }
2005         \dim_set_eq:cN { l__nm_row\___nm_i: _min_dim } \c_max_dim
2006         \dim_zero_new:c { l__nm_row\___nm_i: _max_dim }
2007         \dim_set:cn { l__nm_row\___nm_i: _max_dim } { - \c_max_dim }
2008     }
2009     \int_step_variable:nnNn \l__nm_first_col_int \g__nm_col_total_int \__nm_j:
2010     {
2011         \dim_zero_new:c { l__nm_column\___nm_j: _min_dim }
2012         \dim_set_eq:cN { l__nm_column\___nm_j: _min_dim } \c_max_dim
2013         \dim_zero_new:c { l__nm_column\___nm_j: _max_dim }
2014         \dim_set:cn { l__nm_column\___nm_j: _max_dim } { - \c_max_dim }
2015     }

```

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

```

2016     \int_step_variable:nnNn \l__nm_first_row_int \g__nm_row_total_int \__nm_i:
2017     {
2018         \int_step_variable:nnNn
2019         \l__nm_first_col_int \g__nm_col_total_int \__nm_j:

```

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

```

2020         { \cs_if_exist:cT
2021             { pgf@sh@ns@nm - \int_use:N \g__nm_env_int - \__nm_i: - \__nm_j: }

```

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

```

2022         {
2023             \tikz@parse@node \pgfutil@firstofone
2024             ( nm - \int_use:N \g__nm_env_int
2025               - \__nm_i: - \__nm_j: .south-west )
2026             \dim_set:cn { l__nm_row\___nm_i: _min_dim }
2027             { \dim_min:vn { l__nm_row\___nm_i: _min_dim } \pgf@y }
2028             \seq_if_in:NxF \g__nm_multicolumn_cells_seq { \__nm_i: - \__nm_j: }

```

```

2029         {
2030             \dim_set:cn { l__nm_column _ \__nm_j: _min_dim}
2031             { \dim_min:vn { l__nm_column _ \__nm_j: _min_dim } \pgf@x }
2032         }

```

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`.

```

2033         \tikz@parse@node \pgfutil@firstofone
2034         ( nm - \int_use:N \g__nm_env_int - \__nm_i: - \__nm_j: .north-east )
2035         \dim_set:cn { l__nm_row _ \__nm_i: _ max_dim }
2036         { \dim_max:vn { l__nm_row _ \__nm_i: _ max_dim } \pgf@y }
2037         \seq_if_in:NxF \g__nm_multicolumn_cells_seq { \__nm_i: - \__nm_j: }
2038         {
2039             \dim_set:cn { l__nm_column _ \__nm_j: _ max_dim }
2040             { \dim_max:vn { l__nm_column _ \__nm_j: _ max_dim } \pgf@x }
2041         }
2042     }
2043 }
2044 }

```

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`).

```

2045     \tikzset { name-suffix = -medium }
2046     \__nm_create_nodes:

```

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

```

2047     \int_set:Nn \l__nm_first_row_int 1
2048     \int_set:Nn \l__nm_first_col_int 1

```

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

```

2049     \int_step_variable:nNn { \c@iRow - 1 } \__nm_i:
2050     {
2051         \dim_set:cn { l__nm_row _ \__nm_i: _ min _ dim }
2052         {
2053             (
2054                 \dim_use:c { l__nm_row _ \__nm_i: _ min _ dim } +
2055                 \dim_use:c { l__nm_row _ \int_eval:n { \__nm_i: + 1 } _ max _ dim }
2056             )
2057             / 2
2058         }
2059         \dim_set_eq:cc { l__nm_row _ \int_eval:n { \__nm_i: + 1 } _ max _ dim }
2060         { l__nm_row_\__nm_i: _ min_dim }
2061     }
2062     \int_step_variable:nNn { \c@jCol - 1 } \__nm_j:
2063     {
2064         \dim_set:cn { l__nm_column _ \__nm_j: _ max _ dim }
2065         {
2066             (
2067                 \dim_use:c
2068                 { l__nm_column _ \__nm_j: _ max _ dim } +
2069                 \dim_use:c
2070                 { l__nm_column _ \int_eval:n { \__nm_j: + 1 } _ min _ dim }
2071             )
2072             / 2
2073         }
2074         \dim_set_eq:cc { l__nm_column _ \int_eval:n { \__nm_j: + 1 } _ min _ dim }
2075         { l__nm_column _ \__nm_j: _ max _ dim }
2076     }
2077     \dim_sub:cn
2078     { l__nm_column _ 1 _ min _ dim }
2079     \l__nm_left_margin_dim

```

```

2080 \dim_add:cn
2081 { l__nm_column_ \int_use:N \c@jCol _ max _ dim }
2082 \l__nm_right_margin_dim

```

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

```

2083 \tikzset { name~suffix = -large }
2084 \__nm_create_nodes:
2085 \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.

```

2086 \cs_set:Npn \__nm_create_extra_nodes: { }

```

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

```

2087 \begin { tikzpicture } [ remember~picture , overlay ]
2088 \tikz@parse@node \pgfutil@firstofone
2089 ( nm - \int_use:N \g__nm_env_int - 1 - 1 - large .north-west )
2090 \dim_gset:Nn \g_tmpa_dim \pgf@x
2091 \tikz@parse@node \pgfutil@firstofone
2092 ( nm - \int_use:N \g__nm_env_int - 1 -
2093 \int_use:N \c@jCol - large .north-east )
2094 \dim_gset:Nn \g_tmpb_dim \pgf@x
2095 \end { tikzpicture }
2096 \iow_now:Nn \@mainaux \ExplSyntaxOn
2097 \iow_now:Nx \@mainaux
2098 {
2099 \cs_gset:cpn { __nm_width_ \int_use:N \g__nm_env_int }
2100 { \dim_eval:n { \g_tmpb_dim - \g_tmpa_dim } }
2101 }
2102 \iow_now:Nn \@mainaux \ExplSyntaxOff
2103 }

```

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.

```

2104 \cs_new_protected:Nn \__nm_create_nodes:
2105 {
2106 \int_step_variable:nnNn \l__nm_first_row_int \g__nm_row_total_int \__nm_i:
2107 {
2108 \int_step_variable:nnNn \l__nm_first_col_int \g__nm_col_total_int \__nm_j:

```

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

```

2109 {
2110 \coordinate ( __nm~south~west )
2111 at ( \dim_use:c { l__nm_column_ \__nm_j: _min_dim } ,
2112 \dim_use:c { l__nm_row_ \__nm_i: _min_dim } ) ;
2113 \coordinate ( __nm~north~east )
2114 at ( \dim_use:c { l__nm_column_ \__nm_j: _max_dim } ,
2115 \dim_use:c { l__nm_row_ \__nm_i: _max_dim } ) ;

```

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

```

2116 \node
2117 [
2118 node~contents = { } ,
2119 fit = ( __nm~south~west ) ( __nm~north~east ) ,
2120 inner~sep = \c_zero_dim ,

```



```

2121         name = nm - \int_use:N \g__nm_env_int - \__nm_i: - \__nm_j: ,
2122         alias =
2123             \str_if_empty:NF \l__nm_name_str
2124             { \l__nm_name_str - \__nm_i: - \__nm_j: }
2125     ]
2126     ;
2127 }
2128 }

```

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 .

```

2129     \seq_mapthread_function:NNN
2130     \g__nm_multicolumn_cells_seq
2131     \g__nm_multicolumn_sizes_seq
2132     \__nm_node_for_multicolumn:nn
2133 }
2134 \cs_new_protected:Npn \__nm_extract_coords: #1 - #2 \q_stop
2135 {
2136     \cs_set:Npn \__nm_i: { #1 }
2137     \cs_set:Npn \__nm_j: { #2 }
2138 }

```

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”).

```

2139 \cs_new_protected:Nn \__nm_node_for_multicolumn:nn
2140 {
2141     \__nm_extract_coords: #1 \q_stop
2142     \coordinate ( __nm~south-west ) at
2143     (
2144         \dim_use:c { l__nm_column _ \__nm_j: _ min _ dim } ,
2145         \dim_use:c { l__nm_row _ \__nm_i: _ min _ dim }
2146     ) ;
2147     \coordinate ( __nm~north-east ) at
2148     (
2149         \dim_use:c { l__nm_column _ \int_eval:n { \__nm_j: + #2 - 1 } _ max _ dim } ,
2150         \dim_use:c { l__nm_row _ \__nm_i: _ max _ dim }
2151     ) ;
2152     \node
2153     [
2154         node~contents = { } ,
2155         fit = ( __nm~south-west ) ( __nm~north-east ) ,
2156         inner~sep = \c_zero_dim ,
2157         name = nm - \int_use:N \g__nm_env_int - \__nm_i: - \__nm_j: ,
2158         alias =
2159             \str_if_empty:NF \l__nm_name_str
2160             { \l__nm_name_str - \__nm_i: - \__nm_j: }
2161     ]
2162     ;
2163 }

```

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 `\NewExpandableDocumentCommand` of `xparse` because it has an optionnal argument between `<` and `>` (for TeX instructions put before the math mode of the label) and because it must be expandable since it reduces (in the case of a block of only one row) to a command `\multicolumn`.

```

2164 \NewExpandableDocumentCommand \__nm_Block: { m D < > { } m }
2165 {

```

```

2166   \_nm_Block_i #1 \q_stop { #2 } { #3 }
2167 }

```

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.

```

2168 \cs_new:Npn \_nm_Block_i #1-#2 \q_stop
2169 {
2170   \_nm_Block_ii:nnnn { #1 } { #2 }
2171 }

```

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).

```

2172 \cs_new:Npn \_nm_Block_ii:nnnn #1 #2 #3 #4
2173 {

```

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

```

2174   \int_compare:nNnTF { #1 } = 1
2175   {
2176     \multicolumn { #2 } { C } { \hbox:n { #3 $#4$ } }
2177     \_nm_gobble_ampersands:n { #2 - 1 }
2178   }
2179   { \_nm_Block_iii:nnnn { #1 } { #2 } { #3 } { #4 } }
2180 }

```

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

```

2181 \cs_new_protected:Npn \_nm_Block_iii:nnnn #1 #2 #3 #4
2182 {
2183   \bool_gset_true:N \g__nm_extra_nodes_bool

```

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

```

2184   \tl_gput_left:Nx \g__nm_code_after_tl
2185   {
2186     \_nm_Block_iv:nnnnn
2187     { \int_use:N \c@iRow }
2188     { \int_use:N \c@jCol }
2189     { \int_eval:n { \c@iRow + #1 - 1 } }
2190     { \int_eval:n { \c@jCol + #2 - 1 } }
2191     \exp_not:n { { #3 $ #4 $ } }
2192   }
2193 }

```

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

```

2194 \group_begin:
2195   \char_set_catcode_letter:N \&
2196   \cs_new:Npn \_nm_gobble_ampersands:n #1
2197   {
2198     \int_compare:nNnT { #1 } > 0
2199     {
2200       \peek_charcode_remove_ignore_spaces:NT &
2201       { \_nm_gobble_ampersands:n { #1 - 1 } }
2202     }
2203   }
2204 \group_end:

```

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

```

2205 \cs_new_protected:Npn \__nm_Block_iv:nnnnn #1 #2 #3 #4 #5
2206 {
2207   \bool_if:nTF
2208     {
2209       \int_compare_p:nNn { #3 } > \c@iRow
2210       || \int_compare_p:nNn { #4 } > \c@jCol
2211     }
2212     { \msg_error:nnnn { nicematrix } { Block-too-large } { #1 } { #2 } }
2213     {
2214       \begin{tikzpicture}
2215         \node
2216           [
2217             fit = ( #1 - #2 - medium . north-west )
2218                 ( #3 - #4 - medium . south-east ) ,
2219             inner~sep = 0 pt ,
2220           ]

```

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

```

2221       (#1-#2) { } ;
2222       \node at (#1-#2.center) { #5 } ;
2223     \end{tikzpicture}
2224   }
2225 }

```

17.17 How to draw the dotted lines transparently

```

2226 \cs_set_protected:Npn \__nm_renew_matrix:
2227 {
2228   \RenewDocumentEnvironment { pmatrix } { } {
2229     { \pNiceMatrix }
2230     { \endpNiceMatrix }
2231   }
2232   \RenewDocumentEnvironment { vmatrix } { } {
2233     { \vNiceMatrix }
2234     { \endvNiceMatrix }
2235   }
2236   \RenewDocumentEnvironment { Vmatrix } { } {
2237     { \VNiceMatrix }
2238     { \endVNiceMatrix }
2239   }
2240   \RenewDocumentEnvironment { bmatrix } { } {
2241     { \bNiceMatrix }
2242     { \endbNiceMatrix }
2243   }
2244   \RenewDocumentEnvironment { Bmatrix } { } {
2245     { \BNiceMatrix }
2246     { \endBNiceMatrix }
2247   }
2248 }

```

17.18 We process the options

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

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

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

```

2244 \keys_define:nn { NiceMatrix / Package }
2245 {
2246   renew-dots .bool_set:N = \l__nm_renew_dots_bool ,
2247   renew-dots .value_forbidden:n = true ,
2248   renew-matrix .code:n = \__nm_renew_matrix: ,
2249   renew-matrix .value_forbidden:n = true ,
2250   transparent .meta:n = { renew-dots , renew-matrix } ,
2251   transparent .value_forbidden:n = true ,
2252 }

```

```
2253 \ProcessKeysOptions { NiceMatrix / Package }
```

17.19 Error messages of the package

```
2254 \_nm_msg_new:nn { unknown~cell~for~line~in~code~after }
2255 {
2256   Your~command~\token_to_str:N\line\{#1\}\{#2\}~in~the~'code~after'~
2257   can't~be~executed~because~a~Tikz~node~doesn't~exist.\\
2258   If~you~go~on~this~command~will~be~ignored.
2259 }

2260 \_nm_msg_new:nn { last~col~non~empty~for~NiceArray }
2261 {
2262   In~the~environment~\{g__nm_type_env_str\},~you~must~use~the~option~
2263   'last~col'~without~value~(the~number~of~columns~is~known~by~the~
2264   preamble~of~the~environment).\\
2265   However,~you~can~go~on~for~this~time~
2266   (the~value~'\l_keys_value_tl'~will~be~ignored).
2267 }

2268 \_nm_msg_new:nn { last~col~empty~for~NiceMatrix }
2269 {
2270   In~the~environment~\{g__nm_type_env_str\}~you~can't~use~the~option~
2271   'last~col'~without~value.~You~must~give~the~number~of~that~last~column.\\
2272   If~you~go~on~this~option~will~be~ignored.
2273 }

2274 \_nm_msg_new:nn { Block~too~large }
2275 {
2276   You~try~to~draw~a~block~in~the~cell~#1~#2~of~your~matrix~but~the~matrix~is~
2277   too~small~for~that~block.\\
2278   If~you~go~on,~this~command~line~will~be~ignored.
2279 }

2280 \_nm_msg_new:nn { Impossible~line }
2281 {
2282   A~dotted~line~can't~be~drawn~because~you~have~not~put~
2283   all~the~ampersands~required~on~the~row~#1.\\
2284   If~you~go~on,~this~dotted~line~will~be~ignored.
2285 }

2286 \_nm_msg_new:nn { Wrong~last~row }
2287 {
2288   You~have~used~'last~row=\int_use:N \l__nm_last_row_int'~but~your~environment~
2289   \{g__nm_type_env_str\}~seems~to~have~\int_use:N \c@iRow
2290   rows.~If~you~go~on,~the~value~of~\int_use:N \c@iRow
2291   will~be~used~for~last~row.~You~can~avoid~this~problem~by~using~'last~row'~
2292   without~value~(more~compilations~might~be~necessary).
2293 }

2294 \_nm_msg_new:nn { Yet~in~env }
2295 {
2296   Environments~\{NiceArray\}~(or~\{NiceMatrix\},~etc.)~can't~be~nested.\\
2297   This~error~is~fatal.
2298 }

2299 \_nm_msg_new:nn { Outside~math~mode }
2300 {
2301   The~environment~\{g__nm_type_env_str\}~can~be~used~only~in~math~mode~
2302   (and~not~in~\token_to_str:N \vcenter).\\
2303   This~error~is~fatal.
2304 }

2305 \_nm_msg_new:nn { Option~Transparent~suppressed }
2306 {
2307   The~option~'Transparent'~has~been~renamed~'transparent'.\\
2308   However,~you~can~go~on~for~this~time.
```

```

2309     }
2310 \_nm_msg_new:nn { Option~RenewMatrix~suppressed }
2311 {
2312     The~option~'RenewMatrix'~has~been~renamed~'renew-matrix'.\\
2313     However,~you~can~go~on~for~this~time.
2314 }
2315 \_nm_msg_new:nn { Bad~value~for~letter~for~dotted~lines }
2316 {
2317     The~value~of~key~'\tl_use:N\l_keys_key_tl'~must~be~of~length~1.\\
2318     If~you~go~on,~it~will~be~ignored.
2319 }
2320 \_nm_msg_new:nnn { Unknown~key~for~NiceMatrixOptions }
2321 {
2322     The~key~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~command~
2323     \token_to_str:N \NiceMatrixOptions. \\
2324     If~you~go~on,~it~will~be~ignored. \\
2325     For~a~list~of~the~available~keys,~type~H~<return>.
2326 }
2327 {
2328     The~available~options~are~(in~alphabetic~order):~
2329     allow~duplicate~names,~
2330     code~for~first~col,~
2331     code~for~first~row,~
2332     code~for~last~col,~
2333     code~for~last~row,~
2334     exterior~arraycolsep,~
2335     hlines,~
2336     left~margin,~
2337     letter~for~dotted~lines,~
2338     nullify~dots,~
2339     parallelize~diags,~
2340     renew~dots,~
2341     renew~matrix,~
2342     right~margin,~
2343     small~
2344     and~transparent
2345 }
2346 \_nm_msg_new:nnn { Unknown~option~for~NiceArray }
2347 {
2348     The~option~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~environment~
2349     \{NiceArray\}. \\
2350     If~you~go~on,~it~will~be~ignored. \\
2351     For~a~list~of~the~available~options,~type~H~<return>.
2352 }
2353 {
2354     The~available~options~are~(in~alphabetic~order):~
2355     b,~
2356     c,~
2357     code~after,~
2358     code~for~first~col,~
2359     code~for~first~row,~
2360     code~for~last~col,~
2361     code~for~last~row,~
2362     columns~width,~
2363     create~extra~nodes,~
2364     extra~left~margin,~
2365     extra~right~margin,~
2366     first~col,~
2367     first~row,~
2368     hlines,~
2369     last~col,~
2370     last~row,~
2371     left~margin,~

```

```

2372     name,~
2373     nullify-dots,~
2374     parallelize-diags,~
2375     renew-dots,~
2376     right-margin,~
2377     small~
2378     and~t.
2379 }

```

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).

```

2380 \_nm_msg_new:nnn { Unknown~option~for~NiceMatrix }
2381 {
2382   The~option~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~environment~
2383   \{\g__nm_type_env_str\}. \\
2384   If~you~go~on,~it~will~be~ignored. \\
2385   For~a~list~of~the~available~options,~type~H~<return>.
2386 }
2387 {
2388   The~available~options~are~(in~alphabetic~order):~
2389   code~after,~
2390   code~for~first~col,~
2391   code~for~first~row,~
2392   code~for~last~col,~
2393   code~for~last~row,~
2394   columns~width,~
2395   create~extra~nodes,~
2396   extra~left~margin,~
2397   extra~right~margin,~
2398   first~col,~
2399   first~row,~
2400   hlines,~
2401   last~col,~
2402   last~row,~
2403   left~margin,~
2404   name,~
2405   nullify~dots,~
2406   parallelize~diags,~
2407   renew~dots,~
2408   right~margin~
2409   and~small.
2410 }

```

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

```

2411 \_nm_msg_new:nnn { Duplicate~name }
2412 {
2413   The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
2414   the~same~environment~name~twice.~You~can~go~on,~but,~
2415   maybe,~you~will~have~incorrect~results~especially~
2416   if~you~use~'columns~width=auto'.~If~you~use~nicematrix~inside~some~
2417   environments~of~amsmath,~this~error~may~be~an~artefact.~In~this~case,~
2418   use~the~option~'allow~duplicate~names'.\\
2419   For~a~list~of~the~names~already~used,~type~H~<return>. \\
2420 }
2421 {
2422   The~names~already~defined~in~this~document~are:~
2423   \seq_use:Nnnn \g__nm_names_seq { ,~ } { ,~ } { ~and~ }.
2424 }
2425 \_nm_msg_new:nn { Option~auto~for~columns~width }
2426 {
2427   You~can't~give~the~value~'auto'~to~the~option~'columns~width'~here.~
2428   If~you~go~on,~the~option~will~be~ignored.
2429 }

```

```

2430 \_nm_msg_new:nn { Zero~row }
2431 {
2432   There~is~a~problem.~Maybe~your~environment~\{\g__nm_type_env_str\}~is~empty.~
2433   Maybe~you~have~used~l,~c~and~r~instead~of~L,~C~and~R~in~the~preamble~
2434   of~your~environment.~\{
2435   If~you~go~on,~the~result~may~be~incorrect.
2436 }
2437 \_nm_msg_new:nn { Use~of~::~in~first~position }
2438 {
2439   You~can't~use~the~column~specifier~'\l__nm_letter_for_dotted_lines_str'~in~the~
2440   first~position~of~the~preamble~of~the~environment~\{\g__nm_type_env_str\}.~\{
2441   If~you~go~on,~this~dotted~line~will~be~ignored.
2442 }

```

17.20 Obsolete environments

```

2443 \_nm_msg_new:nn { Obsolete~environment }
2444 {
2445   The~environment~\{\@currentenv\}~is~obsolete.~
2446   We~should~use~#1~instead.
2447 }
2448 \NewDocumentEnvironment { pNiceArrayC } { }
2449 {
2450   \msg_info:nnn { nicematrix } { Obsolete~environment }
2451   { the~option~'last-col' }
2452   \int_set:Nn \l__nm_last_col_int \c_zero_dim
2453   \pNiceArray
2454 }
2455 { \endpNiceArray }
2456 \NewDocumentEnvironment { bNiceArrayC } { }
2457 {
2458   \msg_info:nnn { nicematrix } { Obsolete~environment }
2459   { the~option~'last-col' }
2460   \int_set:Nn \l__nm_last_col_int \c_zero_dim
2461   \bNiceArray
2462 }
2463 { \endbNiceArray }
2464 \NewDocumentEnvironment { BNiceArrayC } { }
2465 {
2466   \msg_info:nnn { nicematrix } { Obsolete~environment }
2467   { the~option~'last-col' }
2468   \int_set:Nn \l__nm_last_col_int \c_zero_dim
2469   \BNiceArray
2470 }
2471 { \endBNiceArray }
2472 \NewDocumentEnvironment { vNiceArrayC } { }
2473 {
2474   \msg_info:nnn { nicematrix } { Obsolete~environment }
2475   { the~option~'last-col' }
2476   \int_set:Nn \l__nm_last_col_int \c_zero_dim
2477   \vNiceArray
2478 }
2479 { \endvNiceArray }
2480 \NewDocumentEnvironment { VNiceArrayC } { }
2481 {
2482   \msg_info:nnn { nicematrix } { Obsolete~environment }
2483   { the~option~'last-col' }
2484   \int_set:Nn \l__nm_last_col_int \c_zero_dim
2485   \VNiceArray
2486 }
2487 { \endVNiceArray }

```

```

2488 \NewDocumentEnvironment { pNiceArrayRC } { }
2489 {
2490   \msg_info:nnn { nicematrix } { Obsolete-environment }
2491   { the~options~'last-col'~and~'first-row' }
2492   \int_set:Nn \l__nm_last_col_int \c_zero_dim
2493   \int_set:Nn \l__nm_first_row_int \c_zero_int
2494   \pNiceArray
2495 }
2496 { \endpNiceArray }

2497 \NewDocumentEnvironment { bNiceArrayRC } { }
2498 {
2499   \msg_info:nnn { nicematrix } { Obsolete-environment }
2500   { the~options~'last-col'~and~'first-row' }
2501   \int_set:Nn \l__nm_last_col_int \c_zero_dim
2502   \int_set:Nn \l__nm_first_row_int \c_zero_int
2503   \bNiceArray
2504 }
2505 { \endbNiceArray }

2506 \NewDocumentEnvironment { BNiceArrayRC } { }
2507 {
2508   \msg_info:nnn { nicematrix } { Obsolete-environment }
2509   { the~options~'last-col'~and~'first-row' }
2510   \int_set:Nn \l__nm_last_col_int \c_zero_dim
2511   \int_set:Nn \l__nm_first_row_int \c_zero_int
2512   \BNiceArray
2513 }
2514 { \endBNiceArray }

2515 \NewDocumentEnvironment { vNiceArrayRC } { }
2516 {
2517   \msg_info:nnn { nicematrix } { Obsolete-environment }
2518   { the~options~'last-col'~and~'first-row' }
2519   \bool_set_true:N \l__nm_last_col_bool
2520   \int_set:Nn \l__nm_first_row_int \c_zero_int
2521   \vNiceArray
2522 }
2523 { \endvNiceArray }

2524 \NewDocumentEnvironment { VNiceArrayRC } { }
2525 {
2526   \msg_info:nnn { nicematrix } { Obsolete-environment }
2527   { the~options~'last-col'~and~'first-row' }
2528   \int_set:Nn \l__nm_last_col_int \c_zero_dim
2529   \int_set:Nn \l__nm_first_row_int \c_zero_int
2530   \VNiceArray
2531 }
2532 { \endVNiceArray }

2533 \NewDocumentEnvironment { NiceArrayCwithDelims } { }
2534 {
2535   \msg_info:nnn { nicematrix } { Obsolete-environment }
2536   { the~option~'last-col' }
2537   \int_set:Nn \l__nm_last_col_int \c_zero_dim
2538   \NiceArrayWithDelims
2539 }
2540 { \endNiceArrayWithDelims }

2541 \NewDocumentEnvironment { NiceArrayRCwithDelims } { }
2542 {
2543   \msg_info:nnn { nicematrix } { Obsolete-environment }
2544   { the~options~'last-col'~and~'first-row' }
2545   \int_set:Nn \l__nm_last_col_int \c_zero_dim
2546   \int_set:Nn \l__nm_first_row_int \c_zero_int
2547   \NiceArrayWithDelims
2548 }

```


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

Changes between version 2.1 and 2.1.2

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

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

³³Before this version, there was an error when using `nicematrix` with Tikz externalization. In any case, it's not possible to externalize the Tikz elements constructed by `nicematrix` because they use the options `overlay` and `remember picture`.

Changes between version 2.1.2 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`).

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

Changes between version 2.1.3 and 2.1.4

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

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

Changes between version 2.1.4 and 2.1.5

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

Option `allow-duplicate-names`.

Changes between version 2.1.5 and 2.2

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

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

Changes between version 2.2 and 2.2.1

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

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

Changes between version 2.2.1 and 2.3

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

Option `hlines`.

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

Changes between version 2.3 and 3.0

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

Composition of exterior rows and columns 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 StackExchange³⁴, 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`.

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> 2155	2311, 2312, 2345, 2346, 2380, 2381, 2396, 2402
<code>\%</code> 2217, 2224, 2231, 2237, 2243, 2258, 2264, 2269, 2274, 2279, 2285, 2286,	948, 2216, 2222, 2230, 2249, 2258, 2263, 2311, 2345, 2394, 2402, 2407
	<code>\}</code> 948, 2216, 2222, 2230,

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

2249, 2258, 2263, 2311, 2345, 2394, 2402, 2407	
\l	964
\	2249, 2250
A	
\array	434
\arraycolsep	161, 163, 165, 460, 603, 604, 663, 666, 674, 678, 703, 710, 742, 780, 782, 796, 855, 883, 1809, 1812, 1854, 1855
\arrayrulewidth	448, 449
\arraystretch	459
\AtBeginDocument	62, 88
B	
\begin	973, 1021, 1255, 1278, 1293, 1610, 1759, 1833, 1846, 1860, 1961, 2047, 2174
\bgroup	350
\Block	507
\BNiceArray	2431, 2474
\bNiceArray	2423, 2465
\BNiceMatrix	2201
\bNiceMatrix	2198
bool commands:	
\bool_do_until:Nn	1126, 1186
\bool_gset_eq:NN	462
\bool_gset_false:N	549
\bool_gset_true:N	537, 862, 1795, 2143
\bool_if:NTF	34, 96, 288, 410, 426, 442, 457, 465, 508, 547, 563, 572, 575, 601, 632, 634, 641, 643, 719, 745, 793, 807, 868, 1018, 1058, 1076, 1085, 1092, 1150, 1178, 1210, 1239, 1247, 1273, 1285, 1310, 1317, 1332, 1347, 1349, 1361, 1365, 1380, 1395, 1397, 1409, 1413, 1418, 1426, 1431, 1437, 1441, 1456, 1460, 1465, 1469, 1498, 1502, 1507, 1511, 1555, 1557, 1568, 1590, 1591, 1592, 1637, 1643, 1649, 1655, 1661, 1686, 1772, 1777, 1784, 1825, 1935, 1945
\bool_if:NTF	654, 667, 684, 1054, 1246, 1636, 1642, 1648, 1654, 1660, 1738, 1752, 1801, 2167
\bool_new:N	11, 27, 52, 53, 60, 61, 83, 86, 87, 128, 129, 131, 132, 133, 135, 136, 1920
\bool_set:Nn	1422, 1435
\bool_set_false:N	994, 1008, 1090, 1091, 1125, 1130, 1185, 1190, 1308, 1359, 1407, 1454, 1496, 1706, 1707, 1750, 1751
\bool_set_true:N	12, 29, 32, 66, 91, 130, 175, 226, 564, 581, 921, 1016, 1134, 1140, 1146, 1154, 1158, 1182, 1194, 1200, 1206, 1214, 1219, 1243, 1713, 1721, 1727, 1735, 1831, 1832, 1925, 1927, 2481
\l_tmpa_bool	994, 1008, 1016, 1018, 1422, 1441
\l_tmpb_bool	1435, 1437
box commands:	
\box_clear_new:N	627
\box_dp:N ..	308, 320, 370, 476, 482, 486, 790
\box_ht:N ..	310, 315, 318, 478, 480, 484, 789
\box_move_down:n	371, 760
\box_move_up:n	385, 751
\box_set_dp:Nn	790
\box_set_ht:Nn	789

\box_use:N	351, 369, 385, 847, 915
\box_use_drop:N	768, 781, 791
\box_wd:N	327, 375, 616, 624, 820, 881
\l_tmpa_box	286, 308, 310, 315, 318, 320, 327, 351, 360, 369, 609, 616, 617, 624, 771, 789, 790, 791, 805, 820, 847, 866, 881, 915
\l_tmpb_box	368, 370, 375, 385

C

\Cdots	499
\cdots	511, 1624
Cdots internal commands:	
_nm_Cdots	499, 511, 1640
cdots internal commands:	
_nm_cdots	1624, 1643
char commands:	
\char_set_catcode_letter:N	2155
\cleaders	1779
\coordinate	379, 384, 665, 682, 693, 705, 2070, 2073, 2102, 2107
\crclr	660
cs commands:	
\cs_generate_variant:Nn	354, 1270, 1957, 1958
\cs_gset:Npn	1063, 1070, 1950, 2059
\cs_gset:Npx	1630
\cs_gset_eq:NN	109, 469
\cs_if_exist:NTF	565, 568, 584, 591, 995, 1009, 1051, 1275, 1287, 1319, 1334, 1367, 1382, 1796, 1843, 1857, 1937, 1980
\cs_if_exist_p:N	1754, 1756
\cs_if_free:NTF	999, 1170, 1231, 1306, 1357, 1405, 1452, 1494
\cs_new:Npn	437, 1670, 1681, 1791, 2128, 2132, 2156
\cs_new_protected:Nn	276, 296, 322, 355, 1042, 1048, 1118, 1249, 1271, 1304, 1313, 1355, 1403, 1450, 1492, 1534, 1628, 1664, 1704, 1744, 1819, 1959, 2064, 2099
\cs_new_protected:Npn	18, 19, 20, 21, 22, 23, 24, 25, 54, 112, 303, 413, 424, 439, 454, 967, 1871, 2094, 2141, 2165
\cs_set:Npn	431, 459, 463, 487, 1120, 1160, 1221, 1742, 1775, 2046, 2096, 2097
\cs_set_eq:NN	100, 428, 429, 430, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 510, 511, 512, 513, 514, 515, 516, 525, 526, 527, 529, 1112, 1623, 1624, 1625, 1626, 1627, 1669, 1773, 1919
\cs_set_protected:Npn	67, 94, 411, 574, 2186

D

\Ddots	501
\ddots	513, 1626
Ddots internal commands:	
_nm_Ddots	501, 513, 1652
ddots internal commands:	
_nm_ddots	1626, 1655
\DeclareOption	12, 13
dim commands:	
\dim_abs:n	1035
\dim_add:Nn	2040
\dim_compare:nNnTF	1034, 1553
\dim_compare_p:nNn	657, 1423, 1436
\dim_eval:n	2060

<code>\dim_gadd:Nn</code>	1351, 1352, 1594, 1602, 1617, 1618, 1854, 1855
<code>\dim_gset:Nn</code> .	307, 309, 314, 317, 319, 326, 476, 478, 480, 482, 484, 486, 603, 604, 616, 624, 673, 677, 816, 877, 1030, 1032, 1260, 1261, 1266, 1267, 1281, 1299, 1439, 1480, 1522, 1762, 1763, 1765, 1766, 1836, 1837, 1840, 1841, 1849, 1852, 1863, 1864, 2050, 2054
<code>\dim_gset_eq:NN</code>	299, 1348, 1350, 1396, 1398, 1444
<code>\dim_gzero:N</code>	300, 301
<code>\dim_gzero_new:N</code>	475, 477, 479, 481, 483, 485, 573, 599, 600, 1251, 1252, 1253, 1254, 1926
<code>\dim_max:nn</code>	308, 310, 315, 318, 320, 327, 818, 879, 1441, 1958, 1996, 2000
<code>\dim_min:nn</code>	1441, 1957, 1987, 1991
<code>\dim_new:N</code>	50, 71, 73, 137, 138, 139, 140, 141, 142
<code>\dim_ratio:nn</code>	1484, 1526, 1560, 1564, 1571, 1575, 1581, 1586, 1597, 1605
<code>\dim_set:Nn</code> .	72, 74, 176, 231, 354, 370, 460, 730, 736, 1094, 1095, 1474, 1476, 1516, 1518, 1537, 1578, 1583, 1778, 1798, 1939, 1967, 1974, 1986, 1990, 1995, 1999, 2011, 2024
<code>\dim_set_eq:NN</code>	608, 619, 1797, 1965, 1972, 2019, 2034
<code>\dim_sub:Nn</code>	2037
<code>\dim_use:N</code>	1543, 1544, 1547, 1548, 1952, 2014, 2015, 2027, 2029, 2071, 2072, 2074, 2075, 2104, 2105, 2109, 2110
<code>\dim_zero:N</code>	733, 739, 1933
<code>\dim_zero_new:N</code>	1080, 1081, 1082, 1083, 1536, 1746, 1747, 1748, 1749, 1827, 1828, 1829, 1830, 1964, 1966, 1971, 1973
<code>\c_max_dim</code>	1965, 1967, 1972, 1974
<code>\g_tmpa_dim</code>	673, 677, 680, 691, 702, 1030, 1035, 2050, 2060
<code>\l_tmpa_dim</code>	370, 371, 385, 730, 733, 752, 777, 789, 1578, 1617, 1797, 1798, 1812
<code>\g_tmpb_dim</code>	1032, 1035, 2054, 2060
<code>\l_tmpb_dim</code> .	736, 739, 762, 784, 790, 1583, 1618
<code>\c_zero_dim</code>	332, 333, 403, 608, 619, 657, 827, 828, 895, 896, 1553, 2080, 2116, 2414, 2422, 2430, 2438, 2446, 2454, 2463, 2472, 2490, 2499, 2507
<code>\dots</code>	515
E	
<code>\egroup</code>	352
else commands:	
<code>\else:</code>	56
<code>\end</code>	984, 1033, 1268, 1282, 1300, 1620, 1767, 1842, 1853, 1865, 2045, 2055, 2183
<code>\endarray</code>	712
<code>\endBNiceArray</code>	2433, 2476
<code>\endbNiceArray</code>	2425, 2467
<code>\endBNiceMatrix</code>	2202
<code>\endbNiceMatrix</code>	2199
<code>\endNiceArrayWithDelims</code>	926, 934, 942, 950, 958, 966, 2502, 2511
<code>\endpNiceArray</code>	2417, 2458
<code>\endpNiceMatrix</code>	2190
<code>\endVNiceArray</code>	2449, 2494
<code>\endvNiceArray</code>	2441, 2485
<code>\endVNiceMatrix</code>	2196
<code>\endvNiceMatrix</code>	2193
<code>\everycr</code>	473, 489
exp commands:	
<code>\exp_after:wN</code>	116
<code>\exp_args:NV</code>	532, 651
<code>\exp_not:n</code>	2151
<code>\ExplSyntaxOff</code>	1074, 1954, 2062
<code>\ExplSyntaxOn</code>	1060, 1947, 2056
F	
<code>\fi</code>	185
fi commands:	
<code>\fi:</code>	58
fp commands:	
<code>\fp_to_dim:n</code>	1539
G	
group commands:	
<code>\group_begin:</code>	98, 607, 1050, 2154
<code>\group_end:</code>	103, 625, 1115, 2164
H	
<code>\halign</code>	491, 493
hbox commands:	
<code>\hbox:n</code>	41, 43, 45, 381, 778, 2136
<code>\hbox_overlap_left:n</code>	822
<code>\hbox_overlap_right:n</code>	884, 1799
<code>\hbox_set:Nn</code>	368, 609, 617, 771
<code>\hbox_set:Nw</code>	286, 360, 647, 805, 866
<code>\hbox_set_end:</code>	325, 366, 716, 814, 875
<code>\hbox_to_wd:nn</code>	375, 1780, 1810
<code>\Hdotsfor</code>	505
<code>\hdotsfor</code>	516
<code>\hdottedline</code>	503
<code>\hfil</code>	377, 529
<code>\hfill</code>	1788
<code>\hrule</code>	448
<code>\Hspace</code>	504
<code>\hspace</code>	1667
<code>\hss</code>	529, 1785
I	
<code>\ialign</code>	463, 487
<code>\iddots</code>	502
<code>\iddots</code>	36, 514, 1627
iddots internal commands:	
<code>_nm_Iddots</code>	502, 514, 1658
iddots internal commands:	
<code>_nm_iddots</code>	1627, 1661
if commands:	
<code>\if_mode_math:</code>	56
<code>\ifstandalone</code>	568
int commands:	
<code>\int_add:Nn</code>	1128, 1129, 1212, 1213
<code>\int_compare:nNnTF</code> .	279, 281, 289, 292, 305, 312, 444, 446, 538, 579, 629, 638, 661, 717, 721, 728, 734, 740, 747, 756, 977, 1012, 1044, 1056, 1131, 1133, 1137, 1139, 1143, 1145, 1191, 1193, 1197, 1199, 1203, 1205, 1472, 1514, 1673, 1710, 1724, 1821, 1873, 1875, 1877, 1879, 1881, 1886, 1888, 1894, 1896, 1902, 1904, 1906, 1911, 1913, 2134, 2158

<code>\int_compare:nTF</code>	237
<code>\int_compare_p:nNn</code>	1807, 2169, 2170
<code>\int_eval:n</code>	979, 1676, 2015, 2019, 2030, 2034, 2109, 2149, 2150
<code>\int_gadd:Nn</code>	1679
<code>\int_gdecr:N</code>	1054
<code>\int_gincr:N</code> 278, 298, 571, 690, 701, 863, 1932	
<code>\int_gset:Nn</code>	284, 521, 548, 683, 864
<code>\int_gset_eq:NN</code>	540, 724, 1053, 1055
<code>\int_gsub:Nn</code>	1057
<code>\int_gzero:N</code>	441
<code>\int_gzero_new:N</code>	520, 522, 523, 524, 546
<code>\int_incr:N</code>	1471, 1513
<code>\int_max:nn</code>	285, 865
<code>\int_new:N</code>	48, 49, 77, 79, 81, 84
<code>\int_set:Nn</code>	78, 80, 82, 85, 252, 586, 593, 1121, 1122, 1123, 1124, 1559, 1563, 1570, 1574, 1588, 1708, 1709, 1712, 1716, 1718, 1720, 1726, 1730, 1732, 1734, 2007, 2008, 2414, 2422, 2430, 2438, 2446, 2454, 2455, 2463, 2464, 2472, 2473, 2482, 2490, 2491, 2499, 2507, 2508
<code>\int_step_inline:nnn</code>	1741
<code>\int_step_inline:nnnn</code>	1611
<code>\int_step_variable:nNn</code>	2009, 2022
<code>\int_step_variable:nnNn</code>	1962, 1969, 1976, 1978, 2066, 2068
<code>\int_sub:Nn</code>	1152, 1153, 1188, 1189
<code>\int_use:N</code>	339, 340, 341, 346, 347, 393, 394, 395, 400, 401, 419, 420, 542, 584, 587, 665, 682, 695, 696, 707, 708, 836, 837, 843, 904, 905, 906, 911, 912, 996, 1002, 1003, 1004, 1010, 1013, 1025, 1026, 1027, 1063, 1064, 1071, 1109, 1163, 1164, 1173, 1174, 1175, 1181, 1224, 1225, 1234, 1235, 1236, 1242, 1257, 1258, 1259, 1263, 1264, 1265, 1276, 1280, 1289, 1290, 1296, 1297, 1322, 1323, 1324, 1337, 1338, 1339, 1370, 1371, 1372, 1385, 1386, 1387, 1631, 1632, 1676, 1697, 1698, 1754, 1756, 1796, 1797, 1839, 1844, 1851, 1858, 1937, 1940, 1951, 1981, 1984, 1994, 2041, 2049, 2052, 2053, 2059, 2081, 2117, 2147, 2148, 2248, 2249, 2250
<code>\int_zero:N</code>	189, 190, 262, 267, 270, 271
<code>\int_zero_new:N</code>	1078, 1079, 1086, 1087, 1088, 1089
<code>\c_one_int</code>	237, 279, 281, 312, 1057, 1309, 1360, 1408, 1455, 1472, 1514
<code>\g_tmpa_int</code>	683, 690, 696, 701, 708
<code>\l_tmpa_int</code>	1559, 1563, 1570, 1574, 1598, 1606, 1611, 1716, 1717, 1730, 1731
<code>\l_tmpb_int</code>	1588, 1600, 1608
<code>\c_zero_int</code>	289, 305, 629, 728, 740, 747, 1044, 1309, 1360, 1408, 1807, 1821, 1873, 1875, 1877, 1879, 1886, 1894, 1902, 1911, 2455, 2464, 2473, 2482, 2491, 2508
iow commands:	
<code>\iow_now:Nn</code>	1060, 1061, 1068, 1074, 1947, 1948, 1954, 2056, 2057, 2062
K	
<code>\kern</code>	45
keys commands:	
<code>\keys_define:nn</code>	143, 171, 194, 216, 248, 255, 265, 1921, 2204
<code>\l_keys_key_tl</code>	2279, 2284, 2310, 2344
<code>\keys_set:nn</code>	247, 576, 577, 972, 1934
<code>\l_keys_value_tl</code>	2226, 2375
L	
<code>\Ldots</code>	498
<code>\ldots</code>	510, 1623
Ldots internal commands:	
<code>__nm_Ldots</code>	498, 510, 515, 1634
ldots internal commands:	
<code>__nm_ldots</code>	1623, 1637
<code>\left</code>	612, 621, 774
<code>\line</code>	1112, 2216
<code>\lineskip</code>	731, 737
M	
<code>\makebox</code>	369
math commands:	
<code>\c_math_toggle_token</code>	287, 324, 611, 613, 620, 622, 650, 713, 773, 787, 806, 813, 867, 874, 1783, 1786
<code>\mathinner</code>	38
<code>\mkern</code>	40, 42, 44, 45
msg commands:	
<code>\msg_error:nn</code>	18
<code>\msg_error:nnn</code>	19, 1180, 1241
<code>\msg_error:nnnn</code>	20, 2172
<code>\msg_fatal:nn</code>	21, 22
<code>\msg_info:nnn</code>	2412, 2420, 2428, 2436, 2444, 2452, 2461, 2470, 2479, 2488, 2497, 2505
<code>\msg_new:nnn</code>	23
<code>\msg_new:nnnn</code>	24
<code>\msg_redirect_name:nnn</code>	26
<code>\msg_warning:nn</code>	35
<code>\multicolumn</code>	506, 1669, 1683, 1689, 1701, 2136
<code>\myfiledate</code>	8
<code>\myfileversion</code>	9
N	
<code>\newcolumnntype</code>	357, 495, 496, 497, 532
<code>\NewDocumentCommand</code>	246, 1634, 1640, 1646, 1652, 1658, 1688, 1692
<code>\NewDocumentEnvironment</code>	557, 919, 927, 935, 943, 951, 959, 969, 1930, 2410, 2418, 2426, 2434, 2442, 2450, 2459, 2468, 2477, 2486, 2495, 2503
<code>\NewExpandableDocumentCommand</code>	2124
<code>\NiceArrayWithDelims</code>	924, 932, 940, 948, 956, 964, 2500, 2509
<code>\NiceMatrixOptions</code>	246, 2285
nm internal commands:	
<code>__nm_actualization_for_first_and-last_row:</code>	303, 328, 815, 876
<code>__nm_actually_draw_Ldots:</code>	1310, 1313, 1740
<code>__nm_adapt_S_column:</code>	94, 109, 561
<code>__nm_add_to_empty_cells:</code>	1628, 1638, 1644, 1650, 1656, 1662, 1666
<code>__nm_adjust_with_col_nodes:</code>	1271, 1346, 1394
<code>__nm_after_array:</code>	798, 1042
<code>__nm_after_array_i:</code>	1045, 1048

<code>_nm_array:</code>	424, 651		
<code>\l_nm_auto_columns_width_bool</code>			1632, 1754, 1756, 1796, 1797, 1844, 1858,
.....	133, 175, 656, 669, 1927		1981, 1984, 1994, 2049, 2052, 2059, 2081, 2117
<code>_nm_begin_of_row:</code>	282, 296, 804		
<code>_nm_Block:</code>	507, 2124		<code>_nm_error:n</code> . 18, 220, 224, 230, 239, 242,
<code>\l_nm_block_auto_columns_width_bool</code> .			251, 253, 261, 263, 269, 274, 723, 1046, 1822
.....	572, 656, 670, 1920, 1925, 1935, 1945		<code>_nm_error:nn</code>
<code>_nm_Block_i</code>	2126, 2128		19, 182
<code>_nm_Block_ii:nnnn</code>	2130, 2132		<code>_nm_error:nnn</code>
<code>_nm_Block_iii:nnnn</code>	2139, 2141		20, 1770
<code>_nm_Block_iv:nnnnn</code>	2146, 2165		<code>_nm_everycr:</code>
<code>\g_nm_Cdots_lines_tl</code>	550, 1101		437, 470, 473
<code>_nm_Cell:</code>	119, 276, 361, 495, 496, 497		<code>_nm_everycr_i:</code>
<code>\g_nm_code_after_tl</code>			438, 439
.....	76, 187, 541, 1113, 1114, 2144		<code>\l_nm_exterior_arraycolsep_bool</code>
<code>\l_nm_code_for_first_col_tl</code>	145, 808	
<code>\l_nm_code_for_first_row_tl</code>	149, 290		128, 227, 634, 643, 1805
<code>\l_nm_code_for_last_col_tl</code>	147, 869		<code>\l_nm_extra_left_margin_dim</code>
<code>\l_nm_code_for_last_row_tl</code>	151, 293	
<code>\g_nm_col_int</code>	278, 279, 285, 341,		141, 166, 649, 852
	347, 395, 401, 420, 441, 523, 538, 540, 542,		<code>\g_nm_extra_nodes_bool</code>
	863, 865, 906, 912, 1053, 1054, 1143, 1203,	
	1290, 1297, 1631, 1676, 1679, 1698, 1724,		136, 462, 537, 1085, 1795, 2143
	1875, 1902, 2022, 2041, 2053, 2148, 2150, 2170		<code>\l_nm_extra_nodes_bool</code>
<code>\g_nm_col_total_int</code>	284, 285,		135, 159, 462
	524, 685, 686, 864, 865, 1053, 1969, 1979, 2068		<code>\l_nm_extra_right_margin_dim</code>
<code>\c_nm_colortbl_loaded_bool</code> ...	61, 66, 465	
<code>\l_nm_columns_width_dim</code>			142, 167, 715, 890
.....	50, 176, 231, 657, 678, 1933, 1939		<code>_nm_extract_coords:</code>
<code>_nm_create_extra_nodes:</code>			2094, 2101
...	1085, 1246, 1247, 1739, 1824, 1959, 2046		<code>_nm_fatal:n</code>
<code>_nm_create_nodes:</code>	2006, 2044, 2064		21, 57, 563
<code>\l_nm_ddots_int</code>	1078, 1471, 1472		<code>_nm_fatal:nn</code>
<code>\g_nm_Ddots_lines_tl</code>	553, 1099		22
<code>_nm_define_env:n</code>			<code>\l_nm_final_i_int</code>
.....	967, 986, 987, 988, 989, 990, 991	
<code>\l_nm_delta_x_one_dim</code> ...	1080, 1474, 1484		1088, 1123, 1128, 1131, 1152, 1157,
<code>\l_nm_delta_x_two_dim</code> ...	1082, 1516, 1526		1163, 1174, 1181, 1264, 1338, 1386, 1709, 1731
<code>\l_nm_delta_y_one_dim</code> ...	1081, 1476, 1484		<code>\l_nm_final_j_int</code>
<code>\l_nm_delta_y_two_dim</code> ...	1083, 1518, 1526		1089, 1124, 1129, 1137, 1143, 1153, 1157,
<code>_nm_dotfill:</code>	1773, 1775, 1815		1164, 1175, 1265, 1339, 1387, 1726, 1732, 1734
<code>\g_nm_dp_ante_last_row_dim</code>			<code>\l_nm_final_open_bool</code>
.....	299, 481, 482, 763	
<code>\g_nm_dp_last_row_dim</code>			1091, 1130, 1134, 1140, 1146,
.....	299, 300, 319, 320, 485, 486, 737, 763		1150, 1178, 1247, 1285, 1332, 1349, 1380,
<code>\g_nm_dp_row_zero_dim</code> 307, 308, 475, 476, 731			1397, 1418, 1431, 1465, 1507, 1557, 1568,
<code>\c_nm_draft_bool</code>			1591, 1592, 1707, 1727, 1735, 1738, 1751, 1832
.....	11, 12, 34, 410, 1686, 1772, 1825		<code>_nm_find_extremities_of_line:nnnn</code> ..
<code>_nm_draw_Cdots:nn</code>	1355	
<code>_nm_draw_Ddots:nn</code>	1450		1118, 1309, 1360, 1408, 1455, 1497
<code>_nm_draw_Hdotsfor:nnn</code>	1696, 1704		<code>\l_nm_first_col_int</code>
<code>_nm_draw_Iddots:nn</code>	1492	
<code>_nm_draw_Ldots:nn</code>	1304		79, 80, 189, 267, 281, 629,
<code>_nm_draw_tikz_line:</code>			661, 740, 1807, 1873, 1969, 1979, 2008, 2068
1353, 1399, 1446, 1488, 1530, 1534, 1768, 1867			<code>\l_nm_first_row_int</code>
<code>_nm_draw_Vdots:nn</code>	1403		77, 78,
<code>_nm_end_Cell:</code> . 121, 322, 365, 495, 496, 497			190, 271, 521, 728, 747, 1877, 1962, 1976,
<code>\g_nm_env_int</code>			2007, 2066, 2455, 2464, 2473, 2482, 2491, 2508
.....	48, 339, 393, 571, 584, 587, 665,		<code>_nm_gobble_ampersands:n</code> 2137, 2156, 2161
	682, 695, 707, 836, 904, 1002, 1015, 1025,		<code>_nm_Hdotsfor:</code>
	1063, 1109, 1173, 1234, 1257, 1263, 1276,		505, 516, 1681
	1280, 1289, 1296, 1322, 1337, 1370, 1385,		<code>_nm_Hdotsfor_i</code>
			1684, 1688, 1692
			<code>\g_nm_Hdotsfor_lines_tl</code> ..
			555, 1097, 1694
			<code>_nm_hdottedline:</code>
			503, 1791
			<code>\l_nm_hlines_bool</code>
			131, 154, 442
			<code>_nm_Hspace:</code>
			504, 1664
			<code>\g_nm_ht_last_row_dim</code>
		
			301, 317, 318, 483, 484, 737
			<code>\g_nm_ht_row_one_dim</code> 314, 315, 479, 480, 752
			<code>\g_nm_ht_row_zero_dim</code>
		
			309, 310, 477, 478, 731, 752
			<code>_nm_i:</code>
			1962, 1964,
			1965, 1966, 1967, 1976, 1981, 1985, 1986,
			1987, 1988, 1994, 1995, 1996, 1997, 2009,
			2011, 2014, 2015, 2019, 2020, 2066, 2072,
			2075, 2081, 2084, 2096, 2105, 2110, 2117, 2120
			<code>\l_nm_iddots_int</code>
			1079, 1513, 1514
			<code>\g_nm_Iddots_lines_tl</code>
			554, 1100
			<code>_nm_if_not_empty_cell:nn</code>
			992
			<code>_nm_if_not_empty_cell:nnTF</code>
		
			1157, 1217, 1717, 1731

<code>\l_nm_impossible_line_bool</code>	<code>\l_nm_NiceArray_bool</code>
..... 60, 1182, 1243, 1308, 1310, 53, 575, 601, 632, 641, 745, 921, 1803
1359, 1361, 1407, 1409, 1454, 1456, 1496, 1498	
<code>\l_nm_in_env_bool</code>	<code>\g_nm_NiceMatrixBlock_int</code>
..... 52, 563, 564 49, 1932, 1937, 1940, 1951
<code>\l_nm_initial_i_int</code>	<code>_nm_node_for_multicolumn:nn</code> ..
..... 1086, 1121, 1188, 1191, 1212, 1218,	2092, 2099
1224, 1235, 1242, 1258, 1323, 1371, 1708, 1717	
<code>\l_nm_initial_j_int</code>	<code>\l_nm_nullify_dots_bool</code>
..... 1087, 1122, 1189, 1197, 1203, 1213, 1218, 132, 158, 1637, 1643, 1649, 1655, 1661
1225, 1236, 1259, 1324, 1372, 1712, 1718, 1720	
<code>\l_nm_initial_open_bool</code>	<code>_nm_old_multicolumn</code>
..... 1090, 1190, 1194,	1669, 1672
1200, 1206, 1210, 1239, 1246, 1273, 1317,	
1347, 1365, 1395, 1413, 1426, 1460, 1502,	<code>\l_nm_parallelize_diags_bool</code>
1555, 1590, 1706, 1713, 1721, 1738, 1750, 1831 129, 130, 155, 1076, 1469, 1511
<code>_nm_instruction_of_type:n</code>	
..... 411, 413, 1636, 1642, 1648, 1654, 1660	<code>\l_nm_pos_env_str</code>
<code>\l_nm_inter_dots_dim</code> 126, 127, 257, 258, 259, 435, 749, 758
..... 71, 72, 1095, 1560, 1564,	<code>_nm_pre_array:</code>
1571, 1575, 1581, 1586, 1598, 1606, 1778, 1781	454, 598
<code>_nm_j:</code>	<code>\c_nm_preamble_first_col_tl</code>
1969, 1971,	630, 800
1972, 1973, 1974, 1979, 1981, 1985, 1988,	<code>\c_nm_preamble_last_col_tl</code>
1990, 1991, 1994, 1997, 1999, 2000, 2022,	639, 858
2024, 2028, 2030, 2034, 2035, 2068, 2071,	<code>\l_nm_radius_dim</code>
2074, 2081, 2084, 2097, 2104, 2109, 2117, 2120	73, 74, 1094, 1615
<code>\l_nm_l_dim</code> 1536, 1537, 1553, 1560, 1564,	<code>\l_nm_renew_dots_bool</code> ..
1571, 1575, 1581, 1586, 1598, 1599, 1606, 1607	156, 226, 508, 2206
<code>\l_nm_last_col_bool</code>	<code>_nm_renew_matrix:</code> 218, 221, 225, 2186, 2208
2481	<code>_nm_renew_NC@rewrite@S:</code>
<code>\g_nm_last_col_found_bool</code>	112, 547
..... 86, 549, 684, 793, 862, 1054	<code>_nm_renewcolumn:nn</code>
<code>\l_nm_last_col_int</code>	355, 528, 529
84, 85, 252,	<code>_nm_retrieve_coords:nn</code>
262, 270, 638, 977, 979, 2414, 2422, 2430,	1249, 1270, 1315, 1363, 1411, 1424, 1458, 1500
2438, 2446, 2454, 2463, 2472, 2490, 2499, 2507	
<code>\l_nm_last_row_int</code>	<code>\c_nm_revtext_bool</code>
81, 82, 191, 272,	27, 29, 32, 426
292, 446, 579, 586, 593, 717, 721, 724, 734,	<code>\g_nm_right_delim_dim</code> ..
756, 1056, 1881, 1888, 1896, 1906, 1913, 2248	600, 604, 624, 888
<code>\l_nm_last_row_without_value_bool</code> ..	<code>\l_nm_right_margin_dim</code>
..... 83, 581, 719, 1058 138, 162, 714, 889, 1813, 2042
<code>\g_nm_last_vdotted_col_int</code>	
..... 538, 540, 546, 548	<code>\g_nm_row_int</code> 289, 292, 298, 305, 312, 340,
<code>\g_nm_ldots_lines_tl</code>	346, 394, 400, 419, 444, 446, 520, 521, 721,
551, 1102	724, 837, 843, 905, 911, 1044, 1055, 1057,
<code>\g_nm_left_delim_dim</code>	1131, 1631, 1676, 1697, 1839, 1851, 1879,
599, 603, 616, 850	1881, 1886, 1888, 1894, 1896, 1904, 1906,
<code>\l_nm_left_margin_dim</code>	1911, 1913, 2009, 2147, 2149, 2169, 2249, 2250
..... 137, 160, 648, 851, 1813, 2039	<code>\g_nm_row_total_int</code>
<code>\l_nm_letter_for_dotted_lines_str</code> 522, 1055, 1064, 1071, 1962, 1976, 2066
..... 238, 244, 245, 531, 532, 2401	<code>\c_nm_siunitx_loaded_bool</code> ..
<code>_nm_line:nn</code>	87, 91, 96, 547
1112, 1744	<code>\l_nm_small_bool</code>
<code>\g_nm_max_cell_width_dim</code> 153, 288, 457, 807, 868, 1092, 1777, 1784
..... 326, 327, 573, 674, 1926, 1952	<code>\l_nm_stop_loop_bool</code>
<code>_nm_msg_new:nn</code>	1125, 1126, 1154, 1158, 1185, 1186, 1214, 1219
23, 2214,	<code>\c_nm_table_collect_begin_tl</code> 104, 106, 119
2220, 2228, 2234, 2240, 2246, 2254, 2256,	<code>\c_nm_table_print_tl</code>
2261, 2267, 2272, 2277, 2387, 2392, 2399, 2405	107, 108, 121
<code>_nm_msg_new:nnn</code> 24, 2282, 2308, 2342, 2373	<code>_nm_test_if_math_mode:</code>
<code>_nm_msg_redirect_name:nn</code> 54, 562, 931, 939, 947, 955, 963
25, 233	<code>\l_nm_the_array_box</code>
<code>_nm_multicolumn:nnn</code>	627, 647, 768, 781
506, 1670	<code>\g_nm_type_env_str</code>
<code>\g_nm_multicolumn_cells_seq</code> 75, 559, 560, 922, 923, 929, 930,
..... 518, 1675, 1988, 1997, 2090	937, 938, 945, 946, 953, 954, 961, 962, 971,
<code>\g_nm_multicolumn_sizes_seq</code> 519, 1677, 2091	1116, 2222, 2230, 2249, 2263, 2345, 2394, 2402
<code>\l_nm_name_str</code>	<code>\g_nm_Vdots_lines_tl</code>
134, 184,	552, 1098
343, 345, 397, 399, 582, 591, 594, 840, 842,	<code>_nm_vdottedline:n</code>
908, 910, 1066, 1070, 2083, 2084, 2119, 2120	542, 1819
<code>\g_nm_names_seq</code>	<code>_nm_vline:</code>
51, 181, 183, 2385	574, 1871
	<code>_nm_vline_i:</code>
 67, 1882, 1889, 1897, 1907, 1914, 1919
	<code>\g_nm_width_first_col_dim</code> 140, 743, 816, 819
	<code>\g_nm_width_last_col_dim</code> 139, 795, 877, 880
	<code>\g_nm_x_final_dim</code>
 1253, 1266, 1299, 1423, 1436, 1442,
	1444, 1475, 1483, 1517, 1525, 1543, 1580,
	1596, 1748, 1765, 1829, 1840, 1852, 1855, 1864
	<code>\g_nm_x_initial_dim</code> ..
	1251, 1260, 1281,
	1423, 1436, 1439, 1442, 1444, 1475, 1483,
	1517, 1525, 1544, 1580, 1594, 1596, 1614,
	1617, 1746, 1762, 1827, 1836, 1849, 1854, 1863

<code>\g_nm_y_final_dim</code> . . . 1254, 1267, 1348, 1350, 1352, 1396, 1398, 1477, 1480, 1519, 1522, 1547, 1585, 1604, 1749, 1766, 1830, 1841	<code>\skip_vertical:n</code> 449, 777, 784
<code>\g_nm_y_initial_dim</code> 1252, 1261, 1348, 1350, 1351, 1396, 1398, 1477, 1482, 1519, 1524, 1548, 1585, 1602, 1604, 1614, 1618, 1747, 1763, 1828, 1837	<code>\c_zero_skip</code> 474, 490
<code>\noalign</code> 438, 469, 1793	str commands:
<code>\node</code> . . 336, 390, 831, 899, 2076, 2112, 2175, 2182	<code>\c_colon_str</code> 245
<code>\normalbaselines</code> 456	<code>\str_gclear:N</code> 1116
<code>\nulldelimiterspace</code> 608, 619	<code>\str_gset:Nn</code> 560, 923, 930, 938, 946, 954, 962, 971
O	<code>\str_if_empty:NTF</code> 343, 397, 559, 582, 840, 908, 922, 929, 937, 945, 953, 961, 1066, 2083, 2119
<code>\omit</code> 661, 662, 689, 700	<code>\str_if_eq:nnTF</code> 174, 229, 749, 758
P	<code>\str_new:N</code> 75, 126, 134, 244
<code>\path</code> 1760	<code>\str_set:Nn</code> 127, 180, 238, 257, 258, 259
peek commands:	<code>\str_set_eq:NN</code> 184, 245
<code>\peek_charcode_remove_ignore_spaces:NTF</code> 2160	<code>\l_tmpa_str</code> 180, 181, 183, 184
<code>\pgfpathcircle</code> 1613	T
<code>\pgfpoint</code> 1614	<code>\tabskip</code> 474, 490
<code>\pgfpointanchor</code> 1029, 1031	TeX and L ^A T _E X 2 _ε commands:
<code>\pgfusepath</code> 1616	<code>\@acol</code> 430
<code>\phantom</code> 1637, 1643, 1649, 1655, 1661	<code>\@acoll</code> 428
<code>\pNiceArray</code> 2415, 2456	<code>\@acolr</code> 429
<code>\pNiceMatrix</code> 2189	<code>\@addtopreamble</code> 574
prg commands:	<code>\@array@array</code> 432
<code>\prg_do_nothing:</code> 100, 109, 469, 1773	<code>\@arrayacol</code> 428, 429, 430
<code>\prg_replicate:nn</code> 685, 686, 1689, 1701	<code>\@arrayrule</code> 574
<code>\prg_return_false:</code> 1006, 1019, 1036	<code>\@arstrutbox</code> 476, 478, 480, 482, 484, 486
<code>\prg_return_true:</code> 997, 1037	<code>\@currenvir</code> 2407
<code>\prg_set_conditional:Npnn</code> 992	<code>\@halignto</code> 431
<code>\ProcessKeysOptions</code> 2213	<code>\@height</code> 448
<code>\ProcessOptions</code> 14	<code>\@ifclassloaded</code> 28, 31
<code>\ProvideDocumentCommand</code> 36	<code>\@ifnextchar</code> 525
<code>\ProvidesExplPackage</code> 6	<code>\@ifpackageloaded</code> 64, 90
Q	<code>\@mainaux</code> 1060, 1061, 1068, 1074, 1947, 1948, 1954, 2056, 2057, 2062
quark commands:	<code>\@temptokena</code> 99, 102, 116, 118
<code>\q_stop</code> 2094, 2101, 2126, 2128	<code>\c@MaxMatrixCols</code> 978
R	<code>\CT@arc@</code> 67
<code>\raise</code> 41, 43, 45	<code>\CT@everycr</code> 467
<code>\relax</code> 14, 526, 527	<code>\CT@row@color</code> 469
<code>\renewcommand</code> 114	<code>\ifmeasuring@</code> 179
<code>\RenewDocumentEnvironment</code> 2188, 2191, 2194, 2197, 2200	<code>\NC@find</code> 100, 123
<code>\RequirePackage</code> 2, 4, 5, 15, 16, 17	<code>\NC@find@W</code> 527
<code>\right</code> 612, 621, 786	<code>\NC@find@w</code> 526
S	<code>\NC@rewrite@S</code> 101, 114
<code>\scriptstyle</code> 288, 807, 868, 1784	<code>\new@ifnextchar</code> 525
seq commands:	<code>\p@</code> 41, 43, 45
<code>\seq_gclear_new:N</code> 518, 519	<code>\pgf@x</code> 1030, 1032, 1260, 1266, 1281, 1299, 1762, 1765, 1836, 1840, 1849, 1852, 1863, 1864, 1991, 2000, 2050, 2054
<code>\seq_gput_left:Nn</code> 183, 1675, 1677	<code>\pgf@y</code> 1261, 1267, 1763, 1766, 1837, 1841, 1987, 1996
<code>\seq_if_in:NnTF</code> 181, 1988, 1997	<code>\pgfutil@firstofone</code> 1256, 1262, 1279, 1294, 1761, 1764, 1834, 1838, 1847, 1850, 1861, 1983, 1993, 2048, 2051
<code>\seq_mapthread_function:NNN</code> 2089	<code>\tikz@library@external@loaded</code> . . . 565, 1051
<code>\seq_new:N</code> 51	<code>\tikz@parse@node</code> 1256, 1262, 1279, 1294, 1761, 1764, 1834, 1838, 1847, 1850, 1861, 1983, 1993, 2048, 2051
<code>\seq_use:Nnnn</code> 2385	tex commands:
skip commands:	<code>\tex_the:D</code> 118
<code>\skip_horizontal:N</code> . . 663, 680, 691, 702, 710	<code>\tikz</code> 329, 378, 383, 389, 664, 681, 692, 704, 824, 892
<code>\skip_horizontal:n</code> 536, 648, 649, 666, 703, 714, 715, 742, 743, 780, 782, 795, 796, 848, 855, 883, 886, 1809	<code>\tikzset</code> 567, 569, 1052, 1103, 2005, 2043

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 exterior-arraycolsep	14
15.5	The class option draft	15
15.6	A technical problem with the argument of <code>\</code>	15
15.7	Obsolete environments	15
16	Examples	15
16.1	Dotted lines	15
16.2	Width of the columns	18
16.3	How to highlight cells of the matrix	18
16.4	Direct utilisation of the Tikz nodes	21
17	Implementation	22
17.1	Declaration of the package and extensions loaded	22
17.2	Technical definitions	23
17.2.1	Variables for the exterior rows and columns	25
17.2.2	The column S of siunitx	26
17.3	The options	28
17.4	Important code used by <code>{NiceArrayWithDelims}</code>	32
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”	48
17.8	After the construction of the array	50
17.9	The actual instructions for drawing the dotted line with Tikz	60
17.10	User commands available in the new environments	62
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	65
17.13	The vertical rules	68
17.14	The environment <code>{NiceMatrixBlock}</code>	69
17.15	The extra nodes	70
17.16	Block matrices	73
17.17	How to draw the dotted lines transparently	75
17.18	We process the options	75
17.19	Error messages of the package	76
17.20	Obsolete environments	79
18	History	81
	Index	83