

The package `nicematrix`*

F. Pantigny
fpantigny@wanadoo.fr

April 6, 2020

Abstract

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

1 Presentation

This package can be used with `xelatex`, `lualatex`, `pdflatex` but also by the classical workflow `latex-dvips-ps2pdf` (or Adobe Distiller). Two or three compilations may be necessary. This package requires and **loads** the packages `l3keys2e`, `xparse`, `array`, `amsmath`, `pgfcore` and the module `shapes` of PGF (`tikz` is *not* loaded). The final user only has to load the package 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.

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

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 3.15 of `nicematrix`, at the date of 2020/04/06.

¹If the class option **draft** is used, these dotted lines will not be drawn for a faster compilation.

2 The environments of this package

The package `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. 8 the section relating to `{NiceArray}`.

3 The continuous dotted lines

Inside the environments of the package `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. It's possible to change the color of these lines with the option `color`.⁵

```
\begin{bNiceMatrix}
a_1      & \Cdots &      & & a_1    \\
\Vdots   & a_2      & \Cdots & & a_2    \\
          & \Vdots & \Ddots[color=red] & & \\
\\
a_1      & a_2      &      & & a_n
\end{bNiceMatrix}
```

$$\begin{bmatrix} a_1 & \cdots & \cdots & \cdots & a_1 \\ \vdots & & & & \vdots \\ \vdots & a_2 & \cdots & \cdots & a_2 \\ \vdots & \vdots & & & \vdots \\ a_1 & a_2 & & & a_n \end{bmatrix}$$

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

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

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

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. 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. 19).

⁵It's also possible to change the color of all theses dotted lines with the option `xdots/color` (`xdots` to remind that it works for `\Cdots`, `\Ldots`, `\Vdots`, etc.): cf. p. 5.

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

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

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

In the first column of this example, there are two instructions `\Vdots` but only one dotted line is drawn (there is no overlapping graphic objects in the resulting PDF⁶).

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

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

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

There are also other means to change the size of the matrix. Someone might want to use the optional argument of the command `\` for the vertical dimension and a command `\hspace*` in a cell for the horizontal dimension.⁷

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

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

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

3.1 The option `nullify-dots`

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

```
$A = \begin{pmatrix}
h & i & j & k & l & m \\
x & & & & & x
\end{pmatrix}$
```

$$A = \begin{pmatrix} h & i & j & k & l & m \\ x & & & & & x \end{pmatrix}$$

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

```
$B = \begin{pmatrix}
h & i & j & k & l & m \\
x & \ldots & \ldots & \ldots & \ldots & x
\end{pmatrix}$
```

$$B = \begin{pmatrix} h & i & j & k & l & m \\ x & \dots & \dots & \dots & \dots & x \end{pmatrix}$$

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

```
$C = \begin{pNiceMatrix}
h & i & j & k & l & m \\
x & \Ldots & \Ldots & \Ldots & \Ldots & x
\end{pNiceMatrix}$
```

$$C = \begin{pmatrix} h & i & j & k & l & m \\ x & \dots & \dots & \dots & \dots & x \end{pmatrix}$$

⁶And it's not possible to draw a `\Ldots` and a `\Cdots` line between the same cells.

⁷In `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. 12

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 row. It's possible by using the option `nullify-dots` (and only one instruction `\Ldots` is necessary).

```
$D = \begin{pNiceMatrix}[nullify-dots]
h & i & j & k & l & m \\
x & \Ldots & & & & x \\
\end{pNiceMatrix}$
```

$$D = \begin{pmatrix} h & i & j & k & l & m \\ x & \dots & & & & x \end{pmatrix}$$

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

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.

```
$\begin{pNiceMatrix}
1 & 2 & 3 & 4 & 5 \\
1 & \Hdotsfor{3} & & & 5 \\
1 & 2 & 3 & 4 & 5 \\
1 & 2 & 3 & 4 & 5 \\
\end{pNiceMatrix}$
```

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & \dots & & & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$$

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

```
$\begin{pNiceMatrix}
1 & 2 & 3 & 4 & 5 \\
& \Hdotsfor{3} \\
1 & 2 & 3 & 4 & 5 \\
1 & 2 & 3 & 4 & 5 \\
\end{pNiceMatrix}$
```

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ & \dots & & & \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$$

Remark: Unlike the command `\hdotsfor` of `amsmath`, the command `\Hdotsfor` may be used when the package `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`.⁸

- 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.

⁸The options `renew-dots`, `renew-matrix` and `transparent` can be fixed with the command `\NiceMatrixOptions` 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.)

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}
```

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

3.4 The labels of the dotted lines

The commands `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots` and `\Hdotsfor` (and the command `\line` in the `code-after` which is described p. 7) accept two optional arguments specified by the tokens `_` and `^` for labels positionned below and above the line. The arguments are composed in math mode with `\scriptstyle`.

```
$\begin{bNiceMatrix}
1 & \hspace*{1cm} & & 0 \\
& \Ddots^{n \text{ times}} & & \\
0 & & & 1 \\
\end{bNiceMatrix}$
```

$$\begin{bmatrix} 1 & & & 0 \\ & \ddots^{n \text{ times}} & & \\ 0 & & & 1 \end{bmatrix}$$

3.5 Customization of the dotted lines

The dotted lines drawn by `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots` and `\Hdotsfor` (and by the command `\line` in the `code-after` which is described p. 7) may be customized by three options (specified between square brackets after the command):

- `color`;
- `shorten`;
- `line-style`.

These options may also be fixed with `\NiceMatrixOptions` or at the level of a given environment but, in those cases, they must be prefixed by `xdots`, and, thus have for names:

- `xdots/color`;
- `xdots/shorten`;
- `xdots/line-style`.

For the clarity of the explanations, we will use those names.

The option `xdots/color`

The option `xdots/color` fixes the color of the dotted line. However, one should remark that the dotted lines drawn in the exterior rows and columns have a special treatment: cf. p. 10.

The option `xdots/shorten`

The option `xdots/shorten` fixes the margin of both extremities of the line. The name is derived from the options “`shorten >`” and “`shorten <`” of Tikz but one should notice that `nicematrix` only provides `xdots/shorten`. The initial value of this parameter is 0.3 em (it is recommended to use a unit of length dependent of the current font).

The option `xdots/line-style`

It should be pointed that, by default, the lines drawn by Tikz with the parameter `dotted` are composed of square dots (and not rounded ones).⁹

⁹The first reason of this behaviour is that the PDF format includes a description for dashed lines. The lines specified with this descriptor are displayed very efficiently by the PDF readers. It's easy, starting from these dashed lines, to create a line composed by square dots whereas a line of rounded dots needs a specification of each dot in the PDF file.

```
\tikz \draw [dotted] (0,0) -- (5,0) ;
```

In order to provide lines with rounded dots in the style of those provided by `\ldots` (at least with the *Computer Modern* fonts), the package `nicematrix` embeds its own system to draw a dotted line (and this system uses PGF and not Tikz). This style is called `standard` and that's the initial value of the parameter `xdots/line-style`.

However (when Tikz is loaded) it's possible to use for `xdots/line-style` any style provided by Tikz, that is to say any sequence of options provided by Tikz for the Tikz pathes (with the exception of “color”, “shorten >” and “shorten <”).

Here is for example a tridiagonal matrix with the style `loosely dotted`:

```
\begin{pNiceMatrix}[nullify-dots,xdots/line-style=loosely dotted]
a      & b      & 0      & & & \Cdots & 0      & \\
b      & a      & b      & & \Ddots & & \Vdots & \\
0      & b      & a      & & \Ddots & & & \\
      & & \Ddots & & \Ddots & & \Ddots & \\
\Vdots & & & & & & & 0      & \\
0      & \Cdots & & & 0      & & b      & a      & \\
\end{pNiceMatrix}
```

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

4 The PGF/Tikz nodes created by `nicematrix`

The package `nicematrix` creates a PGF/Tikz node for each (non-empty) 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 (if Tikz has been loaded¹⁰). 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}
```

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

```
\tikz[remember picture,overlay]
\draw (mymatrix-2-2) circle (2mm) ;
```

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

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

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

In fact, the package `nicematrix` can create “extra nodes”: the “medium nodes” and the “large nodes”. The first ones are created with the option `create-medium-nodes` and the second ones with the option `create-large-nodes`.¹¹

¹⁰We remind that, since the version 3.13, `nicematrix` doesn't load Tikz by default by only PGF (Tikz is a layer over PGF).

¹¹There is also an option `create-extra-nodes` which is an alias for the conjunction of `create-medium-nodes` and `create-large-nodes`.

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

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

The names of the “large nodes” are constructed by adding the suffix “-large” to the names of the “normal nodes”. In the following example, we have underlined the “large nodes”. We consider that this example is self-explanatory.¹²

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

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`.¹³

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

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. 23).

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 thus after the construction of all the nodes).

If Tikz is loaded¹⁴, one may access to that nodes with classical Tikz instructions. The nodes should be designed as *i-j* (without the prefix corresponding to the name of the environment).

Moreover, a special command, called `\line`, is available to draw directly dotted lines between nodes. It may be used, for example, to draw a dotted line between two adjacents cells.

¹²There is no “large nodes” created in the exterior rows and columns (for these rows and columns, cf. p. 10).

¹³The 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.

¹⁴We remind that, since the version 3.13, `nicematrix` doesn’t load Tikz by default but only PGF (Tikz is a layer over PGF).

```

\NiceMatrixOptions{xdots/shorten = 0.6 em}
\begin{pNiceMatrix}[code-after=\line{2-2}{3-3}]
I      & 0      & \Cdots & 0      & \\
0      & I      & \Ddots & \Vdots & \\
\Vdots & \Ddots & I      & 0      & \\
0      & \Cdots & 0      & I      & \\
\end{pNiceMatrix}

```

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

For the readability of the code, an alternative syntax is provided: it's possible to give the instructions of the `\code-after` at the end of the environment, after the keyword `\CodeAfter`.

```

\NiceMatrixOptions{xdots/shorten = 0.6 em}
\begin{pNiceMatrix}
I      & 0      & \Cdots & 0      & \\
0      & I      & \Ddots & \Vdots & \\
\Vdots & \Ddots & I      & 0      & \\
0      & \Cdots & 0      & I      & \\
\CodeAfter
\line{2-2}{3-3}
\end{pNiceMatrix}

```

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

6 The environment `{NiceArray}` and its variants

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 R¹⁵ instead of l, c and r. It's possible to use the constructions `w{...}{...}`, `W{...}{...}`, `|`, `>{...}`, `<{...}`, `@{...}`, `!{...}` and `*{n}{...}` but the letters p, m and b should not be used.¹⁶

The package `nicematrix` provides also the variants `{pNiceArray}`, `{vNiceArray}`, `{VNiceArray}`, `{bNiceArray}` and `{BNiceArray}`.

Of course, one of the benefits of `{pNiceArray}` over `{pNiceMatrix}` is the possibility of drawing vertical rules:

```

$\left[\begin{NiceArray}{CCCC|C}
a_1    & ?      & \Cdots & ?      & ?      & \\
0      & & \Ddots & \Vdots & \Vdots & \\
\Vdots & \Ddots & \Ddots & ?      & & \\
0      & \Cdots & 0      & a_n    & & \\
\end{NiceArray}\right]$

```

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

Another benefit is the possibility of using different alignments of columns.

```

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

```

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

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 or asymmetrical delimiters.

¹⁵The 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.

¹⁶In a command `\multicolumn`, one should also use the letters L, C, R.


```

 $\begin{NiceArrayWithDelims}$ 
   $\{\downarrow\uparrow\}{CCC}[margin]$ 
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9 \\
 $\end{NiceArrayWithDelims}$ 

```

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

7 The vertical position of the arrays

The package `nicematrix` provides a option `baseline` for the vertical position of the arrays. This option takes as value an integer which is the number of the row on which the array will be aligned.

```

 $A = \begin{pNiceMatrix}[baseline=2]$ 
 $\frac{1}{\sqrt{1+p^2}}$  &  $p$  &  $1-p$  \\
1 & 1 & 1 \\
1 & p & 1+p \\
 $\end{pNiceMatrix}$ 

```

$$A = \begin{pmatrix} \frac{1}{\sqrt{1+p^2}} & p & 1-p \\ 1 & 1 & 1 \\ 1 & p & 1+p \end{pmatrix}$$

It's also possible to use the option `baseline` with one of the special values `t`, `c` or `b`. These letters may also be used absolutely like the option of the environment `{array}` of `array`. The initial value of `baseline` is `c`.

In the following example, we use the option `t` (equivalent to `baseline=t`) immediately after an `\item` of list. One should remark that the presence of a `\hline` at the beginning of the array doesn't prevent the alignment of the baseline with the baseline of the first row (with `{array}` of `array`, one must use `\firsthline`¹⁷).

```

 $\begin{enumerate}$ 
 $\item$  an item
 $\smallskip$ 
 $\item$   $\renewcommand{\arraystretch}{1.2}$ 
 $\begin{NiceArray}[t]{LCCCCC}$ 
 $\hline$ 
 $n$  & 0 & 1 & 2 & 3 & 4 & 5 \\
 $u_n$  & 1 & 2 & 4 & 8 & 16 & 32 \\
 $\hline$ 
 $\end{NiceArray}$ 
 $\end{enumerate}$ 

```

1. an item

$$\begin{array}{rcccccc} 2. & n & 0 & 1 & 2 & 3 & 4 & 5 \\ & u_n & 1 & 2 & 4 & 8 & 16 & 32 \end{array}$$

However, it's also possible to use the tools of `booktabs`: `\toprule`, `\bottomrule` and `\midrule`.

```

 $\begin{enumerate}$ 
 $\item$  an item
 $\smallskip$ 
 $\item$ 
 $\begin{NiceArray}[t]{LCCCCC}$ 
 $\toprule$ 
 $n$  & 0 & 1 & 2 & 3 & 4 & 5 \\
 $\midrule$ 
 $u_n$  & 1 & 2 & 4 & 8 & 16 & 32 \\
 $\bottomrule$ 
 $\end{NiceArray}$ 
 $\end{enumerate}$ 

```

1. an item

$$\begin{array}{rcccccc} 2. & n & 0 & 1 & 2 & 3 & 4 & 5 \\ & u_n & 1 & 2 & 4 & 8 & 16 & 32 \end{array}$$

¹⁷It's also possible to use `\firsthline` with `{NiceArray}`.

8 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” (exterior) has the number 0 (and not 1). Idem for the potential “first column”.

```


$$\begin{pmatrix} & C_1 & & \cdots & & C_4 & \\ L_1 & a_{11} & a_{12} & a_{13} & a_{14} & L_1 & \\ \vdots & a_{21} & a_{22} & a_{23} & a_{24} & \vdots & \\ & a_{31} & a_{32} & a_{33} & a_{34} & & \\ L_4 & a_{41} & a_{42} & a_{43} & a_{44} & L_4 & \\ & C_1 & & \cdots & & C_4 & \end{pmatrix}$$


```

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

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: they will automatically (and necessarily) be of type `R` for the first column and `L` for the last one.
- One may wonder how `nicematrix` determines the number of rows and columns which are needed for the composition of the “last row” and “last column”.
 - For the environments with explicit preamble, like `{NiceArray}` and `{pNiceArray}`, the number of columns can obviously be computed from the preamble.
 - When the option `light-syntax` (cf. p. 17) is used, `nicematrix` has, in any case, to load the whole body of the environment (and that’s why it’s not possible to put verbatim material in the array with the option `light-syntax`). The analysis of this whole body gives the number of rows (but not the number of columns).
 - In the other cases, `nicematrix` compute the number of rows and columns during the first compilation and write the result in the `aux` file for the next run.

However, it's possible to provide the number of the last row and the number of the last column as values of the options `last-row` and `last-col`, tending to an acceleration of the whole compilation of the document. That's what we will do throughout the rest of the document.

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

```
\NiceMatrixOptions{code-for-first-row = \color{red},
                  code-for-first-col = \color{blue},
                  code-for-last-row = \color{green},
                  code-for-last-col = \color{magenta}}
$\begin{pNiceArray}{CC|CC}[first-row,last-row=5,first-col,last-col,nullify-dots]
    & C 1 & & \Cdots & & C 4 & & \\
\end{pNiceArray}
```

```

L_1      & a_{11} & a_{12} & a_{13} & a_{14} & L_1      \\
\Vdots & a_{21} & a_{22} & a_{23} & a_{24} & \Vdots \\
\hline
      & a_{31} & a_{32} & a_{33} & a_{34} & \\
L_4      & a_{41} & a_{42} & a_{43} & a_{44} & L_4      \\
      & C_1      & \Cdots &      & C_4      & \\
\end{pNiceArray}$

```

$$\begin{array}{c}
\textcolor{red}{C_1} \dots \dots \dots \textcolor{red}{C_4} \\
\textcolor{blue}{L_1} \left(\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array} \right) \textcolor{magenta}{L_1} \\
\textcolor{blue}{L_4} \left(\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array} \right) \textcolor{magenta}{L_4} \\
\textcolor{green}{C_1} \dots \dots \dots \textcolor{green}{C_4}
\end{array}$$

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.¹⁸
If one wishes to define new specifiers for columns in order to draw vertical rules (for example thicker than the standard rules), he should consider the command `\OnlyMainNiceMatrix` described on page 17.
- A specification of color present in `code-for-first-row` also applies to a dotted line draw in this exterior “first row” (excepted if a value has been given to `xdots/color`). Idem for the other exterior rows and columns.
- Logically, the potential option `columns-width` (described p. 12) 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).

9 The dotted lines to separate rows or columns

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

```

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

```

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

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

```

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

```

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

¹⁸The latter is not true when the package `arydshln` is loaded besides `nicematrix`. In fact, `nicematrix` and `arydshln` are not totally compatible because `arydshln` redefines many internals of `array`. On another hand, if one really wants a vertical rule running in the first and in the last row, he should use `!\vline` instead of | in the preamble of the array.

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

```

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

```

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

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 package `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 = I}
\arrayrulecolor{blue}
\left(\begin{NiceArray}{C|C:C|C}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
\end{NiceArray}\right)
\arrayrulecolor{black}

```

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

We have used the command `\arrayrulecolor` (de `colortbl`) to draw in blue the three rules.

Remark : In the package `array` (on which the package `nicematrix` relies), horizontal and vertical rules make the array larger or wider by a quantity equal to the width of the rule¹⁹. In `nicematrix`, the dotted lines drawn by `\hdottedline` and “:” do likewise.

10 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`. In the environments of `nicematrix`, the cells of such columns are composed in mathematical mode, whereas, in `{array}` of `array`, they are composed in text mode.

```

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

```

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

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

```

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

```

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

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

¹⁹In fact, this is true only for `\hline` and “|” but not for `\cline`.

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.²⁰

```

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

```

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

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

```

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

```

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

But it's also possible to fix a zone where all the matrices will have their columns of the same width, equal to the widest cell of all the matrices. This construction uses the environment `{NiceMatrixBlock}` with the option `auto-columns-width`²¹. The environment `{NiceMatrixBlock}` has no direct link with the command `\Block` presented just below (cf. p. 13).

```

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

```

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

Several compilations may be necessary to achieve the job.

11 Block matrices

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

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

The command `\Block` must be used in the upper leftmost cell of the array with two arguments. The first argument is the size of the block with the syntax `i-j` where `i` is the number of rows of the block and `j` its number of columns. The second argument is the content of the block (composed in math mode). A Tikz node corresponding to the merged cells is created with the name “`i-j-block`”. If the user has required the creation of the “medium nodes”, a node of this type is also created with a name suffixed by `-medium`.

In the following examples, we use the command `\arrayrulecolor` of `colortbl`.

²⁰The 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).

²¹At this time, this is the only usage of the environment `{NiceMatrixBlock}` but it may have other usages in the future.

```

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

```

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

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

```

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

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

12 Advanced features

12.1 Aligment option in NiceMatrix

The environments without preamble (`{NiceMatrix}`, `{pNiceMatrix}`, `{bNiceMatrix}`, etc.) provide two options `l` and `r` (equivalent at L and R) which generate all the columns aligned leftwards (or rightwards).²²

```

$\begin{bNiceMatrix}[R]
\cos x & - \sin x \\
\sin x & \cos x
\end{bNiceMatrix}$

```

$$\begin{bmatrix} \cos x & -\sin x \\ \sin x & \cos x \end{bmatrix}$$

12.2 The command `\rotate`

The package `nicematrix` provides a command `\rotate`. When used in the beginning of a cell, this command composes the contents of the cell after a rotation of 90° in the direct sens.

In the following command, we use that command in the `code-for-first-row`.

```

\NiceMatrixOptions%
{code-for-first-row = \scriptstyle \rotate \text{image of },
code-for-last-col = \scriptstyle }
$A = \begin{pNiceMatrix}[first-row,last-col=4]
e_1 & e_2 & e_3 & \\
1 & 2 & 3 & e_1 \\
4 & 5 & 6 & e_2 \\
7 & 8 & 9 & e_3 \\
\end{pNiceMatrix}$

```

$$A = \begin{pmatrix} \text{image of } e_1 & \text{image of } e_2 & \text{image of } e_3 \\ 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \begin{matrix} e_1 \\ e_2 \\ e_3 \end{matrix}$$

²²This is a part of the functionality provided by the environments `{pmatrix*}`, `{bmatrix*}`, etc. of `mathtools`.

If the command `\rotate` is used in the “last row” (exterior to the matrix), the corresponding elements are aligned upwards as shown below.

```
\NiceMatrixOptions%
{code-for-last-row = \scriptstyle \rotate ,
 code-for-last-col = \scriptstyle }
$A = \begin{pNiceMatrix}[last-row=4,last-col=4]
1 & 2 & 3 & e_1 \\
4 & 5 & 6 & e_2 \\
7 & 8 & 9 & e_3 \\
\text{image of } e_1 & e_2 & e_3 \\
\end{pNiceMatrix}$
```

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \begin{matrix} e_1 \\ e_2 \\ e_3 \end{matrix}$$

image of e_1 e_2 e_3

12.3 The option `small`

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

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

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

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 package `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.

12.4 The counters `iRow` and `jCol`

In the cells of the array, it’s possible to use the LaTeX counters `iRow` and `jCol` which represent the number of the current row and the number of the current column²³. Of course, the user must not change the value of these counters which are used internally by `nicematrix`.

In the `code-after` (cf. p. 7), `iRow` represents the total number of rows (excepted the potential exterior rows) and `jCol` represents the total number of columns (excepted the potential exterior columns).

²³We recall that the exterior “first row” (if it exists) has the number 0 and that the exterior “first column” (if it exists) has also the number 0.

`\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 packages other than `nicematrix` (or by the user), they are shadowed in the environments of `nicematrix`.

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

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

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

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

12.5 The options `hlines`, `vlines` and `hvlines`

You can add horizontal rules between rows in the environments of `nicematrix` with the usual command `\hline` and you can use the specifier “|” to add vertical rules. However, by convenience, the package `nicematrix` also provides the option `hlines` (resp. `vlines`) which will draw all the horizontal (resp. vertical) rules (excepted, of course, the exterior rules corresponding to the exterior rows and columns). The key `hvlines` is an alias for the conjunction for the keys `hlines` et `vlines`.

In the following example, we use the command `\arrayrulecolor` of `colortbl`.

```
\arrayrulecolor{blue}
\begin{NiceArray}{CCCC}%
[hvlines,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}
\arrayrulecolor{black}
```

	<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>

However, there is a difference between the key `vlines` and the use of the specifier “|” in the preamble of the environment: the rules drawn by `vlines` completely cross the double-rules drawn by `\hline\hline`.

```
\begin{NiceArray}{CCCC}[vlines] \hline
a & b & c & d \\ \hline \hline
1 & 2 & 3 & 4 \\
1 & 2 & 3 & 4 \\ \hline
\end{NiceArray}
```

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>
1	2	3	4
1	2	3	4

For the environments with delimiters (for example `\pNiceArray` or `\pNiceMatrix`), the option `vlines` don't draw vertical rules on both sides, where are the delimiters (fortunately).


```
\setlength{\arrayrulewidth}{0.2pt}
$\begin{pNiceMatrix}[vlines]
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 2 & 3 & 4 & 5 & 6 \\
\end{pNiceMatrix}$
```

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

12.6 The option `light-syntax`

The option `light-syntax`²⁴ allows the user to compose the arrays with a lighter syntax, which gives a more readable TeX source.

When this option is used, one should use the semicolon for the end of a row and spaces or tabulations to separate the columns. However, as usual in the TeX world, the spaces after a control sequence are discarded and the elements between curly braces are considered as a whole.

The following example has been composed with XeLaTeX with `unicode-math`, which allows the use of greek letters directly in the TeX source.

```
$\begin{bNiceMatrix}[light-syntax,first-row,first-col]
{} a                b                ;
a 2\cos a            {\cos a + \cos b} ;
b \cos a + \cos b    { 2 \cos b }
\end{bNiceMatrix}$
```

$$\begin{matrix} & a & b \\ a & \begin{bmatrix} 2 \cos a & \cos a + \cos b \end{bmatrix} \\ b & \begin{bmatrix} \cos a + \cos b & 2 \cos b \end{bmatrix} \end{matrix}$$

It's possible to change the character used to mark the end of rows with the option `end-of-row`. As said before, the initial value is a semicolon.

When the option `light-syntax` is used, it is not possible to put verbatim material (for example with the command `\verb`) in the cells of the array.²⁵

12.7 Use of the column type `S` of `siunitx`

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

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

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

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

13 Technical remarks

13.1 Definition of new column types

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

For example, one may wish to define a new column type `?` in order to draw a (black) heavy rule of width 1 pt. The following definition will do the job²⁶:

²⁴This option is inspired by the package `spalign` of Joseph Rabinoff.

²⁵The reason is that, when the option `light-syntax` is used, the whole content of the environment is loaded as a TeX argument to be analyzed. The environment doesn't behave in that case as a standard environment of LaTeX which only put TeX commands before and after the content.

²⁶The command `\vrule` is a TeX (and not LaTeX) command.

```
\newcolumnntype{?}{\OnlyMainNiceMatrix{\vrule width 1 pt}}
```

The heavy vertical rule won't extend in the exterior rows:

```
\begin{pNiceArray}{CC?CC}[first-row,last-row=3]
C_1 & C_2 & C_3 & C_4 \\
a & b & c & d \\
e & f & g & h \\
C_1 & C_2 & C_3 & C_4 \\
\end{pNiceArray}
```

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

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

13.2 The names of the PGF nodes created by `nicematrix`

We have said that, when a name is given to an environment of `nicematrix`, it's possible to access the PGF/Tikz nodes through this name (cf. p. 6).

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

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

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

13.3 Diagonal lines

By default, all the diagonal lines²⁷ of a same array are “parallelized”. That means that the first diagonal line is drawn and, then, the other lines are drawn parallel to the first one (by rotation around the left-most extremity of the line). That's why the position of the instructions `\Ddots` in the array can have a marked effect on the final result.

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

Example with parallelization (default):

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

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

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

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

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

The same example without parallelization:

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

²⁷We speak of the lines created by `\Ddots` and not the lines created by a command `\line` in `code-after`.

13.4 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 which only contains `\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 equal to zero is empty.
- 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`.

13.5 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 idea²⁸. The environment `{matrix}` of `amsmath` and its variants (`{pmatrix}`, `{vmatrix}`, etc.) of `amsmath` prefer to delete these spaces with explicit instructions `\hskip -\arraycolsep`²⁹. The package `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).

13.6 The class option `draft`

When the class option `draft` is used, the dotted lines are not drawn, for a faster compilation.

13.7 A technical problem with the argument of `\`

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

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

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

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

²⁸In 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).*

²⁹And not by inserting `@{}` on both sides of the preamble of the array. As a consequence, the length of the `\hline` is not modified and may appear too long, in particular when using square brackets

```

\begin{pNiceMatrix}
a & \frac{AB}{B} \\
\noalign{\kern2mm}
b & c
\end{pNiceMatrix}

```

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

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

```

\begin{pNiceMatrix}
a & \multicolumn{1C}{\frac{AB}{B}} \\
b & c
\end{pNiceMatrix}

```

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

13.8 Obsolete environments

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

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

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

Since the version 3.12, the only way to use these environments is loading `nicematrix` with the option `obsolete-environments`.

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

14 Examples

14.1 Dotted lines

A permutation matrix (as an example, we have raised the value of `xdots/shorten`).

```

$\begin{pNiceMatrix}[xdots/shorten=0.6em]
0 & 1 & 0 & & & \cdots & 0 & \\
\vdots & & & \ddots & & & \vdots & \\
& & & \ddots & & & & \\
& & & \ddots & & & 0 & \\
0 & 0 & & & & & 1 & \\
1 & 0 & & \cdots & & & 0 & \\
\end{pNiceMatrix}$

```

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

An example with `\Iddots` (we have raised again the value of `xdots/shorten`).

```

$\begin{pNiceMatrix}[xdots/shorten=0.9em]
1 & \cdots & 1 \\
\vdots & & 0 \\
& \Iddots & \Iddots & \vdots \\
1 & 0 & \cdots & 0 \\
\end{pNiceMatrix}$

```

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

An example with `\multicolumn`:

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

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

An example with `\Hdotsfor`:

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

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

An example for the resultant of two polynomials:

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

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

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

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

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

14.2 Dotted lines which are no longer dotted

The option `line-style` controls the style of the lines drawn by `\Ldots`, `\Cdots`, etc. Thus, it's possible with these commands to draw lines which are no longer dotted.

```
\NiceMatrixOptions
{nullify-dots,code-for-first-col = \color{blue},code-for-first-col=\color{blue}}
$\begin{pNiceMatrix}[first-row,first-col]
    & & \Ldots[line-style={solid,<->},shorten=0pt]^{n \text{ columns}} \\\
    & 1 & 1 & 1 & \Ldots & 1 \\\
    & 1 & 1 & 1 & & 1 \\\
\Vdots[line-style={solid,<->}]_{n \text{ rows}} & 1 & 1 & 1 & & 1 \\\
    & 1 & 1 & 1 & & 1 \\\
    & 1 & 1 & 1 & \Ldots & 1 \\\
\end{pNiceMatrix}$
```

$$\begin{array}{c} \xrightarrow{n \text{ columns}} \\ \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 1 & 1 & & 1 \\ 1 & 1 & 1 & & 1 \\ 1 & 1 & 1 & & 1 \\ 1 & 1 & 1 & \dots & 1 \end{pmatrix} \\ \xleftarrow{n \text{ rows}} \end{array}$$

14.3 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,light-syntax}
\setlength{\extrarowheight}{1mm}
$\begin{pNiceArray}{CCCC:C}[last-col]
  1 1 1 1 1 ;
  2 4 8 16 9 ;
  3 9 27 81 36 ;
  4 16 64 256 100
\end{pNiceArray}$
\medskip
$\begin{pNiceArray}{CCCC:C}[last-col]
  1 1 1 1 1 ;
  0 2 6 14 7 { L_2 \gets -2 L_1 + L_2 } ;
  0 6 24 78 33 { L_3 \gets -3 L_1 + L_3 } ;
  0 12 60 252 96 { L_4 \gets -4 L_1 + L_4 }
\end{pNiceArray}$
...
\end{NiceMatrixBlock}

```

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

14.4 How to highlight cells of the matrix

The following examples require Tikz (by default, nicematrix only loads PGF) and the Tikz library `fit`. The following lines in the preamble of your document may do the job:

```

\usepackage{tikz}
\usetikzlibrary{fit}

```

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”).

We redraw the nodes with other nodes by using the Tikz library `fit`. Since we want to redraw the nodes exactly, we have to set `inner sep = 0 pt` (if we don't do that, the new nodes will be larger than the nodes created by `nicematrix`).

```

$\begin{pNiceArray}{>\{\strut\}CCCC}[create-large-nodes,margin,extra-margin = 2pt]
  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}
\CodeAfter

```

```

\begin{tikzpicture}[name suffix = -large,
    every node/.style = {draw,inner sep = 0 pt}]
    \node [fit = (1-1)] {} ;
    \node [fit = (2-2)] {} ;
    \node [fit = (3-3)] {} ;
    \node [fit = (4-4)] {} ;
\end{tikzpicture}
\end{pNiceArray}$

```

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

We should remark that the rules we have drawn are drawn *after* the construction of the array and thus, they don't spread the cells of the array. We recall that, on the other side, the command `\hline`, the specifier “|” and the options `hlines` and `vlines` spread the cells (when the package `array` is loaded but, when the package `nicematrix` is loaded, `array` is always loaded).³⁰

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`.

```

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

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

```

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

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

```

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

```

³⁰On the other side, the command `\cline` doesn't spread the rows of the array.

We recall that, for a rectangle of merged cells (with the command `\Block`), a Tikz node is created for the set of merged cells with the name i - j -block where i and j are the number of the row and the number of the column of the upper left cell (where the command `\Block` has been issued). If the user has required the creation of the `medium` nodes, a node of this type is also created with a name suffixed by `-medium`.

```

 $\begin{pNiceMatrix}[margin,create-medium-nodes]
  \Block{3-3}<\Large>\{A\} \& \& \& 0 \\\
  \& \hspace*{1cm} \& \& \Vdots \\\
  \& \& \& 0 \\\
  0 \& \Cdots \& 0 \& 0
\CodeAfter
  \tikz \node [highlight = (1-1-block-medium)] {};
\end{pNiceMatrix}$ 

```

$$\left(\begin{array}{c|c} \text{A} & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} \\ \hline 0 \dots\dots\dots 0 & 0 \end{array} \right)$$

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

```

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

```

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

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

```

\tikzset{mes-options/.style={remember picture,
                             overlay,
                             name prefix = exemple-,
                             highlight/.style = {fill = red!15,
                                                  blend mode = multiply,
                                                  inner sep = 0pt,
                                                  fit = #1}}}

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

```

We obtain the following matrix.

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

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

```
\begin{pNiceArray}>{\strut}CCCC[create-large-nodes,margin,extra-margin=2pt]
  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}
\CodeAfter
  \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) ;
\end{pNiceArray}
```

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

14.5 Direct use of the Tikz nodes

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

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

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

\NiceMatrixOptions{nullify-dots}
```

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

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

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

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

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_{1n} \\
& \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 = (A-3-1) (A-3-5) ] {} ;
\node [highlight = (B-1-3) (B-5-3) ] {} ;
\draw [color = gray] (A-3-3) to [bend left] (B-3-3) ;
\end{tikzpicture}

```

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

15 Implementation

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

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

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

Of course, the package `nicematrix` uses the features of the package `array`. It tries to be 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`.

Declaration of the package and packages loaded

The prefix `nicematrix` has been registered for this package.

See: <http://mirrors.ctan.org/macros/latex/contrib/l3kernel/l3prefixes.pdf>

<@@=nicematrix>

First, we load `pgfcore` and the module `shapes`. We do so because it's not possible to use `\usepgfmodule` in `\ExplSyntaxOn`.

```
1 \RequirePackage{pgfcore}
2 \usepgfmodule{shapes}
```

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

```
3 \RequirePackage{l3keys2e}
4 \ProvidesExplPackage
5   {nicematrix}
6   {\myfiledate}
7   {\myfileversion}
8   {Mathematical matrices with PGF/TikZ}
```

The version of 2020/02/08 of `expl3` has replaced `\l_keys_key_tl` by `\l_keys_key_str`. We have immediately changed in this file. Now, you test the existence of `\l_keys_key_str` in order to detect whether the version of LaTeX used by the final user is up to date.

```
9 \msg_new:nnn { nicematrix } { expl3-too-old }
10 {
11   Your-version-of-LaTeX-(especially-expl3)-is-too-old.~
12   You-can-go-on-but-you-will-probably-have-other-errors~
13   if-you-use-the-functionalities-of-nicematrix.
14 }
15 \cs_if_exist:Nf \l_keys_key_str
16 { \msg_error:nn { nicematrix } { expl3-too-old } }
```

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

```
17 \bool_new:N \c_@@_draft_bool
18 \DeclareOption { draft } { \bool_set_true:N \c_@@_draft_bool }
19 \DeclareOption* { }
20 \ProcessOptions \relax
```

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

We load some packages.

```
21 \RequirePackage { array }
22 \RequirePackage { amsmath }
23 \RequirePackage { xparse }

24 \cs_new_protected:Npn \@@_error:n { \msg_error:nn { nicematrix } }
25 \cs_new_protected:Npn \@@_error:nn { \msg_error:nnn { nicematrix } }
26 \cs_new_protected:Npn \@@_error:nnn { \msg_error:nnnn { nicematrix } }
27 \cs_new_protected:Npn \@@_fatal:n { \msg_fatal:nn { nicematrix } }
28 \cs_new_protected:Npn \@@_fatal:nn { \msg_fatal:nnn { nicematrix } }
29 \cs_new_protected:Npn \@@_msg_new:nn { \msg_new:nnn { nicematrix } }
30 \cs_new_protected:Npn \@@_msg_new:nnn { \msg_new:nnnn { nicematrix } }

31 \cs_new_protected:Npn \@@_msg_redirect_name:nn
32 { \msg_redirect_name:nnn { nicematrix } }
```

Technical definitions

```

33 \bool_new:N \c_@@_tikz_loaded_bool
34 \AtBeginDocument
35 {
36     \@ifpackageloaded { tikz }
37     {

```

In some constructions, we will have to use a `{pgfpicture}` which *must* be replaced by a `{tikzpicture}` if Tikz is loaded. However, this switch between `{pgfpicture}` and `{tikzpicture}` can't be done dynamically with a conditional because, when the Tikz library `external` is loaded by the user, the pair `\tikzpicture-\endtikzpicture` (or `\begin{tikzpicture}-\end{tikzpicture}`) must be statically “visible” (even when externalization is not activated).

That's why we create `\c_@@_pgfortikzpicture_tl` and `\c_@@_endpgfortikzpicture_tl` which will be used to construct in a `\AtBeginDocument` the correct version of some commands.

```

38     \bool_set_true:N \c_@@_tikz_loaded_bool
39     \tl_const:Nn \c_@@_pgfortikzpicture_tl { \exp_not:N \tikzpicture }
40     \tl_const:Nn \c_@@_endpgfortikzpicture_tl { \exp_not:N \endtikzpicture }
41 }
42 {
43     \tl_const:Nn \c_@@_pgfortikzpicture_tl { \exp_not:N \pgfpicture }
44     \tl_const:Nn \c_@@_endpgfortikzpicture_tl { \exp_not:N \endpgfpicture }
45 }
46 }

```

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.

```

47 \bool_new:N \c_@@_revtex_bool
48 \@ifclassloaded { revtex4-1 }
49 { \bool_set_true:N \c_@@_revtex_bool }
50 { }
51 \@ifclassloaded { revtex4-2 }
52 { \bool_set_true:N \c_@@_revtex_bool }
53 { }

```

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

```

54 \@@_msg_new:nn { Draft-mode }
55 { The~compilation-is-in~draft-mode:-the~dotted-lines~won't~be~drawn. }
56 \bool_if:NT \c_@@_draft_bool { \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.

```

57 \ProvideDocumentCommand \iddots { }
58 {
59     \mathinner
60     {
61         \tex_mkern:D 1 mu
62         \box_move_up:nn { 1 pt } { \hbox:n { . } }
63         \tex_mkern:D 2 mu
64         \box_move_up:nn { 4 pt } { \hbox:n { . } }
65         \tex_mkern:D 2 mu
66         \box_move_up:nn { 7 pt }
67         { \vbox:n { \kern 7 pt \hbox:n { . } } }
68         \tex_mkern:D 1 mu
69     }
70 }

```

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.

```

71 \int_new:N \g_@@_env_int
72 \cs_new:Npn \@@_env: { nm - \int_use:N \g_@@_env_int }
73 \cs_new_protected:Npn \@@_qpoint: #1
74 { \pgfpointanchor { \@@_env: - #1 } { center } }

```

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

```

75 \int_new:N \g_@@_NiceMatrixBlock_int

```

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

```

76 \dim_new:N \l_@@_columns_width_dim

```

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

```

77 \seq_new:N \g_@@_names_seq

```

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

```

78 \bool_new:N \l_@@_in_env_bool

```

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

```

79 \bool_new:N \l_@@_NiceArray_bool

```

```

80 \cs_new_protected:Npn \@@_test_if_math_mode:
81 {
82   \if_mode_math: \else:
83     \@@_fatal:n { Outside~math~mode }
84   \fi:
85 }

```

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

```

86 \bool_new:N \c_@@_colortbl_loaded_bool
87 \AtBeginDocument
88 {
89   \@ifpackageloaded { colortbl }
90   {
91     \bool_set_true:N \c_@@_colortbl_loaded_bool
92     \cs_set_protected:Npn \@@_vline_i: { { \CT@arc@ \vline } }
93   }
94   { }
95 }
96 \colorlet { nicematrix-last-col } { . }
97 \colorlet { nicematrix-last-row } { . }

```

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.

```

98 \dim_new:N \l_@@_inter_dots_dim
99 \dim_set:Nn \l_@@_inter_dots_dim { 0.45 em }

```

The length \l_@@_xdots_shorten_dim is the minimal distance between a node (in fact an anchor of that node) and a dotted line (we say “minimal” because, by definition, a dotted line is not a continuous line and, therefore, this distance may vary a little).

```

100 \dim_new:N \l_@@_xdots_shorten_dim
101 \dim_set:Nn \l_@@_xdots_shorten_dim { 0.3 em }

```

The length `\l_@@_radius_dim` is the radius of the dots for the dotted lines (for `\hdottedline` and `\dottedline` and for all the other dotted lines when `line-style` is equal to `standard`, which is the initial value). The initial value is 0.53 pt but it will be changed if the option `small` is used (to 0.37 pt).

```

102 \dim_new:N \l_@@_radius_dim
103 \dim_set:Nn \l_@@_radius_dim { 0.53 pt }

```

The name of the current environment or the current command (despite the name which contains *env*).

```

104 \str_new:N \g_@@_name_env_str

```

The string `\g_@@_com_or_env_str` will contain the word *command* or *environment* whether we are in a command of `nicematrix` or a an environment of `nicematrix`. The default value is *environment*.

```

105 \str_new:N \g_@@_com_or_env_str
106 \str_set:Nn \g_@@_com_or_env_str { environment }

```

The following control sequence will be able to reconstruct the full name of the current command or environment (despite the name which contains *env*). This command must *not* be protected since it's used in error messages.

```

107 \cs_new:Npn \@@_full_name_env:
108 {
109     \str_if_eq:VnTF \g_@@_com_or_env_str { command }
110     { command \space \c_backslash_str \g_@@_name_env_str }
111     { environment \space \{ \g_@@_name_env_str \} }
112 }

```

```

113 \tl_new:N \g_@@_internal_code_after_tl
114 \tl_new:N \g_@@_code_after_tl

```

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

```

115 \int_new:N \l_@@_save_iRow_int
116 \int_new:N \l_@@_save_jCol_int

```

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

```

117 \bool_new:N \g_@@_row_of_col_done_bool
118 \tl_new:N \l_@@_initial_suffix_tl
119 \tl_new:N \l_@@_initial_anchor_tl
120 \tl_new:N \l_@@_final_suffix_tl
121 \tl_new:N \l_@@_final_anchor_tl
122 \dim_new:N \l_@@_x_initial_dim
123 \dim_new:N \l_@@_y_initial_dim
124 \dim_new:N \l_@@_x_final_dim
125 \dim_new:N \l_@@_y_final_dim
126 \dim_new:N \l_tmpc_dim
127 \dim_new:N \l_tmpd_dim
128 \bool_new:N \g_@@_empty_cell_bool

```

The token list `\l_@@_xdots_line_style_tl` corresponds to the option `tikz` of the commands `\Cdots`, `\Ldots`, etc. and of the options `line-style` for the environments and `\NiceMatrixOptions`. The constant `\c_@@_standard_tl` will be used in some tests.

```

129 \tl_new:N \l_@@_xdots_line_style_tl
130 \tl_const:Nn \c_@@_standard_tl { standard }
131 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl

```

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`.

```
132 \int_new:N \l_@@_first_row_int
133 \int_set:Nn \l_@@_first_row_int 1
```

- **First column**

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

```
134 \int_new:N \l_@@_first_col_int
135 \int_set:Nn \l_@@_first_col_int 1
```

- **Last row**

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

```
136 \int_new:N \l_@@_last_row_int
137 \int_set:Nn \l_@@_last_row_int { -2 }
```

If, in an environment like `{pNiceArray}`, the option `last-row` is used without value, we will globally raise the following flag. It will be used to know if we have, after the construction of the array, to write in the `aux` file the number of the “last row”.³¹

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

Idem for `\l_@@_last_col_without_value_bool`

```
139 \bool_new:N \l_@@_last_col_without_value_bool
```

- **Last column**

For the eventual “last column”, we use an integer. A value of `-2` means that there is no last column. A value of `-1` means that there is a last column but we don’t know its value because the user has used the option `last-col` without value (it’s possible in an environment without preamble like `{pNiceMatrix}`). A value of 0 means that the option `last-col` has been used in an environment with preamble (like `{pNiceArray}`).

```
140 \int_new:N \l_@@_last_col_int
141 \int_set:Nn \l_@@_last_col_int { -2 }
```

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

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

³¹We 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.

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.

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

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

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`.

```
143 \bool_new:N \c_@@_siunitx_loaded_bool
144 \AtBeginDocument
145 {
146   \ifpackageloaded { siunitx }
147     { \bool_set_true:N \c_@@_siunitx_loaded_bool }
148     { }
149 }
```

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

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

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

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

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

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

```
150 \cs_set_protected:Npn \@@_adapt_S_column:
151 {
152   \bool_if:NT \c_@@_siunitx_loaded_bool
153   {
154     \group_begin:
155     \@temptokena = { }
```

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

```
156 \cs_set_eq:NN \NC@find \prg_do_nothing:
157 \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`).

```
158 \tl_gset:NV \g_tmpa_tl \@temptokena
159 \group_end:
160 \tl_new:N \c_@@_table_collect_begin_tl
161 \tl_set:Nx \l_tmpa_tl { \tl_item:Nn \g_tmpa_tl 2 }
162 \tl_gset:Nx \c_@@_table_collect_begin_tl { \tl_item:Nn \l_tmpa_tl 1 }
163 \tl_new:N \c_@@_table_print_tl
164 \tl_gset:Nx \c_@@_table_print_tl { \tl_item:Nn \g_tmpa_tl { -1 } }
```

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

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

```
165 \cs_gset_eq:NN \@@_adapt_S_column: \prg_do_nothing:
166 }
167 }
```

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).

```
168 \cs_new_protected:Npn \@@_renew_NC@rewrite@S:
169 {
170   \renewcommand*{\NC@rewrite@S}[1][ ]
171   {
172     \@temptokena \exp_after:wN
173     {
174       \tex_the:D \@temptokena
175       > { \@@_Cell: \c_@@_table_collect_begin_tl S {##1} }
176       c
177       < { \c_@@_table_print_tl \@@_end_Cell: }
178     }
179     \NC@find
180   }
181 }
```

The following command is only for efficiency. It must *not* be protected because it will be used (for instance) in names of PGF nodes.

```
182 \cs_new:Npn \@@_succ:n #1 { \the \numexpr #1 + 1 \relax }
183 \cs_new:Npn \@@_pred:n #1 { \the \numexpr #1 - 1 \relax }
```

Command for creation of rectangle nodes

The following command should be used in a `{pgfpicture}`. It creates an rectangular (empty but with a name) when the four corners are given.

#1 is the name of the node which will be created; **#2** and **#3** are the coordinates of one of the corner of the rectangle; **#4** and **#5** are the coordinates of the opposite corner.

```
184 \cs_new_protected:Npn \@@_pgf_rect_node:nnnnn #1 #2 #3 #4 #5
185 {
186   \begin { pgfscope }
187   \pgfset
188   {
189     outer~sep = \c_zero_dim ,
190     inner~sep = \c_zero_dim ,
191     minimum~size = \c_zero_dim
192   }
193   \pgftransformshift { \pgfpoint { 0.5 * ( #2 + #4 ) } { 0.5 * ( #3 + #5 ) } }
194   \pgfnode
```

```

195     { rectangle }
196     { center }
197     {
198       \vbox_to_ht:nn
199       { \dim_abs:n { #5 - #3 } }
200       {
201         \vfill
202         \hbox_to_wd:nn { \dim_abs:n { #4 - #2 } } { }
203       }
204     }
205     { #1 }
206     { }
207   \end { pgfscope }
208 }

```

The command `\@@_pgf_rect_node:nnn` is a variant of `\@@_pgr_rect_node:nnnn`: it takes two PGF points as arguments instead of the four dimensions which are the coordinates.

```

209 \cs_new_protected:Npn \@@_pgf_rect_node:nnn #1 #2 #3
210 {
211   \begin { pgfscope }
212   \pgfset
213   {
214     outer~sep = \c_zero_dim ,
215     inner~sep = \c_zero_dim ,
216     minimum~size = \c_zero_dim
217   }
218   \pgftransformshift { \pgfpointscale { 0.5 } { \pgfpointadd { #2 } { #3 } } }
219   \pgfpointdiff { #3 } { #2 }
220   \pgfgetlastxy \l_tmpa_dim \l_tmpb_dim
221   \pgfnode
222   { rectangle }
223   { center }
224   {
225     \vbox_to_ht:nn
226     { \dim_abs:n \l_tmpb_dim }
227     { \vfill \hbox_to_wd:nn { \dim_abs:n \l_tmpa_dim } { } }
228   }
229   { #1 }
230   { }
231   \end { pgfscope }
232 }

```

The options

The boolean `\l_@@_light_syntax_bool` corresponds to the option `light-syntax`.

```

233 \bool_new:N \l_@@_light_syntax_bool

```

The token list `\l_@@_baseline_str` may contain one of the three values `t`, `c` or `b` as in the option of the environment `{array}`. However, it may also contain an integer (which represents the number of the row to which align the array).

```

234 \str_new:N \l_@@_baseline_str
235 \str_set:Nn \l_@@_baseline_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`).

```

236 \bool_new:N \l_@@_exterior_arraycolsep_bool

```

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

```
237 \bool_new:N \l_@@_parallelize_diags_bool
238 \bool_set_true:N \l_@@_parallelize_diags_bool
```

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

```
239 \bool_new:N \l_@@_hlines_bool
240 \bool_new:N \l_@@_vlines_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.).

```
241 \bool_new:N \l_@@_nullify_dots_bool
```

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

```
242 \bool_new:N \l_@@_auto_columns_width_bool
```

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

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

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

```
244 \bool_new:N \l_@@_medium_nodes_bool
245 \bool_new:N \l_@@_large_nodes_bool
```

The dimension `\l_@@_left_margin_dim` correspond to the option `left-margin`. Idem for the right margin. These parameters are involved in the creation of the “medium nodes” but also in the placement of the delimiters and the drawing of the horizontal dotted lines (`\hdottedline`).

```
246 \dim_new:N \l_@@_left_margin_dim
247 \dim_new:N \l_@@_right_margin_dim
```

The following dimensions will be used internally to compute the width of the potential “first column” and “last column”.

```
248 \dim_new:N \g_@@_width_last_col_dim
249 \dim_new:N \g_@@_width_first_col_dim
```

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

```
250 \dim_new:N \l_@@_extra_left_margin_dim
251 \dim_new:N \l_@@_extra_right_margin_dim
```

The token list `\l_@@_end_of_row_tl` corresponds to the option `end-of-row`. It specifies the symbol used to mark the ends of rows when the light syntax is used.

```
252 \tl_new:N \l_@@_end_of_row_tl
253 \tl_set:Nn \l_@@_end_of_row_tl { ; }
```

The following parameter is for the color the dotted lines drawn by `\Cdots`, `\Ldots`, `\Vdots`, `\Ddots`, `\Iddots` and `\Hdotsfor` but *not* the dotted lines drawn by `\hdottedline` and “:”.

```
254 \tl_new:N \l_@@_xdots_color_tl
```

Sometimes, we want to have several arrays vertically juxtaposed in order to have an alignment of the columns of these arrays. To achieve this goal, one may wish to use the same width for all the columns (for example with the option `columns-width` or the option `auto-columns-width` of the environment `{NiceMatrixBlock}`). However, even if we use the same type of delimiters, the width of the delimiters may be different from an array to another because the width of the delimiter is function of its size. That's why we create an option called `max-delimiter-width` which will give to the delimiters the width of a delimiter (of the same type) of big size. The following boolean corresponds to this option.

```
255 \bool_new:N \l_@@_max_delimiter_width_bool
```

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

```
256 \keys_define:nn { NiceMatrix / xdots }
257 {
258   line-style .code:n =
259   {
260     \bool_lazy_or:nnTF
```

We can't use `\c_@@_tikz_loaded_bool` to test whether `tikz` is loaded because `\NiceMatrixOptions` may be used in the preamble of the document.

```
261     { \cs_if_exist_p:N \tikzpicture }
262     { \str_if_eq_p:nn { #1 } { standard } }
263     { \tl_set:Nn \l_@@_xdots_line_style_tl { #1 } }
264     { \@@_error:n { bad~option~for~line~style } }
265   } ,
266   line-style .value_required:n = true ,
267   color .tl_set:N = \l_@@_xdots_color_tl ,
268   color .value_required:n = true ,
269   shorten .dim_set:N = \l_@@_xdots_shorten_dim ,
270   shorten .value_required:n = true ,
```

The options `down` and `up` are not documented for the final user because he should use the syntax with `^` and `_`.

```
271   down .tl_set:N = \l_@@_xdots_down_tl ,
272   up .tl_set:N = \l_@@_xdots_up_tl ,
273   unknown .code:n = \@@_error:n { Unknown~option~for~xdots }
274 }

275 \keys_define:nn { NiceMatrix / Global }
276 {
277   xdots .code:n = \keys_set:nn { NiceMatrix / xdots } { #1 } ,
278   max-delimiter-width .bool_set:N = \l_@@_max_delimiter_width_bool ,
279   light-syntax .bool_set:N = \l_@@_light_syntax_bool ,
280   light-syntax .default:n = true ,
281   end-of-row .tl_set:N = \l_@@_end_of_row_tl ,
282   end-of-row .value_required:n = true ,
283   code-for-first-col .tl_set:N = \l_@@_code_for_first_col_tl ,
284   code-for-first-col .value_required:n = true ,
285   code-for-last-col .tl_set:N = \l_@@_code_for_last_col_tl ,
286   code-for-last-col .value_required:n = true ,
287   code-for-first-row .tl_set:N = \l_@@_code_for_first_row_tl ,
288   code-for-first-row .value_required:n = true ,
289   code-for-last-row .tl_set:N = \l_@@_code_for_last_row_tl ,
290   code-for-last-row .value_required:n = true ,
291   small .bool_set:N = \l_@@_small_bool ,
292   hlines .bool_set:N = \l_@@_hlines_bool ,
293   vlines .bool_set:N = \l_@@_vlines_bool ,
294   hvlines .meta:n = { hlines , vlines } ,
295   parallelize-diags .bool_set:N = \l_@@_parallelize_diags_bool ,
```

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

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

```

297   renew-dots .value_forbidden:n = true ,
298   nullify-dots .bool_set:N = \l_@@_nullify_dots_bool ,

```

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

```

299   create-medium-nodes .bool_set:N = \l_@@_medium_nodes_bool ,
300   create-large-nodes .bool_set:N = \l_@@_large_nodes_bool ,
301   create-extra-nodes .meta:n =
302     { create-medium-nodes , create-large-nodes } ,
303   left-margin .dim_set:N = \l_@@_left_margin_dim ,
304   left-margin .default:n = \arraycolsep ,
305   right-margin .dim_set:N = \l_@@_right_margin_dim ,
306   right-margin .default:n = \arraycolsep ,
307   margin .meta:n = { left-margin = #1 , right-margin = #1 } ,
308   margin .default:n = \arraycolsep ,
309   extra-left-margin .dim_set:N = \l_@@_extra_left_margin_dim ,
310   extra-right-margin .dim_set:N = \l_@@_extra_right_margin_dim ,
311   extra-margin .meta:n =
312     { extra-left-margin = #1 , extra-right-margin = #1 } ,
313   extra-margin .value_required:n = true
314 }

```

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

```

315 \keys_define:nn { NiceMatrix / Env }
316 {

```

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

```

317   c .code:n = \str_set:Nn \l_@@_baseline_str c ,
318   t .code:n = \str_set:Nn \l_@@_baseline_str t ,
319   b .code:n = \str_set:Nn \l_@@_baseline_str b ,
320   baseline .tl_set:N = \l_@@_baseline_str ,
321   baseline .value_required:n = true ,
322   columns-width .code:n =
323     \str_if_eq:nnTF { #1 } { auto }
324     { \bool_set_true:N \l_@@_auto_columns_width_bool }
325     { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,
326   columns-width .value_required:n = true ,
327   name .code:n =

```

We test whether we are in the measuring phase of an environment of `amsmath` (always loaded by `nicematrix`) because we want to avoid a fallacious message of duplicate name in this case.

```

328   \legacy_if:nF { measuring@ }
329   {
330     \str_set:Nn \l_tmpa_str { #1 }
331     \seq_if_in:NVTF \g_@@_names_seq \l_tmpa_str
332     { \@@_error:nn { Duplicate-name } { #1 } }
333     { \seq_gput_left:N \g_@@_names_seq \l_tmpa_str }
334     \str_set_eq:NN \l_@@_name_str \l_tmpa_str
335   } ,
336   name .value_required:n = true ,
337   code-after .tl_gset:N = \g_@@_code_after_tl ,
338   code-after .value_required:n = true ,
339   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
340   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
341   last-row .int_set:N = \l_@@_last_row_int ,
342   last-row .default:n = -1 ,
343 }

```

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

```

344 \keys_define:nn { NiceMatrix }
345 {
346   NiceMatrixOptions .inherit:n =
347   {
348     NiceMatrix / Global ,
349   } ,
350   NiceMatrixOptions / xdots .inherit:n = NiceMatrix / xdots ,
351   NiceMatrix .inherit:n =
352   {
353     NiceMatrix / Global ,
354     NiceMatrix / Env ,
355   } ,
356   NiceMatrix / xdots .inherit:n = NiceMatrix / xdots ,
357   NiceArray .inherit:n =
358   {
359     NiceMatrix / Global ,
360     NiceMatrix / Env ,
361   } ,
362   NiceArray / xdots .inherit:n = NiceMatrix / xdots ,
363   pNiceArray .inherit:n =
364   {
365     NiceMatrix / Global ,
366     NiceMatrix / Env ,
367   } ,
368   pNiceArray / xdots .inherit:n = NiceMatrix / xdots
369 }

```

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

```

370 \keys_define:nn { NiceMatrix / NiceMatrixOptions }
371 {

```

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

```

372   renew-matrix .code:n = \@@_renew_matrix: ,
373   renew-matrix .value_forbidden:n = true ,
374   transparent .meta:n = { renew-dots , renew-matrix } ,
375   transparent .value_forbidden:n = true,

```

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

```

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

```

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

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

```

377   columns-width .code:n =
378     \str_if_eq:nnTF { #1 } { auto }
379     { \@@_error:n { Option-auto-for-columns-width } }
380     { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,

```

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

```

381   allow-duplicate-names .code:n =
382     \@@_msg_redirect_name:nn { Duplicate-name } { none } ,
383   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`).

```

384   letter-for-dotted-lines .code:n =
385   {
386     \int_compare:nTF { \tl_count:n { #1 } = 1 }
387     { \str_set:Nx \l_@@_letter_for_dotted_lines_str { #1 } }
388     { \@@_error:n { Bad~value~for~letter~for~dotted~lines } }
389   } ,
390   letter-for-dotted-lines .value_required:n = true ,
391   unknown .code:n = \@@_error:n { Unknown~key~for~NiceMatrixOptions }
392 }
393 \str_new:N \l_@@_letter_for_dotted_lines_str
394 \str_set_eq:NN \l_@@_letter_for_dotted_lines_str \c_colon_str

```

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

```

395 \NewDocumentCommand \NiceMatrixOptions { m }
396 { \keys_set:nn { NiceMatrix / NiceMatrixOptions } { #1 } }

```

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

```

397 \keys_define:nn { NiceMatrix / NiceMatrix }
398 {
399   last-col .code:n = \tl_if_empty:nTF {#1}
400   {
401     \bool_set_true:N \l_@@_last_col_without_value_bool
402     \int_set:Nn \l_@@_last_col_int { -1 }
403   }
404   { \int_set:Nn \l_@@_last_col_int { #1 } } ,
405   l .code:n = \tl_set:Nn \l_@@_type_of_col_tl L ,
406   r .code:n = \tl_set:Nn \l_@@_type_of_col_tl R ,
407   L .code:n = \tl_set:Nn \l_@@_type_of_col_tl L ,
408   R .code:n = \tl_set:Nn \l_@@_type_of_col_tl R ,
409   unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix }
410 }

```

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

```

411 \keys_define:nn { NiceMatrix / NiceArray }
412 {

```

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

```

413   last-col .code:n = \tl_if_empty:nF { #1 }
414   { \@@_error:n { last-col~non~empty~for~NiceArray } }
415   \int_zero:N \l_@@_last_col_int ,
416   unknown .code:n = \@@_error:n { Unknown~option~for~NiceArray }
417 }
418 \keys_define:nn { NiceMatrix / pNiceArray }
419 {
420   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
421   last-col .code:n = \tl_if_empty:nF {#1}
422   { \@@_error:n { last-col~non~empty~for~NiceArray } }
423   \int_zero:N \l_@@_last_col_int ,
424   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
425   last-row .int_set:N = \l_@@_last_row_int ,
426   last-row .default:n = -1 ,
427   unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix }
428 }

```


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}`).

```
429 \cs_new_protected:Npn \@@_Cell:
430 {
```

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

```
431 \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.

```
432 \int_compare:nNnT \c@jCol = 1
433 { \int_compare:nNnT \l_@@_first_col_int = 1 \@@_begin_of_row: }
434 \int_gset:Nn \g_@@_col_total_int { \int_max:nn \g_@@_col_total_int \c@jCol }
```

The content of the cell is composed in the box `\l_@@_cell_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).

```
435 \hbox_set:Nw \l_@@_cell_box
436 \c_math_toggle_token
437 \bool_if:NT \l_@@_small_bool \scriptstyle
```

We will call *corners* of the matrix the cases which are at the intersection of the exterior rows and exterior columns (of course, the four corners doesn't always exist simultaneously).

The codes `\l_@@_code_for_first_row_tl` and *al* don't apply in the corners of the matrix.

```
438 \int_compare:nNnTF \c@iRow = 0
439 {
440   \int_compare:nNnT \c@jCol > 0
441   {
442     \l_@@_code_for_first_row_tl
443     \xglobal \colorlet { nicematrix-first-row } { . }
444   }
445 }
446 {
447   \int_compare:nNnT \c@iRow = \l_@@_last_row_int
448   {
449     \l_@@_code_for_last_row_tl
450     \xglobal \colorlet { nicematrix-last-row } { . }
451   }
452 }
453 }
```

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

```
454 \cs_new_protected:Npn \@@_begin_of_row:
455 {
456   \int_gincr:N \c@iRow
457   \dim_gset_eq:NN \g_@@_dp_ante_last_row_dim \g_@@_dp_last_row_dim
458   \dim_gset:Nn \g_@@_dp_last_row_dim { \box_dp:N \@arstrutbox }
459   \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
460   \pgfpicture
461   \pgfrememberpicturepositiononpagetrue
462   \pgfcoordinate
463   { \@@_env: - row - \int_use:N \c@iRow - base }
464   \pgfpointorigin
465   \str_if_empty:NF \l_@@_name_str
466   {
467     \pgfnodealias
468     { \@@_env: - row - \int_use:N \c@iRow - base }
```

```

469         { \l_@@_name_str - row - \int_use:N \c@iRow - base }
470     }
471     \endpgfpicture
472 }

```

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. If the user uses the `last-row`, some lines will be dynamically added to this command.

```

473 \cs_new_protected:Npn \@@_update_for_first_and_last_row:
474 {
475     \int_compare:nNnTF \c@iRow = 0
476     {
477         \dim_gset:Nn \g_@@_dp_row_zero_dim
478         { \dim_max:nn \g_@@_dp_row_zero_dim { \box_dp:N \l_@@_cell_box } }
479         \dim_gset:Nn \g_@@_ht_row_zero_dim
480         { \dim_max:nn \g_@@_ht_row_zero_dim { \box_ht:N \l_@@_cell_box } }
481     }
482     {
483         \int_compare:nNnT \c@iRow = 1
484         {
485             \dim_gset:Nn \g_@@_ht_row_one_dim
486             { \dim_max:nn \g_@@_ht_row_one_dim { \box_ht:N \l_@@_cell_box } }
487         }
488     }
489 }
490 \cs_new_protected:Npn \@@_end_Cell:
491 {
492     \c_math_toggle_token
493     \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”).

```

494     \dim_gset:Nn \g_@@_max_cell_width_dim
495     { \dim_max:nn \g_@@_max_cell_width_dim { \box_wd:N \l_@@_cell_box } }

```

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

```

496     \@@_update_for_first_and_last_row:

```

If the cell is empty, or may be considered as if, we must not create the PGF node, for two reasons:

- it’s a waste of time since such a node would be rather pointless;
- we test the existence of these nodes in order to determine whether a cell is empty when we search the extremities of a dotted line.

However, it’s very difficult to determine whether a cell is empty. As of now, we use the following technic:

- if the width of the box `\l_@@_cell_box` (created with the content of the cell) is equal to zero, we consider the cell as empty (however, this is not perfect since the user may have use a `\rlap`, a `\llap` or a `\mathclap` of `mathtools`).
- the cells with a command `\Ldots` or `\Cdots`, `\Vdots`, etc., should also be considered as empty; if `nullify-dots` is in force, there would be nothing to do (in this case the previous commands only write an instruction in a kind of `code-after`); however, if `nullify-dots` is not in force, a phantom of `\ldots`, `\cdots`, `\vdots` is inserted and its width is not equal to zero; that’s why these commands raise a boolean `\g_@@_empty_cell_bool` and we begin by testing this boolean.

```

497     \bool_if:NTF \g_@@_empty_cell_bool
498     {
499         \box_use_drop:N \l_@@_cell_box
500         \bool_gset_false:N \g_@@_empty_cell_bool
501     }

```

```

502 {
503   \dim_compare:nNnTF { \box_wd:N \l_@@_cell_box } > \c_zero_dim
504     \@@_node_for_the_cell:
505     { \box_use_drop:N \l_@@_cell_box }
506   }
507   \bool_gset_false:N \g_@@_empty_cell_bool
508 }

```

The following command creates the PGF name of the node with, of course, `\l_@@_cell_box` as the content.

```

509 \cs_new_protected:Npn \@@_node_for_the_cell:
510 {
511   \pgfpicture
512   \pgfsetbaseline \c_zero_dim
513   \pgfrememberpicturepositiononpagetrue
514   \pgfset
515   {
516     inner~sep = \c_zero_dim ,
517     minimum~width = \c_zero_dim
518   }
519   \pgfnode
520   { rectangle }
521   { base }
522   { \box_use_drop:N \l_@@_cell_box }
523   { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
524   { }
525   \str_if_empty:NF \l_@@_name_str
526   {
527     \pgfnodealias
528     { \l_@@_name_str - \int_use:N \c@iRow - \int_use:N \c@jCol }
529     { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
530   }
531   \endpgfpicture
532 }

```

The first argument of the following command `\@@_instruction_of_type:nn` defined below is the type of the instruction (`Cdots`, `Vdots`, `Ddots`, etc.). The second argument is the list of options. This command writes in the corresponding `\g_@@_type_lines_tl` the instruction which will actually 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[color=red] & & 
\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:nnn {2}{2}{}
\@@_draw_Cdots:nnn {3}{2}{color=red}

```

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

```

533 \bool_if:NTF \c_@@_draft_bool
534 { \cs_set_protected:Npn \@@_instruction_of_type:nn #1 #2 { } }
535 {
536   \cs_new_protected:Npn \@@_instruction_of_type:nn #1 #2
537   {

```

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).

```

538     \tl_gput_right:cx
539     { g_@@_ #1 _ lines _ tl }
540     {
541         \use:c { @@ _ draw _ #1 : nnn }
542         { \int_use:N \c@iRow }
543         { \int_use:N \c@jCol }
544         { \exp_not:n { #2 } }
545     }
546 }
547 }
```

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`.

```

548 \cs_new_protected:Npn \@@_array:
549 {
550     \bool_if:NTF \c_@@_revtex_bool
551     {
552         \cs_set_eq:NN \@acoll \@arrayacol
553         \cs_set_eq:NN \@acolr \@arrayacol
554         \cs_set_eq:NN \@acol \@arrayacol
555         \cs_set:Npn \@halignto { }
556         \@array@array
557     }
558     \array
```

`\l_@@_baseline_str` may have the value `t`, `c` or `b`. However, if the value is `b`, we compose the `\array` (of `array`) with the option `t` and the right translation will be done further.

```

559     [ \str_if_eq:VnTF \l_@@_baseline_str c c t ]
560 }
```

We keep in memory the standard version of `\ialign` because we will redefine `\ialign` in the environment `{NiceArrayWithDelims}` but restore the standard version for use in the cells of the array.

```

561 \cs_set_eq:NN \@@_standard_ialign: \ialign
```

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

```

562 \cs_new:Npn \@@_everycr: { \noalign { \@@_everycr_i: } }
563 \cs_new_protected:Npn \@@_everycr_i:
564 {
565     \int_gzero:N \c@jCol
```

The `\hbox:n` (or `\hbox`) is mandatory.

```

566     \hbox
567     {
568         \pgfpicture
569         \pgfrememberpicturerepositiononpagetrue
570         \pgfcoordinate { \@@_env: - row - \@@_succ:n \c@iRow }
571         \pgfpintorigin
572         \str_if_empty:NF \l_@@_name_str
573         {
574             \pgfnodealias
575             { \@@_env: - row - \int_use:N \c@iRow - row }
576             { \l_@@_name_str - row - \int_use:N \c@iRow - row }
577         }
578         \endpgfpicture
579     }
```

We add the potential horizontal lines specified by the option `hlines`.

```

580     \bool_if:NT \l_@@_hlines_bool
581     {
```

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.

```

582     \int_compare:nNnT \c@iRow > { -1 }
583     {
584         \bool_if:NF \g_@@_row_of_col_done_bool
585         {
586             \int_compare:nNnF \c@iRow = \l_@@_last_row_int
587             {
588                 \bool_if:NTF \c_@@_colortbl_loaded_bool
589                 { { \CT@arc@ \hrule height \arrayrulewidth } }
590                 { \hrule height \arrayrulewidth }
591             }
592         }
593     }
594 }
595 }
```

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

```

596 \cs_new_protected:Npn \@@_pre_array:
597 {
598     \box_clear_new:N \l_@@_cell_box
599     \cs_if_exist:NT \theiRow
600     { \int_set_eq:NN \l_@@_save_iRow_int \c@iRow }
601     \int_gzero_new:N \c@iRow
602     \cs_if_exist:NT \thejCol
603     { \int_set_eq:NN \l_@@_save_jCol_int \c@jCol }
604     \int_gzero_new:N \c@jCol
605     \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}`).

```

606     \bool_if:NT \l_@@_small_bool
607     {
608         \cs_set:Npn \arraystretch { 0.47 }
609         \dim_set:Nn \arraycolsep { 1.45 pt }
610     }
```

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`.

```

611     \cs_set:Npn \ialign
612     {
613         \bool_if:NTF \c_@@_colortbl_loaded_bool
614         {
615             \CT@everycr
616             {
617                 \noalign { \cs_gset_eq:NN \CT@row@color \prg_do_nothing: }
618                 \@@_everycr:
619             }
620         }
621         { \everycr { \@@_everycr: } }
622         \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`³² 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

³²The option `small` of `nicematrix` changes (among other) the value of `\arraystretch`. This is done, of course, before the call of `{array}`.

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`.

```

623 \dim_gzero_new:N \g_@@_dp_row_zero_dim
624 \dim_gset:Nn \g_@@_dp_row_zero_dim { \box_dp:N \@arstrutbox }
625 \dim_gzero_new:N \g_@@_ht_row_zero_dim
626 \dim_gset:Nn \g_@@_ht_row_zero_dim { \box_ht:N \@arstrutbox }
627 \dim_gzero_new:N \g_@@_ht_row_one_dim
628 \dim_gset:Nn \g_@@_ht_row_one_dim { \box_ht:N \@arstrutbox }
629 \dim_gzero_new:N \g_@@_dp_ante_last_row_dim
630 \dim_gzero_new:N \g_@@_ht_last_row_dim
631 \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
632 \dim_gzero_new:N \g_@@_dp_last_row_dim
633 \dim_gset:Nn \g_@@_dp_last_row_dim { \box_dp:N \@arstrutbox }

```

After its first use, 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`.³³

```

634 \cs_set_eq:NN \ialign \@_standard_ialign:
635 \halign
636 }

```

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}`.

```

637 \newcolumntype L { > \@_Cell: l < \@_end_Cell: }
638 \newcolumntype C { > \@_Cell: c < \@_end_Cell: }
639 \newcolumntype R { > \@_Cell: r < \@_end_Cell: }

```

We keep in memory the old versions of `\ldots`, `\cdots`, etc. only because we use them inside `\phantom` commands in order that the new commands `\Ldots`, `\Cdots`, etc. give the same spacing (except when the option `nullify-dots` is used).

```

640 \cs_set_eq:NN \@_ldots \ldots
641 \cs_set_eq:NN \@_cdots \cdots
642 \cs_set_eq:NN \@_vdots \vdots
643 \cs_set_eq:NN \@_ddots \ddots
644 \cs_set_eq:NN \@_iddots \iddots
645 \cs_set_eq:NN \firsthline \hline
646 \cs_set_eq:NN \lasthline \hline
647 \cs_set_eq:NN \Ldots \@_Ldots
648 \cs_set_eq:NN \Cdots \@_Cdots
649 \cs_set_eq:NN \Vdots \@_Vdots
650 \cs_set_eq:NN \Ddots \@_Ddots
651 \cs_set_eq:NN \Iddots \@_Iddots
652 \cs_set_eq:NN \hdottedline \@_hdottedline:
653 \cs_set_eq:NN \Hspace \@_Hspace:
654 \cs_set_eq:NN \Hdotsfor \@_Hdotsfor:
655 \cs_set_eq:NN \multicolumn \@_multicolumn:nnn
656 \cs_set_eq:NN \Block \@_Block:
657 \cs_set_eq:NN \rotate \@_rotate:
658 \cs_set_eq:NN \OnlyMainNiceMatrix \@_OnlyMainNiceMatrix:n
659 \cs_set_eq:NN \CodeAfter \@_CodeAfter:n
660 \bool_if:NT \l_@@_renew_dots_bool
661 {
662 \cs_set_eq:NN \ldots \@_Ldots
663 \cs_set_eq:NN \cdots \@_Cdots
664 \cs_set_eq:NN \vdots \@_Vdots
665 \cs_set_eq:NN \ddots \@_Ddots
666 \cs_set_eq:NN \iddots \@_Iddots
667 \cs_set_eq:NN \dots \@_Ldots
668 \cs_set_eq:NN \hdotsfor \@_Hdotsfor:
669 }

```

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`,

³³The user will probably not employ directly `\ialign` in the array... but more likely environments that utilize `\ialign` internally (e.g.: `{substack}`).

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).

```
670 \seq_gclear_new:N \g_@@_multicolumn_cells_seq
671 \seq_gclear_new:N \g_@@_multicolumn_sizes_seq
```

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

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

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

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

```
673 \int_gzero_new:N \g_@@_row_total_int
```

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

```
674 \int_gzero_new:N \g_@@_col_total_int
675 \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:`.

```
676 \cs_set_eq:NN \NC@find@w \relax
677 \cs_set_eq:NN \NC@find@W \relax
678 \newcolumnntype w [ 2 ]
679 {
680   > {
681     \hbox_set:Nw \l_@@_cell_box
682     \@@_Cell:
683   }
684   c
685   < {
686     \@@_end_Cell:
687     \hbox_set_end:
```

The `\str_lowercase:n` is only for giving the user the ability to write `wC{1cm}` instead of `wc{1cm}` for homogeneity with the letters `L`, `C` and `R` used elsewhere in the preamble instead of `l`, `c` and `r`.

```
688     \makebox [ ##2 ] [ \str_lowercase:n { ##1 } ]
689     { \box_use_drop:N \l_@@_cell_box }
690   }
691 }
692 \newcolumnntype W [ 2 ]
693 {
694   > {
695     \hbox_set:Nw \l_@@_cell_box
696     \@@_Cell:
697   }
698   c
699   < {
700     \@@_end_Cell:
701     \hbox_set_end:
702     \cs_set_eq:NN \hss \hfil
703     \makebox [ ##2 ] [ \str_lowercase:n { ##1 } ]
704     { \box_use_drop:N \l_@@_cell_box }
705   }
706 }
```

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 packages, 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).

```

707 \tl_set_rescan:Nno
708 \l_@@_letter_for_dotted_lines_str { } \l_@@_letter_for_dotted_lines_str
709 \exp_args:NV \newcolumntype \l_@@_letter_for_dotted_lines_str
710 {
711 !
712 }
```

The following code because we want the dotted line to have exactly the same position as a vertical rule drawn by “|” (considering the rule having a width equal to the diameter of the dots).

```

713 \int_compare:nNnF \c@iRow = 0
714 {
715 \int_compare:nNnF \c@iRow = \l_@@_last_row_int
716 { \skip_horizontal:N 2\l_@@_radius_dim }
717 }
```

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.

```

718 \int_compare:nNnT \c@jCol > \g_@@_last_vdotted_col_int
719 {
720 \int_gset_eq:NN \g_@@_last_vdotted_col_int \c@jCol
721 \tl_gput_right:Nx \g_@@_internal_code_after_tl
```

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

```

722 { \@@_vdottedline:n { \int_use:N \c@jCol } }
723 }
724 }
725 }
726 \int_gzero_new:N \g_@@_last_vdotted_col_int
727 \bool_if:NT \c_@@_siunitx_loaded_bool \@@_renew_NC@rewrite@S:
728 \int_gset:Nn \g_@@_last_vdotted_col_int { -1 }
729 \bool_gset_false:N \g_@@_last_col_found_bool
```

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

```

730 \tl_gclear_new:N \g_@@_Cdots_lines_tl
731 \tl_gclear_new:N \g_@@_Ldots_lines_tl
732 \tl_gclear_new:N \g_@@_Vdots_lines_tl
733 \tl_gclear_new:N \g_@@_Ddots_lines_tl
734 \tl_gclear_new:N \g_@@_Iddots_lines_tl
735 \tl_gclear_new:N \g_@@_Hdotsfor_lines_tl
736 }
```


The environment {NiceArrayWithDelims}

```

737 \NewDocumentEnvironment { NiceArrayWithDelims } { m m O { } m ! O { } }
738 {
739   \tl_set:Nn \l_@@_left_delim_tl { #1 }
740   \tl_set:Nn \l_@@_right_delim_tl { #2 }
741   \bool_gset_false:N \g_@@_row_of_col_done_bool
742   \str_if_empty:NT \g_@@_name_env_str
743     { \str_gset:Nn \g_@@_name_env_str { NiceArrayWithDelims } }
744   \@@_adapt_S_column:
745   \@@_test_if_math_mode:
746   \bool_if:NT \l_@@_in_env_bool { \@@_fatal:n { Yet-in-env } }
747   \bool_set_true:N \l_@@_in_env_bool

```

We deactivate Tikz externalization because we will use PGF pictures with the options `overlay` and `remember picture` (or equivalent forms).

```

748   \cs_if_exist:NT \tikz@library@external@loaded
749     {
750       \tikzset { external / export = false }
751       \cs_if_exist:NT \ifstandalone
752         { \tikzset { external / optimize = false } }
753     }

```

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

```

754   \int_gincr:N \g_@@_env_int
755   \bool_if:NF \l_@@_block_auto_columns_width_bool
756     { \dim_gzero_new:N \g_@@_max_cell_width_dim }

```

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

```

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

```

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

```

758   \bool_if:NTF \l_@@_NiceArray_bool
759     { \keys_set:nn { NiceMatrix / NiceArray } }
760     { \keys_set:nn { NiceMatrix / pNiceArray } }
761     { #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).

```

762   \int_compare:nNnT \l_@@_last_row_int > { -2 }
763     {
764       \tl_put_right:Nn \@@_update_for_first_and_last_row:
765         {
766           \dim_gset:Nn \g_@@_ht_last_row_dim
767             { \dim_max:nn \g_@@_ht_last_row_dim { \box_ht:N \l_@@_cell_box } }
768           \dim_gset:Nn \g_@@_dp_last_row_dim
769             { \dim_max:nn \g_@@_dp_last_row_dim { \box_dp:N \l_@@_cell_box } }
770         }
771     }
772   \int_compare:nNnT \l_@@_last_row_int = { -1 }
773     {
774       \bool_set_true:N \l_@@_last_row_without_value_bool

```

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

```

775   \str_if_empty:NTF \l_@@_name_str
776     {
777       \cs_if_exist:cT { @@_last_row_ \int_use:N \g_@@_env_int }
778         {
779           \int_set:Nn \l_@@_last_row_int
780             { \use:c { @@_last_row_ \int_use:N \g_@@_env_int } }
781         }
782     }
783   {

```

```

784         \cs_if_exist:cT { @@_last_row_ \l_@@_name_str }
785         {
786             \int_set:Nn \l_@@_last_row_int
787             { \use:c { @@_last_row_ \l_@@_name_str } }
788         }
789     }
790 }

```

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

```

791     \int_compare:nNnT \l_@@_last_col_int = { -1 }
792     {
793         \str_if_empty:NTF \l_@@_name_str
794         {
795             \cs_if_exist:cT { @@_last_col_ \int_use:N \g_@@_env_int }
796             {
797                 \int_set:Nn \l_@@_last_col_int
798                 { \use:c { @@_last_col_ \int_use:N \g_@@_env_int } }
799             }
800         }
801         {
802             \cs_if_exist:cT { @@_last_col_ \l_@@_name_str }
803             {
804                 \int_set:Nn \l_@@_last_col_int
805                 { \use:c { @@_last_col_ \l_@@_name_str } }
806             }
807         }
808     }

```

The code in `\@@_pre_array:` is used only by `{NiceArrayWithDelims}`.

```

809     \@@_pre_array:

```

We compute the width of the two delimiters.

```

810     \dim_zero_new:N \l_@@_left_delim_dim
811     \dim_zero_new:N \l_@@_right_delim_dim
812     \bool_if:NTF \l_@@_NiceArray_bool
813     {
814         \dim_gset:Nn \l_@@_left_delim_dim { 2 \arraycolsep }
815         \dim_gset:Nn \l_@@_right_delim_dim { 2 \arraycolsep }
816     }
817     {

```

The command `\bBigg@` is a command of `amsmath`.

```

818     \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 #1 $ }
819     \dim_set:Nn \l_@@_left_delim_dim { \box_wd:N \l_tmpa_box }
820     \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 #2 $ }
821     \dim_set:Nn \l_@@_right_delim_dim { \box_wd:N \l_tmpa_box }
822 }

```

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

```

823     \box_clear_new:N \l_@@_the_array_box

```

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

```

824     \tl_set:Nn \l_tmpa_tl { #4 }
825     \int_compare:nNnTF \l_@@_first_col_int = 0
826     { \tl_put_left:NV \l_tmpa_tl \c_@@_preamble_first_col_tl }
827     {
828         \bool_lazy_all:nT
829         {
830             \l_@@_NiceArray_bool
831             { \bool_not_p:n \l_@@_vlines_bool }
832             { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }

```

```

833     }
834     { \tl_put_left:Nn \l_tmpa_tl { @ { } } }
835   }
836   \int_compare:nNnTF \l_@@_last_col_int > { -1 }
837   { \tl_put_right:NV \l_tmpa_tl \c_@@_preamble_last_col_tl }
838   {
839     \bool_lazy_all:nT
840     {
841       \l_@@_NiceArray_bool
842       { \bool_not_p:n \l_@@_vlines_bool }
843       { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
844     }
845     { \tl_put_right:Nn \l_tmpa_tl { @ { } } }
846   }
847   \tl_put_right:Nn \l_tmpa_tl { > { \@@_error_too_much_cols: } L }

```

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).

```

848   \hbox_set:Nw \l_@@_the_array_box

```

If the key `\vlines` is used, we increase `\arraycolsep` by `0.5\arrayrulewidth` in order to reserve space for the width of the vertical rules drawn with Tikz after the end of the array. However, the first `\arraycolsep` is used once (between columns, `\arraycolsep` is used twice). That's why we add a `0.5\arrayrulewidth` more.

```

849   \bool_if:NT \l_@@_vlines_bool
850   {
851     \dim_add:Nn \arraycolsep { 0.5 \arrayrulewidth }
852     \skip_horizontal:N 0.5\arrayrulewidth
853   }
854   \skip_horizontal:N \l_@@_left_margin_dim
855   \skip_horizontal:N \l_@@_extra_left_margin_dim
856   \c_math_toggle_token
857   \bool_if:NTF \l_@@_light_syntax_bool
858   { \begin { @@-light-syntax } }
859   { \begin { @@-normal-syntax } }
860 }
861 {
862   \bool_if:NTF \l_@@_light_syntax_bool
863   { \end { @@-light-syntax } }
864   { \end { @@-normal-syntax } }
865   \c_math_toggle_token
866   \skip_horizontal:N \l_@@_right_margin_dim
867   \skip_horizontal:N \l_@@_extra_right_margin_dim

```

If the key `\vlines` is used, we have increased `\arraycolsep` by `0.5\arrayrulewidth` in order to reserve space for the width of the vertical rules drawn with Tikz after the end of the array. However, the last `\arraycolsep` is used once (between columns, `\arraycolsep` is used twice). That's we add a `0.5 \arrayrulewidth` more.

```

868   \bool_if:NT \l_@@_vlines_bool { \skip_horizontal:N 0.5\arrayrulewidth }
869   \hbox_set_end:

```

End of the construction of the array (in the box `\l_@@_the_array_box`).

It the user has used the key `last-row` with a value, we control that the given value is correct (since we have just constructed the array, we know the real number of rows of the array).

```

870   \int_compare:nNnT \l_@@_last_row_int > { -2 }
871   {
872     \bool_if:NF \l_@@_last_row_without_value_bool
873     {
874       \int_compare:nNnF \l_@@_last_row_int = \c@iRow
875       {
876         \@@_error:n { Wrong-last-row }
877         \int_gset_eq:NN \l_@@_last_row_int \c@iRow
878       }

```

```

879     }
880 }

```

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”.³⁴

```

881 \int_gset_eq:NN \c@jCol \g_@@_col_total_int
882 \bool_if:nT \g_@@_last_col_found_bool { \int_gdecr:N \c@jCol }

```

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

```

883 \int_gset_eq:NN \g_@@_row_total_int \c@iRow
884 \int_compare:nNnT \l_@@_last_row_int > { -1 } { \int_gdecr:N \c@iRow }

```

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. 59).

```

885 \int_compare:nNnT \l_@@_first_col_int = 0
886 {
887   \skip_horizontal:N \arraycolsep
888   \skip_horizontal:N \g_@@_width_first_col_dim
889 }

```

The construction of the real box is different in `{NiceArray}` and in the other environments because, in `{NiceArray}`, we have to take into account the value of `baseline` and we have no delimiter to put. We begin with `{NiceArray}`.

```

890 \bool_if:NTF \l_@@_NiceArray_bool
891 {

```

Remember that, when the key `b` is used, the `\array` (of `array`) is constructed with the option `t` (and not `b`). Now, we do the translation to take into account the option `b`.

```

892 \str_if_eq:VnTF \l_@@_baseline_str { b }
893 {
894   \pgfpicture
895     \@@_qpoint: { row - 1 }
896     \dim_gset_eq:NN \g_tmpa_dim \pgf@y
897     \@@_qpoint: { row - \int_use:N \c@iRow - base }
898     \dim_gsub:Nn \g_tmpa_dim \pgf@y
899   \endpgfpicture
900   \int_compare:nNnT \l_@@_first_row_int = 0
901   {
902     \dim_gadd:Nn \g_tmpa_dim
903       { \g_@@_ht_row_zero_dim + \g_@@_dp_row_zero_dim }
904   }
905   \box_move_up:nn \g_tmpa_dim { \box_use_drop:N \l_@@_the_array_box }
906 }
907 {
908   \str_if_eq:VnTF \l_@@_baseline_str { c }
909   { \box_use_drop:N \l_@@_the_array_box }
910   {

```

We convert a value of `t` to a value of 1.

```

911 \str_if_eq:VnT \l_@@_baseline_str { t }
912 { \str_set:Nn \l_@@_baseline_str { 1 } }

```

Now, we convert the value of `\l_@@_baseline_str` (which should represent an integer) to an integer stored in `\l_tmpa_int`.

```

913 \int_set:Nn \l_tmpa_int \l_@@_baseline_str
914 \bool_if:nT
915 {
916   \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int
917   || \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int
918 }
919 {
920   \@@_error:n { bad-value-for-baseline }
921   \int_set:Nn \l_tmpa_int 1

```

³⁴We remind that the potential “first column” (exterior) has the number 0.

```

922     }
923     \pgfpicture
924     \@@_qpoint: { row - 1 }
925     \dim_gset_eq:Nn \g_tmpa_dim \pgf@y
926     \@@_qpoint: { row - \int_use:N \l_tmpa_int - base }
927     \dim_gsub:Nn \g_tmpa_dim \pgf@y
928     \endpgfpicture
929     \int_compare:nNnT \l_@@_first_row_int = 0
930     {
931         \dim_gadd:Nn \g_tmpa_dim
932         { \g_@@_ht_row_zero_dim + \g_@@_dp_row_zero_dim }
933     }
934     \box_move_up:nn \g_tmpa_dim
935     { \box_use_drop:N \l_@@_the_array_box }
936 }
937 }
938 }

```

Now, in the case of an environment `{pNiceArray}`, `{bNiceArray}`, etc. We compute `\l_tmpa_dim` which is the total height of the “first row” above the array (when the key `first-row` is used).

```

939 {
940     \int_compare:nNnTF \l_@@_first_row_int = 0
941     {
942         \dim_set_eq:Nn \l_tmpa_dim \g_@@_dp_row_zero_dim
943         \dim_add:Nn \l_tmpa_dim \g_@@_ht_row_zero_dim
944     }
945     { \dim_zero:N \l_tmpa_dim }

```

We compute `\l_tmpb_dim` which is the total height 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”.³⁵

```

946     \int_compare:nNnTF \l_@@_last_row_int > { -2 }
947     {
948         \dim_set_eq:Nn \l_tmpb_dim \g_@@_ht_last_row_dim
949         \dim_add:Nn \l_tmpb_dim \g_@@_dp_last_row_dim
950     }
951     { \dim_zero:N \l_tmpb_dim }
952     \hbox_set:Nn \l_tmpa_box
953     {
954         \c_math_toggle_token
955         \left #1
956         \vcenter
957         {

```

We take into account the “first row” (we have previously computed its total height in `\l_tmpa_dim`). The `\hbox:n` (or `\hbox`) is necessary here.

```

958         \skip_vertical:N -\l_tmpa_dim
959         \hbox
960         {
961             \skip_horizontal:N -\arraycolsep
962             \box_use_drop:N \l_@@_the_array_box
963             \skip_horizontal:N -\arraycolsep
964         }

```

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

```

965         \skip_vertical:N -\l_tmpb_dim
966     }
967     \right #2
968     \c_math_toggle_token
969 }

```

Now, the box `\l_tmpa_box` is created with the correct delimiters.

We will put the box in the TeX flow. However, we have a small work to do when the option `max-delimiter-width` is used.

³⁵A value of `-1` for `\l_@@_last_row_int` means that there is a “last row” but the the user have not set the value with the option `last row` (and we are in the first compilation).

```

970     \bool_if:NTF \l_@@_max_delimiter_width_bool
971     { \@@_put_box_in_flow_bis:nn { #1 } { #2 } }
972     \@@_put_box_in_flow:
973 }

```

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. 60).

```

974     \bool_if:NT \g_@@_last_col_found_bool
975     {
976         \skip_horizontal:N \g_@@_width_last_col_dim
977         \skip_horizontal:N \arraycolsep
978     }
979     \@@_after_array:
980 }

```

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

The command `\@@_put_box_in_flow:` puts the box `\l_tmpa_box` (which contains the array) in the flow. It is used for the environments with delimiters. First, we have to modify the height and the depth to take back into account the potential exterior rows (the total height of the first row has been computed in `\l_tmpa_dim` and the total height of the potential last row in `\l_tmpb_dim`).

```

981 \cs_new_protected:Npn \@@_put_box_in_flow:
982 {
983     \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }
984     \box_set_dp:Nn \l_tmpa_box { \box_dp:N \l_tmpa_box + \l_tmpb_dim }
985     \str_if_eq:VnTF \l_@@_baseline_str { c }
986     { \box_use_drop:N \l_tmpa_box }
987     \@@_put_box_in_flow_i:
988 }

```

The command `\@@_put_box_in_flow_i:` is used when the value of `\l_@@_baseline_str` is different of `c` (which is the initial value and the most used).

```

989 \cs_new_protected:Npn \@@_put_box_in_flow_i:
990 {
991     \str_case:VnF \l_@@_baseline_str
992     {
993         { t } { \int_set:Nn \l_tmpa_int 1 }
994         { b } { \int_set_eq:NN \l_tmpa_int \c@iRow }
995     }
996     { \int_set:Nn \l_tmpa_int \l_@@_baseline_str }
997     \bool_if:nTF
998     {
999         \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int
1000         || \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int
1001     }
1002     {
1003         \@@_error:n { bad~value~for~baseline }
1004         \int_set:Nn \l_tmpa_int 1
1005     }
1006     \pgfpicture
1007     \@@_qpoint: { row - 1 }
1008     \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1009     \@@_qpoint: { row - \@@_succ:n \c@iRow }
1010     \dim_gadd:Nn \g_tmpa_dim \pgf@y
1011     \dim_gset:Nn \g_tmpa_dim { 0.5 \g_tmpa_dim }

```

Now, `\g_tmpa_dim` contains the y -value of the center of the array (the delimiters are centered in relation with this value).

```

1012     \@@_qpoint: { row - \int_use:N \l_tmpa_int - base }
1013     \dim_gsub:Nn \g_tmpa_dim \pgf@y

```

We take into account the position of the mathematical axis.

```

1014     \dim_gsub:Nn \g_tmpa_dim { \fontdimen22 \textfont2 }

```

Now, `\g_tmpa_dim` contains the value of the y translation we have to do.

```

1015     \endpgfpicture

```

```

1016 \box_move_up:nn \g_tmpa_dim { \box_use_drop:N \l_tmpa_box }
1017 \box_use_drop:N \l_tmpa_box
1018 }

```

The command `\@@_put_box_in_flow_bis:` is used when the option `max-delimiter-width` is used because, in this case, we have to adjust the widths of the delimiters. The arguments `#1` and `#2` are the delimiters specified by the user.

```

1019 \cs_new_protected:Npn \@@_put_box_in_flow_bis:nn #1 #2
1020 {

```

We will compute the real width of both delimiters used.

```

1021 \dim_zero_new:N \l_@@_real_left_delim_dim
1022 \dim_zero_new:N \l_@@_real_right_delim_dim
1023 \hbox_set:Nn \l_tmpb_box
1024 {
1025   \c_math_toggle_token
1026   \left #1
1027   \vcenter
1028   {
1029     \vbox_to_ht:nn
1030     { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1031     { }
1032   }
1033   \right .
1034   \c_math_toggle_token
1035 }
1036 \dim_set:Nn \l_@@_real_left_delim_dim
1037 { \box_wd:N \l_tmpb_box - \nullldelimiterspace }
1038 \hbox_set:Nn \l_tmpb_box
1039 {
1040   \c_math_toggle_token
1041   \left .
1042   \vbox_to_ht:nn
1043   { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1044   { }
1045   \right #2
1046   \c_math_toggle_token
1047 }
1048 \dim_set:Nn \l_@@_real_right_delim_dim
1049 { \box_wd:N \l_tmpb_box - \nullldelimiterspace }

```

Now, we can put the box in the TeX flow with the horizontal adjustments on both sides.

```

1050 \skip_horizontal:N \l_@@_left_delim_dim
1051 \skip_horizontal:N -\l_@@_real_left_delim_dim
1052 \@@_put_box_in_flow:
1053 \skip_horizontal:N \l_@@_right_delim_dim
1054 \skip_horizontal:N -\l_@@_real_right_delim_dim
1055 }

```

The construction of the array in the environment `{NiceArrayWithDelims}` is, in fact, done by the environment `{@@-light-syntax}` or by the environment `{@@-normal-syntax}` (whether the option `light-syntax` is used or not). When the key `light-syntax` is not used, the construction is a standard environment (and, thus, it's possible to use verbatim in the array).

```

1056 \NewDocumentEnvironment { @@-normal-syntax } { }

```

First, we test whether the environment is empty. If it is empty, we raise a fatal error (it's only a security). In order to detect whether it is empty, we test whether the next token is `\end` and, if it's the case, we test if this is the end of the environment (if it is not, an standard error will be raised by LaTeX for incorrect nested environments).

```

1057 {
1058   \peek_meaning_ignore_spaces:NTF \end
1059   { \@@_analyze_end:Nn }

```

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

```

1060     { \exp_args:NV \@@_array: \l_tmpa_tl }
1061   }
1062   {
1063     \@@_create_col_nodes:
1064   \endarray
1065   }

```

When the key `light-syntax` is used, we use an environment which takes its whole body as an argument (with the specifier `b` of `xparse`).

```

1066 \NewDocumentEnvironment { @@-light-syntax } { b }
1067   {

```

First, we test whether the environment is empty. It's only a security. Of course, this test is more easy than the similar test for the “normal syntax” because we have the whole body of the environment in `#1`.

```

1068   \tl_if_empty:nT { #1 } { \@@_fatal:n { empty-environment } }
1069   \tl_map_inline:nn { #1 }
1070   {
1071     \tl_if_eq:nnT { ##1 } { & }
1072     { \@@_fatal:n { ampersand-in-light-syntax } }
1073     \tl_if_eq:nnT { ##1 } { \ }
1074     { \@@_fatal:n { double-backslash-in-light-syntax } }
1075   }

```

The body of the environment, which is stored in the argument `#1`, is now splitted into items (and *not* tokens)

```

1076   \seq_gclear_new:N \g_@@_rows_seq
1077   \tl_set_rescan:Nno \l_@@_end_of_row_tl { } \l_@@_end_of_row_tl
1078   \exp_args:NNV \seq_gset_split:Nnn \g_@@_rows_seq \l_@@_end_of_row_tl { #1 }

```

If the environment uses the option `last-row` without value (i.e. without saying the number of the rows), we have now the opportunity to know that value. We do it, and so, if the token list `\l_@@_code_for_last_row_tl` is not empty, we will use directly where it should be.

```

1079   \int_compare:nNnT \l_@@_last_row_int = { -1 }
1080   { \int_set:Nn \l_@@_last_row_int { \seq_count:N \g_@@_rows_seq } }

```

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

```

1081   \exp_args:NV \@@_array: \l_tmpa_tl

```

We need a global affectation because, when executing `\l_tmpa_tl`, we will exit the first cell of the array.

```

1082   \seq_gpop_left:NN \g_@@_rows_seq \l_tmpa_tl
1083   \exp_args:NV \@@_line_with_light_syntax_i:n \l_tmpa_tl
1084   \seq_map_function:NN \g_@@_rows_seq \@@_line_with_light_syntax:n
1085   \@@_create_col_nodes:
1086 \endarray
1087 }

```

Now, the second part of the environment. It is empty. That's not surprising because we have caught the whole body of the environment with the specifier `b` provided by `xparse`.

```

1088   { }

1089 \cs_new_protected:Npn \@@_line_with_light_syntax_i:n #1
1090   {
1091     \seq_gclear_new:N \g_@@_cells_seq
1092     \seq_gset_split:Nnn \g_@@_cells_seq { ~ } { #1 }
1093     \seq_gpop_left:NN \g_@@_cells_seq \l_tmpa_tl
1094     \l_tmpa_tl
1095     \seq_map_function:NN \g_@@_cells_seq \@@_cell_with_light_syntax:n
1096   }

1097 \cs_new_protected:Npn \@@_line_with_light_syntax:n #1
1098   {
1099     \tl_if_empty:nF { #1 }

```



```

1100      { \\\ \@@_line_with_light_syntax_i:n { #1 } }
1101    }
1102 \cs_new_protected:Npn \@@_cell_with_light_syntax:n #1 { & #1 }

```

The following command is used by the code which detects whether the environment is empty (we raise a fatal error in this case: it's only a security).

```

1103 \cs_new_protected:Npn \@@_analyze_end:Nn #1 #2
1104 {
1105   \str_if_eq:VnT \g_@@_name_env_str { #2 }
1106   { \@@_fatal:n { empty~environment } }

```

We repeat in the stream the `\end{...}` we have extracted and the user will have an error for incorrect nested environments.

```

1107   \end { #2 }
1108 }

```

The command `\@@_create_col_nodes:` will construct a special last row. That last row is a false row used to create the col-nodes and to fix the width of the columns (when the array is constructed with an option which specify the width of the columns).

```

1109 \cs_new:Npn \@@_create_col_nodes:
1110 {
1111   \crrc
1112   \int_compare:nNnT \c@iRow = 0 { \@@_fatal:n { Zero~row } }
1113   \int_compare:nNnT \l_@@_first_col_int = 0
1114   {
1115     \omit
1116     \skip_horizontal:N -2\arraycolsep
1117     \pgfpicture
1118     \pgfrememberpicturepositiononpagetrue
1119     \pgfcoordinate { \@@_env: - col - 0 } \pgfpointorigin
1120     \str_if_empty:NF \l_@@_name_str
1121     { \pgfnodealias { \@@_env: - col - 0 } { \l_@@_name_str - col - 0 } }
1122     \endpgfpicture
1123     &
1124   }
1125   \omit

```

The following instruction must be put after the instruction `\omit`.

```

1126   \bool_gset_true:N \g_@@_row_of_col_done_bool

```

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

```

1127   \pgfpicture
1128   \pgfrememberpicturepositiononpagetrue
1129   \pgfcoordinate { \@@_env: - col - 1 } \pgfpointorigin
1130   \str_if_empty:NF \l_@@_name_str
1131   { \pgfnodealias { \@@_env: - col - 1 } { \l_@@_name_str - col - 1 } }
1132   \endpgfpicture

```

We compute in `\g_tmpa_skip` the common width of the columns (it's a skip and not a dimension). 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_skip`, like all the affectations, must be done after the `\omit` of the cell.

We give a default value for `\g_tmpa_skip` (0 pt plus 1 fill) but it will just after erased by a fixed value in the concerned cases.

```

1133   \skip_gset:Nn \g_tmpa_skip { 0 pt+plus 1 fill }
1134   \bool_if:NF \l_@@_auto_columns_width_bool
1135   { \dim_compare:nNnT \l_@@_columns_width_dim > \c_zero_dim }
1136   {
1137     \bool_lazy_and:nnTF
1138     \l_@@_auto_columns_width_bool
1139     { \bool_not_p:n \l_@@_block_auto_columns_width_bool }
1140     { \skip_gset_eq:NN \g_tmpa_skip \g_@@_max_cell_width_dim }
1141     { \skip_gset_eq:NN \g_tmpa_skip \l_@@_columns_width_dim }

```

```

1142     \skip_gadd:Nn \g_tmpa_skip { 2 \arraycolsep }
1143   }
1144   \skip_horizontal:N \g_tmpa_skip
1145   \hbox
1146   {
1147     \pgfpicture
1148     \pgfrememberpicturepositiononpagetrue
1149     \pgfcoordinate { \@@_env: - col - 2 } \pgfpintorigin
1150     \str_if_empty:NF \l_@@_name_str
1151     { \pgfnodealias { \@@_env: - col - 2 } { \l_@@_name_str - col - 2 } }
1152     \endpgfpicture
1153   }

```

We begin a loop over the columns. The integer `\g_tmpa_int` will be the number of the current column. This integer is used for the Tikz nodes.

```

1154   \int_gset:Nn \g_tmpa_int 1
1155   \bool_if:NTF \g_@@_last_col_found_bool
1156   { \prg_replicate:nn { \g_@@_col_total_int - 2 } }
1157   { \prg_replicate:nn { \g_@@_col_total_int - 1 } }
1158   {
1159     &
1160     \omit

```

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

```

1161     \int_gincr:N \g_tmpa_int
1162     \skip_horizontal:N \g_tmpa_skip

```

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

```

1163     \pgfpicture
1164     \pgfrememberpicturepositiononpagetrue
1165     \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1166     \pgfpintorigin
1167     \str_if_empty:NF \l_@@_name_str
1168     {
1169       \pgfnodealias
1170       { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1171       { \l_@@_name_str - col - \@@_succ:n \g_tmpa_int }
1172     }
1173     \endpgfpicture
1174   }
1175   \bool_if:NT \g_@@_last_col_found_bool
1176   {
1177     \skip_horizontal:N 2\arraycolsep
1178     \pgfpicture
1179     \pgfrememberpicturepositiononpagetrue
1180     \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
1181     \pgfpintorigin
1182     \str_if_empty:NF \l_@@_name_str
1183     {
1184       \pgfnodealias
1185       { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
1186       { \l_@@_name_str - col - \@@_succ:n \g_@@_col_total_int }
1187     }
1188     \endpgfpicture
1189     \skip_horizontal:N -2\arraycolsep
1190   }
1191   \cr
1192 }

```

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

```

1193 \tl_const:Nn \c_@@_preamble_first_col_tl
1194 {
1195   >
1196   {

```

```
1197 \l_@@_begin_of_row:
```

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

```
1198 \hbox_set:Nw \l_@@_cell_box
1199 \c_math_toggle_token
1200 \bool_if:NT \l_@@_small_bool \scriptstyle
```

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

```
1201 \bool_lazy_and:nnT
1202 { \int_compare_p:nNn \c@iRow > 0 }
1203 {
1204   \bool_lazy_or_p:nn
1205   { \int_compare_p:nNn \l_@@_last_row_int < 0 }
1206   { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
1207 }
1208 {
1209   \l_@@_code_for_first_col_tl
1210   \xglobal \colorlet { nicematrix-first-col } { . }
1211 }
1212 }
```

Be careful: despite this letter `l` the cells of the “first column” are composed in a `R` manner since they are composed in a `\hbox_overlap_left:n`.

```
1213 l
1214 <
1215 {
1216   \c_math_toggle_token
1217   \hbox_set_end:
1218   \l_@@_update_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.

```
1219 \dim_gset:Nn \g_@@_width_first_col_dim
1220 { \dim_max:nn \g_@@_width_first_col_dim { \box_wd:N \l_@@_cell_box } }
```

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

```
1221 \hbox_overlap_left:n
1222 {
1223   \dim_compare:nNnTF { \box_wd:N \l_@@_cell_box } > \c_zero_dim
1224   \l_@@_node_for_the_cell:
1225   { \box_use_drop:N \l_@@_cell_box }
1226   \skip_horizontal:N \l_@@_left_delim_dim
1227   \skip_horizontal:N \l_@@_left_margin_dim
1228   \skip_horizontal:N \l_@@_extra_left_margin_dim
1229 }
1230 \skip_horizontal:N -2\arraycolsep
1231 }
1232 }
```

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

```
1233 \tl_const:Nn \c_@@_preamble_last_col_tl
1234 {
1235   >
1236   {
```

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

```
1237 \bool_gset_true:N \g_@@_last_col_found_bool
1238 \int_gincr:N \c@jCol
1239 \int_gset_eq:NN \g_@@_col_total_int \c@jCol
```

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

```
1240 \hbox_set:Nw \l_@@_cell_box
1241 \c_math_toggle_token
1242 \bool_if:NT \l_@@_small_bool \scriptstyle
```

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

```

1243     \int_compare:nNnT \c@iRow > 0
1244     {
1245         \bool_lazy_or:nnT
1246         { \int_compare_p:nNn \l_@@_last_row_int < 0 }
1247         { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
1248         {
1249             \l_@@_code_for_last_col_tl
1250             \xglobal \colorlet { nicematrix-last-col } { . }
1251         }
1252     }
1253 }
1254 \l
1255 <
1256 {
1257     \c_math_toggle_token
1258     \hbox_set_end:
1259     \@@_update_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.

```

1260     \dim_gset:Nn \g_@@_width_last_col_dim
1261     { \dim_max:nn \g_@@_width_last_col_dim { \box_wd:N \l_@@_cell_box } }
1262     \skip_horizontal:N -2\arraycolsep

```

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

```

1263     \hbox_overlap_right:n
1264     {
1265         \dim_compare:nNnT { \box_wd:N \l_@@_cell_box } > \c_zero_dim
1266         {
1267             \skip_horizontal:N \l_@@_right_delim_dim
1268             \skip_horizontal:N \l_@@_right_margin_dim
1269             \skip_horizontal:N \l_@@_extra_right_margin_dim
1270             \@@_node_for_the_cell:
1271         }
1272     }
1273 }
1274 }

```

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.

```

1275 \NewDocumentEnvironment { NiceArray } { }
1276 {
1277     \bool_set_true:N \l_@@_NiceArray_bool
1278     \str_if_empty:NT \g_@@_name_env_str
1279     { \str_gset:Nn \g_@@_name_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).

```

1280     \NiceArrayWithDelims . .
1281 }
1282 { \endNiceArrayWithDelims }

```

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

```

1283 \NewDocumentEnvironment { pNiceArray } { }
1284 {
1285     \str_if_empty:NT \g_@@_name_env_str
1286     { \str_gset:Nn \g_@@_name_env_str { pNiceArray } }
1287     \@@_test_if_math_mode:
1288     \NiceArrayWithDelims ( )
1289 }

```

```

1290 { \endNiceArrayWithDelims }
1291 \NewDocumentEnvironment { bNiceArray } { }
1292 {
1293   \str_if_empty:NT \g_@@_name_env_str
1294   { \str_gset:Nn \g_@@_name_env_str { bNiceArray } }
1295   \@@_test_if_math_mode:
1296   \NiceArrayWithDelims [ ]
1297 }
1298 { \endNiceArrayWithDelims }
1299 \NewDocumentEnvironment { BNiceArray } { }
1300 {
1301   \str_if_empty:NT \g_@@_name_env_str
1302   { \str_gset:Nn \g_@@_name_env_str { BNiceArray } }
1303   \@@_test_if_math_mode:
1304   \NiceArrayWithDelims \{ \}
1305 }
1306 { \endNiceArrayWithDelims }
1307 \NewDocumentEnvironment { vNiceArray } { }
1308 {
1309   \str_if_empty:NT \g_@@_name_env_str
1310   { \str_gset:Nn \g_@@_name_env_str { vNiceArray } }
1311   \@@_test_if_math_mode:
1312   \NiceArrayWithDelims | |
1313 }
1314 { \endNiceArrayWithDelims }
1315 \NewDocumentEnvironment { VNiceArray } { }
1316 {
1317   \str_if_empty:NT \g_@@_name_env_str
1318   { \str_gset:Nn \g_@@_name_env_str { VNiceArray } }
1319   \@@_test_if_math_mode:
1320   \NiceArrayWithDelims \| \|
1321 }
1322 { \endNiceArrayWithDelims }

```

The environment `{NiceMatrix}` and its variants

```

1323 \cs_new_protected:Npn \@@_define_env:n #1
1324 {
1325   \NewDocumentEnvironment { #1 NiceMatrix } { ! 0 { } }
1326   {
1327     \str_gset:Nn \g_@@_name_env_str { #1 NiceMatrix }
1328     \tl_set:Nn \l_@@_type_of_col_tl C
1329     \keys_set:nn { NiceMatrix / NiceMatrix } { ##1 }
1330     \exp_args:Nnx \@@_begin_of_NiceMatrix:nn { #1 } \l_@@_type_of_col_tl
1331   }
1332   { \end { #1 NiceArray } }
1333 }
1334 \cs_new_protected:Npn \@@_begin_of_NiceMatrix:nn #1 #2
1335 {
1336   \begin { #1 NiceArray }
1337   {
1338     *
1339     {
1340       \int_compare:nNnTF \l_@@_last_col_int < 0
1341       \c@MaxMatrixCols
1342       { \@@_pred:n \l_@@_last_col_int }
1343     }
1344     #2
1345   }
1346 }
1347 \@@_define_env:n { }

```

```

1348 \@@_define_env:n p
1349 \@@_define_env:n b
1350 \@@_define_env:n B
1351 \@@_define_env:n v
1352 \@@_define_env:n V

```

After the construction of the array

```

1353 \cs_new_protected:Npn \@@_after_array:
1354 {
1355   \group_begin:

```

When the option `last-col` is used in the environments with explicit preambles (like `{NiceArray}`, `{pNiceArray}`, etc.) a special type of column is used at the end of the preamble in order to compose the cells in an overlapping position (with `\hbox_overlap_right:n`) but (if `last-col` has been used), we don't have the number of that last column. However, we have to know that number for the color of the potential `\Vdots` drawn in that last column. That's why we fix the correct value of `\l_@@_last_col_int` in that case.

```

1356   \bool_if:NT \g_@@_last_col_found_bool
1357   { \int_set_eq:NN \l_@@_last_col_int \g_@@_col_total_int }

```

If we are in an environment without preamble (like `{NiceMatrix}` or `{pNiceMatrix}`) and if the option `last-col` has been used without value we fix the real value of `\l_@@_last_col_int`.

```

1358   \bool_if:NT \l_@@_last_col_without_value_bool
1359   {
1360     \dim_set_eq:NN \l_@@_last_col_int \g_@@_col_total_int
1361     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
1362     \iow_shipout:Nx \@mainaux
1363     {
1364       \cs_gset:cpn { @@_last_col_ \int_use:N \g_@@_env_int }
1365       { \int_use:N \g_@@_col_total_int }
1366     }
1367     \str_if_empty:NF \l_@@_name_str
1368     {
1369       \iow_shipout:Nx \@mainaux
1370       {
1371         \cs_gset:cpn { @@_last_col_ \l_@@_name_str }
1372         { \int_use:N \g_@@_col_total_int }
1373       }
1374     }
1375     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
1376   }

```

It's also time to give to `\l_@@_last_row_int` its real value. But, if the user had used the option `last-row` without value, we write in the aux file the number of that last row for the next run.

```

1377   \bool_if:NT \l_@@_last_row_without_value_bool
1378   {
1379     \dim_set_eq:NN \l_@@_last_row_int \g_@@_row_total_int

```

If the option `light-syntax` is used, we have nothing to write since, in this case, the number of rows is directly determined.

```

1380   \bool_if:NF \l_@@_light_syntax_bool
1381   {
1382     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
1383     \iow_shipout:Nx \@mainaux
1384     {
1385       \cs_gset:cpn { @@_last_row_ \int_use:N \g_@@_env_int }
1386       { \int_use:N \g_@@_row_total_int }
1387     }

```

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.

```

1388     \str_if_empty:NF \l_@@_name_str
1389     {
1390       \iow_shipout:Nx \@mainaux

```

```

1391         {
1392             \cs_gset:cpn { @@_last_row_ \l_@@_name_str }
1393             { \int_use:N \g_@@_row_total_int }
1394         }
1395     }
1396     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
1397 }
1398 }

```

By default, the diagonal lines will be parallelized³⁶. 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.

```

1399     \bool_if:NT \l_@@_parallelize_diags_bool
1400     {
1401         \int_gzero_new:N \g_@@_ddots_int
1402         \int_gzero_new:N \g_@@_iddots_int

```

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

```

1403         \dim_gzero_new:N \g_@@_delta_x_one_dim
1404         \dim_gzero_new:N \g_@@_delta_y_one_dim
1405         \dim_gzero_new:N \g_@@_delta_x_two_dim
1406         \dim_gzero_new:N \g_@@_delta_y_two_dim
1407     }
1408     \bool_if:nTF \l_@@_medium_nodes_bool
1409     {
1410         \bool_if:NTF \l_@@_large_nodes_bool
1411         \@@_create_medium_and_large_nodes:
1412         \@@_create_medium_nodes:
1413     }
1414     { \bool_if:NT \l_@@_large_nodes_bool \@@_create_large_nodes: }
1415     \int_zero_new:N \l_@@_initial_i_int
1416     \int_zero_new:N \l_@@_initial_j_int
1417     \int_zero_new:N \l_@@_final_i_int
1418     \int_zero_new:N \l_@@_final_j_int
1419     \bool_set_false:N \l_@@_initial_open_bool
1420     \bool_set_false:N \l_@@_final_open_bool

```

If the option `small` is used, the values `\l_@@_radius_dim` and `\l_@@_inter_dots_dim` (used to draw the dotted lines created by `\hdottedline` and `\vdotteline` and also for all the other dotted lines when `line-style` is equal to `standard`, which is the initial value) are changed.

```

1421     \bool_if:NT \l_@@_small_bool
1422     {
1423         \dim_set:Nn \l_@@_radius_dim { 0.37 pt }
1424         \dim_set:Nn \l_@@_inter_dots_dim { 0.25 em }

```

The dimension `\l_@@_xdots_shorten_dim` corresponds to the option `xdots/shorten` available to the user. That's why we give a new value according to the current value, and not an absolute value.

```

1425         \dim_set:Nn \l_@@_xdots_shorten_dim { 0.6 \l_@@_xdots_shorten_dim }
1426     }

```

Now, we really draw the lines.

```

1427     \@@_draw_dotted_lines:

```

We draw the vertical rules of the option `vlines` before the `internal-code-after` because the option `white` of a `\Block` may have to erase these vertical rules.

```

1428     \bool_if:NT \l_@@_vlines_bool \@@_draw_vlines:
1429     \g_@@_internal_code_after_tl
1430     \tl_gclear:N \g_@@_internal_code_after_tl
1431     \bool_if:NT \c_@@_tikz_loaded_bool
1432     {

```

³⁶It's possible to use the option `parallelize-diags` to disable this parallelization.

```

1433     \tikzset
1434     {
1435         every~picture / .style =
1436         {
1437             overlay ,
1438             remember~picture ,
1439             name-prefix = \@@_env: -
1440         }
1441     }
1442 }
1443 \cs_set_eq:NN \line \@@_line
1444 \g_@@_code_after_tl
1445 \tl_gclear:N \g_@@_code_after_tl
1446 \group_end:
1447 \str_gclear:N \g_@@_name_env_str
1448 \@@_restore_iRow_jCol:
1449 }

```

We recall that, when externalization is used, `\tikzpicture` and `\endtikzpicture` (or `\pgfpicture` and `\endpgfpicture`) must be directly “visible”. That’s why we have to define the adequate version of `\@@_draw_dotted_lines:` whether Tikz is loaded or not (in that case, only PGF is loaded).

```

1450 \AtBeginDocument
1451 {
1452     \cs_new_protected:Npx \@@_draw_dotted_lines:
1453     {
1454         \c_@@_pgfortikzpicture_tl
1455         \@@_draw_dotted_lines_i:
1456         \c_@@_endpgfortikzpicture_tl
1457     }
1458 }

```

The following command *must* be protected because it will appear in the construction of the command `\@@_draw_dotted_lines:`.

```

1459 \cs_new_protected:Npn \@@_draw_dotted_lines_i:
1460 {
1461     \pgfrememberpicturepositiononpagetrue
1462     \pgf@relevantforpicturesizefalse
1463     \g_@@_Hdotsfor_lines_tl
1464     \g_@@_Vdots_lines_tl
1465     \g_@@_Ddots_lines_tl
1466     \g_@@_Iddots_lines_tl
1467     \g_@@_Cdots_lines_tl
1468     \g_@@_Ldots_lines_tl
1469 }

1470 \cs_new_protected:Npn \@@_restore_iRow_jCol:
1471 {
1472     \cs_if_exist:NT \theiRow { \int_gset_eq:NN \c@iRow \l_@@_save_iRow_int }
1473     \cs_if_exist:NT \thejCol { \int_gset_eq:NN \c@jCol \l_@@_save_jCol_int }
1474 }

```

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.

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

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.

```
1475 \cs_new_protected:Npn \l_@@_find_extremities_of_line:nnnn #1 #2 #3 #4
1476 {
```

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

```
1477 \cs_set:cpn { @@ _ dotted _ #1 - #2 } { }
```

Initialization of variables.

```
1478 \int_set:Nn \l_@@_initial_i_int { #1 }
1479 \int_set:Nn \l_@@_initial_j_int { #2 }
1480 \int_set:Nn \l_@@_final_i_int { #1 }
1481 \int_set:Nn \l_@@_final_j_int { #2 }
```

We will do two loops: one when determining the initial cell and the other when determining the final cell. The boolean `\l_@@_stop_loop_bool` will be used to control these loops. In the first loop, we search the “final” extremity of the line.

```
1482 \bool_set_false:N \l_@@_stop_loop_bool
1483 \bool_do_until:Nn \l_@@_stop_loop_bool
1484 {
1485   \int_add:Nn \l_@@_final_i_int { #3 }
1486   \int_add:Nn \l_@@_final_j_int { #4 }
```

We test if we are still in the matrix.

```
1487   \bool_set_false:N \l_@@_final_open_bool
1488   \int_compare:nNnTF \l_@@_final_i_int > \c@iRow
1489   {
1490     \int_compare:nNnTF { #3 } = 1
1491     { \bool_set_true:N \l_@@_final_open_bool }
1492     {
1493       \int_compare:nNnT \l_@@_final_j_int > \c@jCol
1494       { \bool_set_true:N \l_@@_final_open_bool }
1495     }
1496   }
1497   {
1498     \int_compare:nNnTF \l_@@_final_j_int < 1
1499     {
1500       \int_compare:nNnT { #4 } = { -1 }
1501       { \bool_set_true:N \l_@@_final_open_bool }
1502     }
1503     {
1504       \int_compare:nNnT \l_@@_final_j_int > \c@jCol
1505       {
1506         \int_compare:nNnT { #4 } = 1
1507         { \bool_set_true:N \l_@@_final_open_bool }
1508       }
1509     }
1510   }
1511   \bool_if:NTF \l_@@_final_open_bool
```

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

```
1512 {
```

We do a step backwards.

```

1513     \int_sub:Nn \l_@@_final_i_int { #3 }
1514     \int_sub:Nn \l_@@_final_j_int { #4 }
1515     \bool_set_true:N \l_@@_stop_loop_bool
1516 }

```

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`.

```

1517 {
1518   \cs_if_exist:cTF
1519   {
1520     @@ _ dotted _
1521     \int_use:N \l_@@_final_i_int -
1522     \int_use:N \l_@@_final_j_int
1523   }
1524   {
1525     \int_sub:Nn \l_@@_final_i_int { #3 }
1526     \int_sub:Nn \l_@@_final_j_int { #4 }
1527     \bool_set_true:N \l_@@_final_open_bool
1528     \bool_set_true:N \l_@@_stop_loop_bool
1529   }
1530   {
1531     \cs_if_exist:cTF
1532     {
1533       pgf @ sh @ ns @ \@@_env:
1534       - \int_use:N \l_@@_final_i_int
1535       - \int_use:N \l_@@_final_j_int
1536     }
1537     { \bool_set_true:N \l_@@_stop_loop_bool }

```

If the case is empty, we declare that the cell as non-empty. Indeed, we will draw a dotted line and the cell will be on that dotted line. All the cells of a dotted line have to be mark as “dotted” because we don’t want intersections between dotted lines. We recall that the research of the extremities of the lines are all done in the same TeX group (the group of the environnement), even though, when the extremities are found, each line is drawn in a TeX group that we will open for the options of the line.

```

1538   {
1539     \cs_set:cpn
1540     {
1541       @@ _ dotted _
1542       \int_use:N \l_@@_final_i_int -
1543       \int_use:N \l_@@_final_j_int
1544     }
1545     { }
1546   }
1547 }
1548 }
1549 }

```

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

```

1550   \bool_set_false:N \l_@@_stop_loop_bool
1551   \bool_do_until:Nn \l_@@_stop_loop_bool
1552   {
1553     \int_sub:Nn \l_@@_initial_i_int { #3 }
1554     \int_sub:Nn \l_@@_initial_j_int { #4 }
1555     \bool_set_false:N \l_@@_initial_open_bool
1556     \int_compare:nNnTF \l_@@_initial_i_int < 1
1557     {
1558       \int_compare:nNnTF { #3 } = 1
1559       { \bool_set_true:N \l_@@_initial_open_bool }
1560     }

```

```

1561         \int_compare:nNnT \l_@@_initial_j_int = 0
1562         { \bool_set_true:N \l_@@_initial_open_bool }
1563     }
1564 }
1565 {
1566     \int_compare:nNnTF \l_@@_initial_j_int < 1
1567     {
1568         \int_compare:nNnT { #4 } = 1
1569         { \bool_set_true:N \l_@@_initial_open_bool }
1570     }
1571     {
1572         \int_compare:nNnT \l_@@_initial_j_int > \c@jCol
1573         {
1574             \int_compare:nNnT { #4 } = { -1 }
1575             { \bool_set_true:N \l_@@_initial_open_bool }
1576         }
1577     }
1578 }
1579 \bool_if:NTF \l_@@_initial_open_bool
1580 {
1581     \int_add:Nn \l_@@_initial_i_int { #3 }
1582     \int_add:Nn \l_@@_initial_j_int { #4 }
1583     \bool_set_true:N \l_@@_stop_loop_bool
1584 }
1585 {
1586     \cs_if_exist:cTF
1587     {
1588         @@ _ dotted _
1589         \int_use:N \l_@@_initial_i_int -
1590         \int_use:N \l_@@_initial_j_int
1591     }
1592     {
1593         \int_add:Nn \l_@@_initial_i_int { #3 }
1594         \int_add:Nn \l_@@_initial_j_int { #4 }
1595         \bool_set_true:N \l_@@_initial_open_bool
1596         \bool_set_true:N \l_@@_stop_loop_bool
1597     }
1598     {
1599         \cs_if_exist:cTF
1600         {
1601             pgf @ sh @ ns @ \@@_env:
1602             - \int_use:N \l_@@_initial_i_int
1603             - \int_use:N \l_@@_initial_j_int
1604         }
1605         { \bool_set_true:N \l_@@_stop_loop_bool }
1606         {
1607             \cs_set:cpn
1608             {
1609                 @@ _ dotted _
1610                 \int_use:N \l_@@_initial_i_int -
1611                 \int_use:N \l_@@_initial_j_int
1612             }
1613             { }
1614         }
1615     }
1616 }
1617 }
1618 }

1619 \cs_new:Nn \@@_initial_cell:
1620 { \@@_env: - \int_use:N \l_@@_initial_i_int - \int_use:N \l_@@_initial_j_int }
1621 \cs_new:Nn \@@_final_cell:
1622 { \@@_env: - \int_use:N \l_@@_final_i_int - \int_use:N \l_@@_final_j_int }
1623 \cs_new_protected:Npn \@@_set_initial_coords:

```

```

1624 {
1625   \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
1626   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
1627 }
1628 \cs_new_protected:Npn \@@_set_final_coords:
1629 {
1630   \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
1631   \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
1632 }
1633 \cs_new_protected:Npn \@@_set_initial_coords_from_anchor:n #1
1634 {
1635   \pgfpointanchor \@@_initial_cell: { #1 }
1636   \@@_set_initial_coords:
1637 }
1638 \cs_new_protected:Npn \@@_set_final_coords_from_anchor:n #1
1639 {
1640   \pgfpointanchor \@@_final_cell: { #1 }
1641   \@@_set_final_coords:
1642 }

```

The first and the second arguments are the coordinates of the cell where the command has been issued. The third argument is the list of the options.

```

1643 \cs_new_protected:Npn \@@_draw_Ldots:nnn #1 #2 #3
1644 {
1645   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1646   {
1647     \@@_find_extremities_of_line:nnnn { #1 } { #2 } 0 1

```

The previous command may have changed the current environment by marking some cells as “dotted”, but, fortunately, it is outside the group for the options of the line.

```

1648   \group_begin:
1649     \int_compare:nNnTF { #1 } = 0
1650     { \color { nicematrix-first-row } }
1651     {

```

We remind that, when there is a “last row” `\l_@@_last_row_int` will always be (after the construction of the array) the number of that “last row” even if the option `last-row` has been used without value.

```

1652       \int_compare:nNnT { #1 } = \l_@@_last_row_int
1653       { \color { nicematrix-last-row } }
1654     }
1655     \keys_set:nn { NiceMatrix / xdots } { #3 }
1656     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
1657     \@@_actually_draw_Ldots:
1658   \group_end:
1659 }
1660 }

```

The command `\@@_actually_draw_Ldots:` has the following implicit arguments:

- `\l_@@_initial_i_int`
- `\l_@@_initial_j_int`
- `\l_@@_initial_open_bool`
- `\l_@@_final_i_int`
- `\l_@@_final_j_int`
- `\l_@@_final_open_bool`.

The following function is also used by `\Hdotsfor`.

```

1661 \cs_new_protected:Npn \@@_actually_draw_Ldots:
1662 {
1663   \bool_if:NTF \l_@@_initial_open_bool
1664   {
1665     \@@_qpoint: { col - \int_use:N \l_@@_initial_j_int }
1666     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
1667     \dim_add:Nn \l_@@_x_initial_dim \arraycolsep
1668     \@@_qpoint: { row - \int_use:N \l_@@_initial_i_int - base }
1669     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
1670   }
1671   { \@@_set_initial_coords_from_anchor:n { base-east } }
1672   \bool_if:NTF \l_@@_final_open_bool
1673   {
1674     \@@_qpoint: { col - \@@_succ:n \l_@@_final_j_int }
1675     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
1676     \dim_sub:Nn \l_@@_x_final_dim \arraycolsep
1677     \@@_qpoint: { row - \int_use:N \l_@@_final_i_int - base }
1678     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
1679   }
1680   { \@@_set_final_coords_from_anchor:n { base-west } }

```

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

```

1681   \dim_add:Nn \l_@@_y_initial_dim \l_@@_radius_dim
1682   \dim_add:Nn \l_@@_y_final_dim \l_@@_radius_dim
1683   \@@_draw_line:
1684 }

```

The first and the second arguments are the coordinates of the cell where the command has been issued. The third argument is the list of the options.

```

1685 \cs_new_protected:Npn \@@_draw_Cdots:nnn #1 #2 #3
1686 {
1687   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1688   {
1689     \@@_find_extremities_of_line:nnnn { #1 } { #2 } 0 1

```

The previous command may have changed the current environment by marking some cells as “dotted”, but, fortunately, it is outside the group for the options of the line.

```

1690     \group_begin:
1691     \int_compare:nNnTF { #1 } = 0
1692     { \color { nicematrix-first-row } }
1693     {

```

We remind that, when there is a “last row” `\l_@@_last_row_int` will always be (after the construction of the array) the number of that “last row” even if the option `last-row` has been used without value.

```

1694         \int_compare:nNnT { #1 } = \l_@@_last_row_int
1695         { \color { nicematrix-last-row } }
1696     }
1697     \keys_set:nn { NiceMatrix / xdots } { #3 }
1698     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
1699     \@@_actually_draw_Cdots:
1700   \group_end:
1701 }
1702 }

```

The command `\@@_actually_draw_Cdots:` has the following implicit arguments:

- `\l_@@_initial_i_int`
- `\l_@@_initial_j_int`
- `\l_@@_initial_open_bool`

- \l_@@_final_i_int
- \l_@@_final_j_int
- \l_@@_final_open_bool.

```

1703 \cs_new_protected:Npn \@@_actually_draw_Cdots:
1704 {
1705   \bool_if:NTF \l_@@_initial_open_bool
1706   {
1707     \@@_qpoint: { col - \int_use:N \l_@@_initial_j_int }
1708     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
1709     \dim_add:Nn \l_@@_x_initial_dim \arraycolsep
1710   }
1711   { \@@_set_initial_coords_from_anchor:n { mid-east } }
1712   \bool_if:NTF \l_@@_final_open_bool
1713   {
1714     \@@_qpoint: { col - \@@_succ:n \l_@@_final_j_int }
1715     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
1716     \dim_sub:Nn \l_@@_x_final_dim \arraycolsep
1717   }
1718   { \@@_set_final_coords_from_anchor:n { mid-west } }
1719   \bool_lazy_and:nnTF
1720     \l_@@_initial_open_bool
1721     \l_@@_final_open_bool
1722   {
1723     \@@_qpoint: { row - \int_use:N \l_@@_initial_i_int }
1724     \dim_set_eq:NN \l_tmpa_dim \pgf@y
1725     \@@_qpoint: { row - \@@_succ:n \l_@@_initial_i_int }
1726     \dim_set:Nn \l_@@_y_initial_dim { ( \l_tmpa_dim + \pgf@y ) / 2 }
1727     \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim
1728   }
1729   {
1730     \bool_if:NT \l_@@_initial_open_bool
1731     { \dim_set_eq:NN \l_@@_y_initial_dim \l_@@_y_final_dim }
1732     \bool_if:NT \l_@@_final_open_bool
1733     { \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim }
1734   }
1735   \@@_draw_line:
1736 }

```

The first and the second arguments are the coordinates of the cell where the command has been issued. The third argument is the list of the options.

```

1737 \cs_new_protected:Npn \@@_draw_Vdots:nnn #1 #2 #3
1738 {
1739   \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
1740   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1741   {
1742     \@@_find_extremities_of_line:nnnn { #1 } { #2 } 1 0

```

The previous command may have changed the current environment by marking some cells as “dotted”, but, fortunately, it is outside the group for the options of the line.

```

1743   \group_begin:
1744     \int_compare:nNnTF { #2 } = 0
1745     { \color { nicematrix-first-col } }
1746     {
1747       \int_compare:nNnT { #2 } = \l_@@_last_col_int
1748       { \color { nicematrix-last-col } }
1749     }
1750     \keys_set:nn { NiceMatrix / xdots } { #3 }
1751     \@@_actually_draw_Vdots:
1752   \group_end:
1753 }
1754 }

```

The command `\l_@@_actually_draw_Vdots:` has the following implicit arguments:

- `\l_@@_initial_i_int`
- `\l_@@_initial_j_int`
- `\l_@@_initial_open_bool`
- `\l_@@_final_i_int`
- `\l_@@_final_j_int`
- `\l_@@_final_open_bool.`

```
1755 \cs_new_protected:Npn \l_@@_actually_draw_Vdots:
1756 {
```

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

```
1757   \bool_set_false:N \l_tmpa_bool
1758   \bool_lazy_or:nnF \l_@@_initial_open_bool \l_@@_final_open_bool
1759   {
1760     \@@_set_initial_coords_from_anchor:n { south-west }
1761     \@@_set_final_coords_from_anchor:n { north-west }
1762     \bool_set:Nn \l_tmpa_bool
1763       { \dim_compare_p:nNn \l_@@_x_initial_dim = \l_@@_x_final_dim }
1764   }
```

Now, we try to determine whether the column is of type c (C of `{NiceArray}`) or may be considered as if.

```
1765   \bool_if:NTF \l_@@_initial_open_bool
1766   {
1767     \@@_qpoint: { row - 1 }
1768     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
1769   }
1770   { \@@_set_initial_coords_from_anchor:n { south } }
1771   \bool_if:NTF \l_@@_final_open_bool
1772   {
1773     \@@_qpoint: { row - \@@_succ:n \c@iRow }
1774     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
1775   }
1776   { \@@_set_final_coords_from_anchor:n { north } }
1777   \bool_if:NTF \l_@@_initial_open_bool
1778   {
1779     \bool_if:NTF \l_@@_final_open_bool
1780     {
1781       \@@_qpoint: { col - \int_use:N \l_@@_initial_j_int }
1782       \dim_set_eq:NN \l_tmpa_dim \pgf@x
1783       \@@_qpoint: { col - \@@_succ:n \l_@@_initial_j_int }
1784       \dim_set:Nn \l_@@_x_initial_dim { ( \pgf@x + \l_tmpa_dim ) / 2 }
1785       \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim
1786     }
1787     { \dim_set_eq:NN \l_@@_x_initial_dim \l_@@_x_final_dim }
1788   }
1789   {
1790     \bool_if:NTF \l_@@_final_open_bool
1791     { \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim }
1792     {
```

Now the case where both extremities are closed. The first conditional tests whether the column is of type c (C of `{NiceArray}`) or may be considered as if.

```
1793       \dim_compare:nNnF \l_@@_x_initial_dim = \l_@@_x_final_dim
1794       {
1795         \dim_set:Nn \l_@@_x_initial_dim
1796         {
1797           \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
```

```

1798         \l_@@_x_initial_dim \l_@@_x_final_dim
1799     }
1800     \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim
1801 }
1802 }
1803 }
1804 \@@_draw_line:
1805 }

```

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.

The first and the second arguments are the coordinates of the cell where the command has been issued. The third argument is the list of the options.

```

1806 \cs_new_protected:Npn \@@_draw_Ddots:nnn #1 #2 #3
1807 {
1808     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1809     {
1810         \@@_find_extremities_of_line:nnnn { #1 } { #2 } 1 1

```

The previous command may have changed the current environment by marking some cells as “dotted”, but, fortunately, it is outside the group for the options of the line.

```

1811     \group_begin:
1812         \keys_set:nn { NiceMatrix / xdots } { #3 }
1813         \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
1814         \@@_actually_draw_Ddots:
1815     \group_end:
1816 }
1817 }

```

The command `\@@_actually_draw_Ddots:` has the following implicit arguments:

- `\l_@@_initial_i_int`
- `\l_@@_initial_j_int`
- `\l_@@_initial_open_bool`
- `\l_@@_final_i_int`
- `\l_@@_final_j_int`
- `\l_@@_final_open_bool`.

```

1818 \cs_new_protected:Npn \@@_actually_draw_Ddots:
1819 {
1820     \bool_if:NTF \l_@@_initial_open_bool
1821     {
1822         \@@_qpoint: { row - \int_use:N \l_@@_initial_i_int }
1823         \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
1824         \@@_qpoint: { col - \int_use:N \l_@@_initial_j_int }
1825         \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
1826     }
1827     { \@@_set_initial_coords_from_anchor:n { south-east } }
1828     \bool_if:NTF \l_@@_final_open_bool
1829     {
1830         \@@_qpoint: { row - \@@_succ:n \l_@@_final_i_int }
1831         \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
1832         \@@_qpoint: { col - \@@_succ:n \l_@@_final_j_int }
1833         \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
1834     }
1835     { \@@_set_final_coords_from_anchor:n { north-west } }

```


We have retrieved the coordinates in the usual way (they are stored in `\l_@@_x_initial_dim`, etc.). If the parallelization of the diagonals is set, we will have (maybe) to adjust the fourth coordinate.

```

1836     \bool_if:NT \l_@@_parallelize_diags_bool
1837     {
1838         \int_gincr:N \g_@@_ddots_int

```

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

```

1839         \int_compare:nNnTF \g_@@_ddots_int = 1

```

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.

```

1840         {
1841             \dim_gset:Nn \g_@@_delta_x_one_dim
1842             { \l_@@_x_final_dim - \l_@@_x_initial_dim }
1843             \dim_gset:Nn \g_@@_delta_y_one_dim
1844             { \l_@@_y_final_dim - \l_@@_y_initial_dim }
1845         }

```

If the diagonal line is not the first one, we have to adjust the second extremity of the line by modifying the coordinate `\l_@@_x_initial_dim`.

```

1846         {
1847             \dim_set:Nn \l_@@_y_final_dim
1848             {
1849                 \l_@@_y_initial_dim +
1850                 ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
1851                 \dim_ratio:nn \g_@@_delta_y_one_dim \g_@@_delta_x_one_dim
1852             }
1853         }
1854     }
1855     \@@_draw_line:
1856 }

```

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

The first and the second arguments are the coordinates of the cell where the command has been issued. The third argument is the list of the options.

```

1857 \cs_new_protected:Npn \@@_draw_Iddots:nnn #1 #2 #3
1858 {
1859     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1860     {
1861         \@@_find_extremities_of_line:nnnn { #1 } { #2 } 1 { -1 }

```

The previous command may have changed the current environment by marking some cells as “dotted”, but, fortunately, it is outside the group for the options of the line.

```

1862         \group_begin:
1863             \keys_set:nn { NiceMatrix / xdots } { #3 }
1864             \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
1865             \@@_actually_draw_Iddots:
1866         \group_end:
1867     }
1868 }

```

The command `\@@_actually_draw_Iddots:` has the following implicit arguments:

- `\l_@@_initial_i_int`
- `\l_@@_initial_j_int`
- `\l_@@_initial_open_bool`
- `\l_@@_final_i_int`
- `\l_@@_final_j_int`

- `\l_@@_final_open_bool`.

```

1869 \cs_new_protected:Npn \@@_actually_draw_Iddots:
1870 {
1871   \bool_if:NTF \l_@@_initial_open_bool
1872   {
1873     \@@_qpoint: { row - \int_use:N \l_@@_initial_i_int }
1874     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
1875     \@@_qpoint: { col - \@@_succ:n \l_@@_initial_j_int }
1876     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
1877   }
1878   { \@@_set_initial_coords_from_anchor:n { south-west } }
1879   \bool_if:NTF \l_@@_final_open_bool
1880   {
1881     \@@_qpoint: { row - \@@_succ:n \l_@@_final_i_int }
1882     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
1883     \@@_qpoint: { col - \int_use:N \l_@@_final_j_int }
1884     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
1885   }
1886   { \@@_set_final_coords_from_anchor:n { north-east } }
1887   \bool_if:NT \l_@@_parallelize_diags_bool
1888   {
1889     \int_gincr:N \g_@@_iddots_int
1890     \int_compare:nNnTF \g_@@_iddots_int = 1
1891     {
1892       \dim_gset:Nn \g_@@_delta_x_two_dim
1893       { \l_@@_x_final_dim - \l_@@_x_initial_dim }
1894       \dim_gset:Nn \g_@@_delta_y_two_dim
1895       { \l_@@_y_final_dim - \l_@@_y_initial_dim }
1896     }
1897     {
1898       \dim_set:Nn \l_@@_y_final_dim
1899       {
1900         \l_@@_y_initial_dim +
1901         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
1902         \dim_ratio:nn \g_@@_delta_y_two_dim \g_@@_delta_x_two_dim
1903       }
1904     }
1905   }
1906   \@@_draw_line:
1907 }

```

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

```

1908 \NewExpandableDocumentCommand \NiceMatrixLastEnv { }
1909 { \int_use:N \g_@@_env_int }

```

The actual instructions for drawing the dotted line with Tikz

The command `\@@_draw_line:` should be used in a `{pgfpicture}`. It has six implicit arguments:

- `\l_@@_x_initial_dim`
- `\l_@@_y_initial_dim`
- `\l_@@_x_final_dim`
- `\l_@@_y_final_dim`
- `\l_@@_initial_open_bool`

- `\l_@@_final_open_bool`

```

1910 \cs_new_protected:Npn \@@_draw_line:
1911 {
1912   \pgfrememberpicturepositiononpagetrue
1913   \pgf@relevantforpicturesizefalse
1914   \tl_if_eq:NNTF \l_@@_xdots_line_style_tl \c_@@_standard_tl
1915     \@@_draw_standard_dotted_line:
1916     \@@_draw_non_standard_dotted_line:
1917 }

```

We have to do a special construction with `\exp_args:NV` to be able to put in the list of options in the correct place in the Tikz instruction.

```

1918 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:
1919 {
1920   \begin { scope }
1921   \exp_args:No \@@_draw_non_standard_dotted_line:n
1922     { \l_@@_xdots_line_style_tl , \l_@@_xdots_color_tl }
1923 }

```

We have used the fact that, in PGF, un color name can be put directly in a list of options (that's why we have put directly `\l_@@_xdots_color_tl`).

The argument of `\@@_draw_non_standard_dotted_line:n` is, in fact, the list of options.

```

1924 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:n #1
1925 {
1926   \draw
1927   [
1928     #1 ,
1929     shorten-> = \l_@@_xdots_shorten_dim ,
1930     shorten-< = \l_@@_xdots_shorten_dim ,
1931   ]
1932     ( \l_@@_x_initial_dim , \l_@@_y_initial_dim )
1933     -- node [ sloped , above ]
1934       { \c_math_toggle_token \scriptstyle \l_@@_xdots_up_tl \c_math_toggle_token }
1935     node [ sloped , below ]
1936       {
1937         \c_math_toggle_token
1938         \scriptstyle \l_@@_xdots_down_tl
1939         \c_math_toggle_token
1940       }
1941     ( \l_@@_x_final_dim , \l_@@_y_final_dim ) ;
1942   \end { scope }
1943 }

```

The command `\@@_draw_standard_dotted_line:` draws the line with our system of points (which give a dotted line with real round points).

```

1944 \cs_new_protected:Npn \@@_draw_standard_dotted_line:
1945 {

```

First, we put the labels.

```

1946   \bool_lazy_and:nnF
1947     { \tl_if_empty_p:N \l_@@_xdots_up_tl }
1948     { \tl_if_empty_p:N \l_@@_xdots_down_tl }
1949   {
1950     \pgfscope
1951     \pgftransformshift
1952     {
1953       \pgfpointlineattime { 0.5 }
1954       { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
1955       { \pgfpoint \l_@@_x_final_dim \l_@@_y_final_dim }
1956     }
1957     \pgftransformrotate
1958     {

```

```

1959         \fp_eval:n
1960         {
1961             atand
1962             (
1963                 \l_@@_y_final_dim - \l_@@_y_initial_dim ,
1964                 \l_@@_x_final_dim - \l_@@_x_initial_dim
1965             )
1966         }
1967     }
1968     \pgfnode
1969     { rectangle }
1970     { south }
1971     {
1972         \c_math_toggle_token
1973         \scriptstyle \l_@@_xdots_up_tl
1974         \c_math_toggle_token
1975     }
1976     { }
1977     { \pgfusepath { } }
1978     \pgfnode
1979     { rectangle }
1980     { north }
1981     {
1982         \c_math_toggle_token
1983         \scriptstyle \l_@@_xdots_down_tl
1984         \c_math_toggle_token
1985     }
1986     { }
1987     { \pgfusepath { } }
1988     \endpgfscope
1989 }
1990 \pgfrememberpicturepositiononpagetrue
1991 \pgf@relevantforpicturesizefalse
1992 \group_begin:

```

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.

```

1993     \dim_zero_new:N \l_@@_l_dim
1994     \dim_set:Nn \l_@@_l_dim
1995     {
1996         \fp_to_dim:n
1997         {
1998             sqrt
1999             (
2000                 ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) ^ 2
2001                 +
2002                 ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) ^ 2
2003             )
2004         }
2005     }

```

It seems that, during the first compilations, the value of `\l_@@_l_dim` may be erroneous (equal to zero or very large). We must detect these cases because they would cause errors during the drawing of the dotted line. Maybe we should also write something in the `aux` file to say that one more compilation should be done.

```

2006     \bool_lazy_or:nnF
2007     { \dim_compare_p:nNn { \dim_abs:n \l_@@_l_dim } > \c_@@_max_l_dim }
2008     { \dim_compare_p:nNn \l_@@_l_dim = \c_zero_dim }
2009     \@@_draw_standard_dotted_line_i:
2010 \group_end:
2011 }
2012 \dim_const:Nn \c_@@_max_l_dim { 50 cm }
2013 \cs_new_protected:Npn \@@_draw_standard_dotted_line_i:
2014 {

```

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

```

2015   \bool_if:NTF \l_@@_initial_open_bool
2016   {
2017     \bool_if:NTF \l_@@_final_open_bool
2018     {
2019       \int_set:Nn \l_tmpa_int
2020       { \dim_ratio:nn \l_@@_l_dim \l_@@_inter_dots_dim }
2021     }
2022     {
2023       \int_set:Nn \l_tmpa_int
2024       {
2025         \dim_ratio:nn
2026         { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
2027         \l_@@_inter_dots_dim
2028       }
2029     }
2030   }
2031   {
2032     \bool_if:NTF \l_@@_final_open_bool
2033     {
2034       \int_set:Nn \l_tmpa_int
2035       {
2036         \dim_ratio:nn
2037         { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
2038         \l_@@_inter_dots_dim
2039       }
2040     }
2041     {
2042       \int_set:Nn \l_tmpa_int
2043       {
2044         \dim_ratio:nn
2045         { \l_@@_l_dim - 2 \l_@@_xdots_shorten_dim }
2046         \l_@@_inter_dots_dim
2047       }
2048     }
2049   }

```

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

```

2050   \dim_set:Nn \l_tmpa_dim
2051   {
2052     ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2053     \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
2054   }
2055   \dim_set:Nn \l_tmpb_dim
2056   {
2057     ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
2058     \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
2059   }

```

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`.

```

2060   \int_set:Nn \l_tmpb_int
2061   {
2062     \bool_if:NTF \l_@@_initial_open_bool
2063     { \bool_if:NTF \l_@@_final_open_bool 1 0 }
2064     { \bool_if:NTF \l_@@_final_open_bool 2 1 }
2065   }

```

In the loop over the dots, the dimensions `\l_@@_x_initial_dim` and `\l_@@_y_initial_dim` will be used for the coordinates of the dots. But, before the loop, we must move until the first dot.

```

2066   \dim_gadd:Nn \l_@@_x_initial_dim

```

```

2067 {
2068   ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2069   \dim_ratio:nn
2070   { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
2071   { 2 \l_@@_l_dim }
2072   * \l_tmpb_int
2073 }
2074 \dim_gadd:Nn \l_@@_y_initial_dim
2075 {
2076   ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
2077   \dim_ratio:nn
2078   { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
2079   { 2 \l_@@_l_dim }
2080   * \l_tmpb_int
2081 }
2082 \pgf@relevantforpicturesizefalse
2083 \int_step_inline:nnn 0 \l_tmpa_int
2084 {
2085   \pgfpathcircle
2086   { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
2087   { \l_@@_radius_dim }
2088   \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
2089   \dim_add:Nn \l_@@_y_initial_dim \l_tmpb_dim
2090 }
2091 \pgfusepathqfill
2092 }

```

User commands available in the new environments

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, as for now, they are still available with an error.

The syntax of these commands uses the character `_` as embellishment and that's why we have to insert a character `_` in the *arg spec* of these commands. However, we don't know the future catcode of `_` in the main document (maybe the user will use `underscore`, and, in that case, the catcode is 13 because `underscore` activates `_`). That's why these commands will be defined in a `\AtBeginDocument` and the *arg spec* will be rescanned.

```

2093 \AtBeginDocument
2094 {
2095   \tl_set:Nn \l_@@_argspec_tl { s 0 { } E { _ ^ } { { } { } } }
2096   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
2097   \exp_args:NNV \NewDocumentCommand \@@_Ldots \l_@@_argspec_tl
2098   {
2099     \bool_if:nTF { #1 }
2100     { \@@_error:n { starred-commands } }
2101     { \@@_instruction_of_type:nn { Ldots } { #2 , down = #3 , up = #4 } }
2102     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_ldots }
2103     \bool_gset_true:N \g_@@_empty_cell_bool
2104   }

2105   \exp_args:NNV \NewDocumentCommand \@@_Cdots \l_@@_argspec_tl
2106   {
2107     \bool_if:nTF { #1 }
2108     { \@@_error:n { starred-commands } }
2109     { \@@_instruction_of_type:nn { Cdots } { #2 , down = #3 , up = #4 } }

```

```

2110     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_cdots }
2111     \bool_gset_true:N \g_@@_empty_cell_bool
2112 }

2113 \exp_args:NNV \NewDocumentCommand \@@_Vdots \l_@@_argspec_tl
2114 {
2115     \bool_if:nTF { #1 }
2116     { \@@_error:n { starred-commands } }
2117     { \@@_instruction_of_type:nn { Vdots } { #2 , down = #3 , up = #4 } }
2118     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_vdots }
2119     \bool_gset_true:N \g_@@_empty_cell_bool
2120 }

2121 \exp_args:NNV \NewDocumentCommand \@@_Ddots \l_@@_argspec_tl
2122 {
2123     \bool_if:nTF { #1 }
2124     { \@@_error:n { starred-commands } }
2125     { \@@_instruction_of_type:nn { Ddots } { #2 , down = #3 , up = #4 } }
2126     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_ddots }
2127     \bool_gset_true:N \g_@@_empty_cell_bool
2128 }

2129 \exp_args:NNV \NewDocumentCommand \@@_Iddots \l_@@_argspec_tl
2130 {
2131     \bool_if:nTF { #1 }
2132     { \@@_error:n { starred-commands } }
2133     { \@@_instruction_of_type:nn { Iddots } { #2 , down = #3 , up = #4 } }
2134     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_iddots }
2135     \bool_gset_true:N \g_@@_empty_cell_bool
2136 }
2137 }

```

End of the `\AtBeginDocument`.

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

```

2138 \cs_new_protected:Npn \@@_Hspace:
2139 {
2140     \bool_gset_true:N \g_@@_empty_cell_bool
2141     \hspace
2142 }

```

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

```

2143 \cs_set_eq:NN \@@_old_multicolumn \multicolumn
2144 \cs_new:Npn \@@_multicolumn:nnn #1 #2 #3
2145 {
2146     \@@_old_multicolumn { #1 } { #2 } { #3 }
2147     \int_compare:nNnT #1 > 1
2148     {
2149         \seq_gput_left:Nx \g_@@_multicolumn_cells_seq
2150         { \int_eval:n \c@iRow - \int_use:N \c@jCol }
2151         \seq_gput_left:Nn \g_@@_multicolumn_sizes_seq { #1 }
2152     }
2153     \int_gadd:Nn \c@jCol { #1 - 1 }
2154 }

```

The command `\@@_Hdotsfor` will be linked to `\Hdotsfor` in `{NiceArrayWithDelims}`. This command uses an optional argument (as does `\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 also the implicit cells of the `\Hdotsfor` (maybe we should modify that point).

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

```

2155 \cs_new:Npn \@@_Hdotsfor:
2156 {
2157   \multicolumn { 1 } { C } { }
2158   \@@_Hdotsfor_i
2159 }

2160 \ExplSyntaxOff
2161 \def \@tempa { 0 { } m 0 { } E { _ ^ } { { } { } } }
2162 \ExplSyntaxOn
2163 \tl_set_eq:NN \l_@@_a_signature_tl \@tempa

```

The command `\@@_Hdotsfor_i` is defined with the tools of `xparse` because it has an optional 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:`).

```

2164 \AtBeginDocument
2165 {
2166   \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
2167   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
2168   \bool_if:NTF \c_@@_draft_bool
2169   {

```

We don't put `!` before the last optionnal argument for homogeneity with `\Cdots`, etc. which have only one optional argument.

```

2170     \exp_args:NNV \NewDocumentCommand \@@_Hdotsfor_i \l_@@_argspec_tl
2171     { \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { C } { } } }
2172   }
2173   {
2174     \exp_args:NNV \NewDocumentCommand \@@_Hdotsfor_i \l_@@_argspec_tl
2175     {
2176       \tl_gput_right:Nx \g_@@_Hdotsfor_lines_tl
2177       {
2178         \@@_Hdotsfor:nnnn
2179         { \int_use:N \c@iRow }
2180         { \int_use:N \c@jCol }
2181         { #2 }
2182         { #1 , #3 , down = #4 , up = #5 }
2183       }
2184       \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { C } { } }
2185     }
2186   }
2187 }

```

Enf of `\AtBeginDocument`.

```

2188 \cs_new_protected:Npn \@@_Hdotsfor:nnnn #1 #2 #3 #4
2189 {
2190   \bool_set_false:N \l_@@_initial_open_bool
2191   \bool_set_false:N \l_@@_final_open_bool

```

For the row, it's easy.

```

2192   \int_set:Nn \l_@@_initial_i_int { #1 }
2193   \int_set_eq:NN \l_@@_final_i_int \l_@@_initial_i_int

```

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

```

2194   \int_compare:nNnTF #2 = 1
2195   {
2196     \int_set:Nn \l_@@_initial_j_int 1
2197     \bool_set_true:N \l_@@_initial_open_bool
2198   }
2199   {
2200     \cs_if_exist:cTF
2201     {
2202       pgf @ sh @ ns @ \@@_env:
2203       - \int_use:N \l_@@_initial_i_int

```



```

2204         - \int_eval:n { #2 - 1 }
2205     }
2206     { \int_set:Nn \l_@@_initial_j_int { #2 - 1 } }
2207     {
2208         \int_set:Nn \l_@@_initial_j_int { #2 }
2209         \bool_set_true:N \l_@@_initial_open_bool
2210     }
2211 }
2212 \int_compare:nNnTF { #2 + #3 - 1 } = \c@jCol
2213 {
2214     \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
2215     \bool_set_true:N \l_@@_final_open_bool
2216 }
2217 {
2218     \cs_if_exist:cTF
2219     {
2220         pgf @ sh @ ns @ \@@_env:
2221         - \int_use:N \l_@@_final_i_int
2222         - \int_eval:n { #2 + #3 }
2223     }
2224     { \int_set:Nn \l_@@_final_j_int { #2 + #3 } }
2225     {
2226         \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
2227         \bool_set_true:N \l_@@_final_open_bool
2228     }
2229 }
2230 \group_begin:
2231 \int_compare:nNnTF { #1 } = 0
2232 { \color { nicematrix-first-row } }
2233 {
2234     \int_compare:nNnT { #1 } = \g_@@_row_total_int
2235     { \color { nicematrix-last-row } }
2236 }
2237 \keys_set:nn { NiceMatrix / xdots } { #4 }
2238 \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2239 \@@_actually_draw_Ldots:
2240 \group_end:

```

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).

```

2241     \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
2242     { \cs_set:cpn { @@ _ dotted _ #1 - ##1 } { } }
2243 }

```

The control sequence `\@@_rotate:` will be linked to `\rotate` in `{NiceArrayWithDelims}`.

The command will exit three levels of groups in order to execute the command

`“\box_rotate:Nn \l_@@_cell_box { 90 }”`

just after the construction of the box `\l_@@_cell_box`.

```

2244 \cs_new_protected:Npn \@@_rotate: { \group_insert_after:N \@@_rotate_i: }
2245 \cs_new_protected:Npn \@@_rotate_i: { \group_insert_after:N \@@_rotate_ii: }
2246 \cs_new_protected:Npn \@@_rotate_ii: { \group_insert_after:N \@@_rotate_iii: }
2247 \cs_new_protected:Npn \@@_rotate_iii:
2248 {
2249     \box_rotate:Nn \l_@@_cell_box { 90 }

```

If we are in the last row, we want all the boxes composed with the command `\rotate` aligned upwards.

```

2250 \int_compare:nNnT \c@iRow = \l_@@_last_row_int
2251 {
2252     \vbox_set_top:Nn \l_@@_cell_box
2253     {
2254         \vbox_to_zero:n { }

```

0.8 ex will be the distance between the principal part of the array and our element (which is composed with \rotate.

```

2255         \skip_vertical:n { - \box_ht:N \@arstrutbox + 0.8 ex }
2256         \box_use:N \l_@@_cell_box
2257     }
2258 }
2259 }

```

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 specifications of two cells in the array (in the format *i-j*) and draws a dotted line between these cells.

First, we write a command with an argument of the format *i-j* and applies the command \int_eval:n to *i* and *j*; this must *not* be protected (and is, of course fully expandable).³⁷

```

2260 \cs_new:Npn \@@_double_int_eval:n #1-#2 \q_stop
2261 { \int_eval:n { #1 } - \int_eval:n { #2 } }
2262 \ExplSyntaxOff
2263 \def \@tempa { 0 { } m m ! 0 { } E { _ ^ } { { } { } } }
2264 \ExplSyntaxOn
2265 \tl_set_eq:NN \l_@@_a_signature_tl \@tempa

```

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

```

2266 \AtBeginDocument
2267 {
2268     \tl_set:Nn \l_@@_argspec_tl { 0 { } m m ! 0 { } E { _ ^ } { { } { } } }
2269     \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
2270     \exp_args:NNV \NewDocumentCommand \@@_line \l_@@_argspec_tl
2271     {
2272         \group_begin:
2273         \keys_set:nn { NiceMatrix / xdots } { #1 , #4 , down = #5 , up = #6 }
2274         \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2275         \use:x
2276         {
2277             \@@_line_i:nn
2278             { \@@_double_int_eval:n #2 \q_stop }
2279             { \@@_double_int_eval:n #3 \q_stop }
2280         }
2281         \group_end:
2282     }
2283 }
2284 \bool_if:NTF \c_@@_draft_bool
2285 { \cs_new_protected:Npn \@@_line_i:nn #1 #2 { } }
2286 {
2287     \cs_new_protected:Npn \@@_line_i:nn #1 #2
2288     {
2289         \bool_set_false:N \l_@@_initial_open_bool
2290         \bool_set_false:N \l_@@_final_open_bool
2291         \bool_if:NTF
2292         {
2293             \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #1 }
2294             ||
2295             \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #2 }
2296         }

```

³⁷Indeed, we want that the user may use the command \line in `code-after` with LaTeX counters in the arguments — with the command \value.

```

2297     {
2298         \@_error:nnn { unknown~cell~for~line~in~code~after } { #1 } { #2 }
2299     }
2300     { \@_draw_line_ii:nn { #1 } { #2 } }
2301 }
2302 }
2303 \AtBeginDocument
2304 {
2305     \cs_new_protected:Npx \@_draw_line_ii:nn #1 #2
2306     {

```

We recall that, when externalization is used, `\tikzpicture` and `\endtikzpicture` (or `\pgfpicture` and `\endpgfpicture`) must be directly “visible” and that why we do this static construction of the command `\@_draw_line_ii:`.

```

2307         \c_@_pgfortikzpicture_tl
2308         \@_draw_line_iii:nn { #1 } { #2 }
2309         \c_@_endpgfortikzpicture_tl
2310     }
2311 }

```

The following command *must* be protected (it’s used in the construction of `\@_draw_line_ii:nn`).

```

2312 \cs_new_protected:Npn \@_draw_line_iii:nn #1 #2
2313 {
2314     \pgfrememberpicturepositiononpagetrue
2315     \pgfpointshapeborder { \@_env: - #1 } { \@_qpoint: { #2 } }
2316     \dim_set_eq:NN \l_@_x_initial_dim \pgf@x
2317     \dim_set_eq:NN \l_@_y_initial_dim \pgf@y
2318     \pgfpointshapeborder { \@_env: - #2 } { \@_qpoint: { #1 } }
2319     \dim_set_eq:NN \l_@_x_final_dim \pgf@x
2320     \dim_set_eq:NN \l_@_y_final_dim \pgf@y
2321     \@_draw_line:
2322 }

```

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).

The vertical rules

We give to the user the possibility to define new types of columns (with `\newcolumnntype` of `array`) for special vertical rules (*e.g.* rules thicker than the standard ones) which will not extend in the potential exterior rows of the array.

We provide the command `\OnlyMainNiceMatrix` in that goal. However, that command must be no-op outside the environments of `nicematrix` (and so the user will be allowed to use the same new type of column in the environments of `nicematrix` and in the standard environments of `array`).

That’s why we provide first a global definition of `\OnlyMainNiceMatrix`.

```

2323 \cs_set_eq:NN \OnlyMainNiceMatrix \use:n

```

Another definition of `\OnlyMainNiceMatrix` will be linked to the command in the environments of `nicematrix`. Here is that definition, called `\@_OnlyMainNiceMatrix:n`.

```

2324 \cs_new_protected:Npn \@_OnlyMainNiceMatrix:n #1
2325 {
2326     \int_compare:nNnTF \l_@_first_col_int = 0
2327     { \@_OnlyMainNiceMatrix_i:n { #1 } }
2328     {
2329         \int_compare:nNnTF \c@jCol = 0
2330         {
2331             \int_compare:nNnF \c@iRow = { -1 }
2332             { \int_compare:nNnF \c@iRow = { \l_@_last_row_int - 1 } { #1 } }
2333         }
2334         { \@_OnlyMainNiceMatrix_i:n { #1 } }

```

```

2335     }
2336 }

```

This definition may seem complicated by we must remind that the number of row `\c@iRow` is incremented in the first cell of the row, *after* an potential vertical rule on the left side of the first cell.

The command `\@@_OnlyMainNiceMatrix_i:n` is only a short-cut which is used twice in the above command. This command must *not* be protected.

```

2337 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix_i:n #1
2338 {
2339     \int_compare:nNnF \c@iRow = 0
2340     { \int_compare:nNnF \c@iRow = \l_@@_last_row_int { #1 } }
2341 }

```

Remember that `\c@iRow` is not always inferior to `\l_@@_last_row_int` because `\l_@@_last_row_int` may be equal to -2 or -1 (we can't write `\int_compare:nNnT \c@iRow < \l_@@_last_row_int`).

In fact, independently of `\OnlyMainNiceMatrix`, which is a convenience given to the user, we have to modify the behaviour of the standard specifier “|”.

Remark first that the natural way to do that would be to redefine the specifier “|” with `\newcolumnmtype`:

```

\newcolumnmtype { | } { ! { \OnlyMainNiceMatrix \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 have done a redefinition of the macro `\@arrayrule` of `array` and this redefinition will add `\@@_vline:` instead of `\vline` to the preamble (that definition is in the beginning of `{NiceArrayWithDelims}`).

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 in `\@@_OnlyMainNiceMatrix:n` must be effective in each row and not once for all when the preamble is constructed).

```

2342 \cs_new_protected:Npn \@@_vline: { \@@_OnlyMainNiceMatrix:n { \@@_vline_i: } }

```

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

```

2343 \cs_set_eq:NN \@@_vline_i: \vline

```

The command `\@@_draw_vlines` will be executed when the user uses the option `vlines` (which draws all the vlines of the array).

```

2344 \cs_new_protected:Npn \@@_draw_vlines:
2345 {
2346     \group_begin:

```

The command `\CT@arc@` is a command of color from `colortbl`.

```

2347     \bool_if:NT \c_@@_colortbl_loaded_bool \CT@arc@
2348     \pgfpicture
2349     \pgfrememberpicturerepositiononpagetrue
2350     \pgf@relevantforpicturesizefalse
2351     \pgfsetlinewidth \arrayrulewidth

```

First, we compute in `\l_tmpa_dim` the height of the rules we have to draw.

```

2352     \@@_qpoint: { row - 1 }
2353     \dim_set_eq:NN \l_tmpa_dim \pgf@y
2354     \pgfusepathqfill
2355     \@@_qpoint: { row - \@@_succ:n \c@iRow }
2356     \dim_sub:Nn \l_tmpa_dim \pgf@y
2357     \pgfusepathqfill

```

We translate vertically to take into account the potential “last row”.

```

2358     \dim_zero:N \l_tmpb_dim
2359     \int_compare:nNnT \l_@@_last_row_int > { -1 }
2360     {
2361         \dim_set_eq:NN \l_tmpb_dim \g_@@_dp_last_row_dim
2362         \dim_add:Nn \l_tmpb_dim \g_@@_ht_last_row_dim

```

We adjust the value of `\l_tmpa_dim` by the width of the horizontal rule just before the “last row”.

```

2363     \@@_qpoint: { row - \@@_succ:n\c@iRow }
2364     \dim_add:Nn \l_tmpa_dim \pgf@y
2365     \@@_qpoint: { row - \@@_succ:n \g_@@_row_total_int }
2366     \dim_sub:Nn \l_tmpa_dim \pgf@y
2367     \dim_sub:Nn \l_tmpa_dim \l_tmpb_dim
2368 }

```

Now, we can draw the lines with a loop.

```

2369 \int_step_inline:nnn
2370 { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
2371 { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@jCol } \c@jCol }
2372 {
2373   \pgfpathmoveto
2374   {
2375     \pgfpointadd
2376     { \@@_qpoint: { col - ##1 } }
2377     {
2378       \pgfpoint
2379       {
2380         -0.5 \arrayrulewidth
2381         \int_compare:nNtT { ##1 } = 1
2382         {
2383           \int_compare:nNtT \l_@@_first_col_int = 1
2384             { + \arrayrulewidth }
2385         }
2386       }
2387       { \l_tmpb_dim }
2388     }
2389   }
2390   \pgfpathlineto
2391   {
2392     \pgfpointadd
2393     { \@@_qpoint: { col - ##1 } }
2394     {
2395       \pgfpoint
2396       {
2397         -0.5 \arrayrulewidth
2398         \int_compare:nNtT { ##1 } = 1
2399         {
2400           \int_compare:nNtT \l_@@_first_col_int = 1
2401             { + \arrayrulewidth }
2402         }
2403       }
2404       { \l_tmpb_dim + \l_tmpa_dim }
2405     }
2406   }
2407 }
2408 \pgfusepathqstroke
2409 \endpgfpicture
2410 \group_end:
2411 }

```

The commands to draw dotted lines to separate columns and rows

These commands don’t use the normal nodes, the medium nor the large nodes. They only use the `col`-nodes and the `row`-nodes.

Horizontal dotted lines

The following command must *not* be protected because it’s meant to be expanded in a `\noalign`.

```

2412 \bool_if:NTF \c_@@_draft_bool

```

```

2413 { \cs_new:Npn \@@_hdottedline: { } }
2414 {
2415   \cs_new:Npn \@@_hdottedline:
2416     {
2417       \noalign { \skip_vertical:N 2\l_@@_radius_dim }
2418       \@@_hdottedline_i:
2419     }
2420 }

```

On the other side, the following command should be protected.

```

2421 \cs_new_protected:Npn \@@_hdottedline_i:
2422 {

```

We write in the code-after the instruction that will eventually draw the dotted line. It’s not possible to draw this dotted line now because we don’t know the length of the line (we don’t even know the number of columns).

```

2423   \tl_gput_right:Nx \g_@@_internal_code_after_tl
2424   { \@@_hdottedline:n { \int_use:N \c@iRow } }
2425 }

```

The command `\@@_hdottedline:n` is the command written in the code-after that will actually draw the dotted line. Its argument is the number of the row *before* which we will draw the row.

```

2426 \AtBeginDocument
2427 {

```

We recall that, when externalization is used, `\tikzpicture` and `\endtikzpicture` (or `\pgfpicture` and `\endpgfpicture`) must be directly “visible”.

```

2428   \cs_new_protected:Npx \@@_hdottedline:n #1
2429   {
2430     \bool_set_true:N \exp_not:N \l_@@_initial_open_bool
2431     \bool_set_true:N \exp_not:N \l_@@_final_open_bool
2432     \c_@@_pgfortikzpicture_tl
2433     \@@_hdottedline_i:n { #1 }
2434     \c_@@_endpgfortikzpicture_tl
2435   }
2436 }

```

The following command *must* be protected since it is used in the construction of `\@@_hdottedline:n`.

```

2437 \cs_new_protected:Npn \@@_hdottedline_i:n #1
2438 {
2439   \pgfrememberpicturepositiononpagetrue
2440   \@@_qpoint: { row - #1 }

```

We do a translation par `-\l_@@_radius_dim` because we want the dotted line to have exactly the same position as a vertical line drawn by “|” (considering the rule having a width equal to the diameter of the dots).

```

2441   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2442   \dim_sub:Nn \l_@@_y_initial_dim \l_@@_radius_dim
2443   \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim

```

The dotted line will be extended if the user uses `margin` (or `left-margin` and `right-margin`).

The aim is that, by standard the dotted line fits between square brackets (`\hline` doesn’t).

```

\begin{bNiceMatrix}
1 & 2 & 3 & 4 \\
\hline
1 & 2 & 3 & 4 \\
\hdottedline
1 & 2 & 3 & 4 \\
\end{bNiceMatrix}

```

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

But, if the user uses `margin`, the dotted line extends to have the same width as a `\hline`.

```

\begin{bNiceMatrix}[margin]
1 & 2 & 3 & 4 \\
\hline
1 & 2 & 3 & 4 \\
\hdottedline
1 & 2 & 3 & 4 \\
\end{bNiceMatrix}

```

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ \hdots & \hdots & \hdots & \hdots \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

```

2444 \@@_qpoint: { col - 1 }
2445 \dim_set:Nn \l_@@_x_initial_dim
2446 { \pgf@x + \arraycolsep - \l_@@_left_margin_dim }
2447 \@@_qpoint: { col - \@@_succ:n \c@jCol }
2448 \dim_set:Nn \l_@@_x_final_dim
2449 { \pgf@x - \arraycolsep + \l_@@_right_margin_dim }

```

For reasons purely aesthetic, we do an adjustment in the case of a rounded bracket. The correction by $0.5 \text{ } \l_@@_inter_dots_dim$ is *ad hoc* for a better result.

```

2450 \tl_set:Nn \l_tmpa_tl { ( }
2451 \tl_if_eq:NNF \l_@@_left_delim_tl \l_tmpa_tl
2452 { \dim_gadd:Nn \l_@@_x_initial_dim { 0.5 \l_@@_inter_dots_dim } }
2453 \tl_set:Nn \l_tmpa_tl { ) }
2454 \tl_if_eq:NNF \l_@@_right_delim_tl \l_tmpa_tl
2455 { \dim_gsub:Nn \l_@@_x_final_dim { 0.5 \l_@@_inter_dots_dim } }

```

As for now, we have no option to control the style of the lines drawn by `\hdottedline` and the specifier “.” in the preamble. That’s why we impose the style `standard`.

```

2456 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
2457 \@@_draw_line:
2458 }

```

Vertical dotted lines

```

2459 \bool_if:nTF \c_@@_draft_bool
2460 { \cs_new_protected:Npn \@@_vdottedline:n #1 { } }
2461 {
2462   \cs_new_protected:Npn \@@_vdottedline:n #1
2463   {
2464     \bool_set_true:N \l_@@_initial_open_bool
2465     \bool_set_true:N \l_@@_final_open_bool

```

We recall that, when externalization is used, `\tikzpicture` and `\endtikzpicture` (or `\pgfpicture` and `\endpgfpicture`) must be directly “visible”.

```

2466 \bool_if:NTF \c_@@_tikz_loaded_bool
2467 {
2468   \tikzpicture
2469   \@@_vdottedline_i:n { #1 }
2470   \endtikzpicture
2471 }
2472 {
2473   \pgfpicture
2474   \@@_vdottedline_i:n { #1 }
2475   \endpgfpicture
2476 }
2477 }
2478 }

```

```

2479 \cs_new_protected:Npn \@@_vdottedline_i:n #1
2480 {

```

The command `\CT@arc@` is a command of color from `colortbl`.

```

2481 \bool_if:NT \c_@@_colortbl_loaded_bool \CT@arc@
2482 \pgfrememberpicturepositiononpagetrue
2483 \@@_qpoint: { col - \int_eval:n { #1 + 1 } }

```

We do a translation par `- \l_@@_radius_dim` because we want the dotted line to have exactly the same position as a vertical rule drawn by “|” (considering the rule having a width equal to the diameter of the dots).

```
2484 \dim_set:Nn \l_@@_x_initial_dim { \pgf@x - \l_@@_radius_dim }
2485 \dim_set:Nn \l_@@_x_final_dim { \pgf@x - \l_@@_radius_dim }
2486 \@@_qpoint: { row - 1 }
```

We arbitrary decrease the height of the dotted line by a quantity equal to `\l_@@_inter_dots_dim` in order to improve the visual impact.

```
2487 \dim_set:Nn \l_@@_y_initial_dim { \pgf@y - 0.5 \l_@@_inter_dots_dim }
2488 \@@_qpoint: { row - \@@_succ:n \c@iRow }
2489 \dim_set:Nn \l_@@_y_final_dim { \pgf@y + 0.5 \l_@@_inter_dots_dim }
```

As for now, we have no option to control the style of the lines drawn by `\hdottedline` and the specifier “:” in the preamble. That’s why we impose the style `standard`.

```
2490 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
2491 \@@_draw_line:
2492 }
```

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.

```
2493 \bool_new:N \l_@@_block_auto_columns_width_bool
```

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

```
2494 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
2495 {
2496   auto-columns-width .code:n =
2497   {
2498     \bool_set_true:N \l_@@_block_auto_columns_width_bool
2499     \dim_gzero_new:N \g_@@_max_cell_width_dim
2500     \bool_set_true:N \l_@@_auto_columns_width_bool
2501   }
2502 }

2503 \NewDocumentEnvironment { NiceMatrixBlock } { ! 0 { } }
2504 {
2505   \int_gincr:N \g_@@_NiceMatrixBlock_int
2506   \dim_zero:N \l_@@_columns_width_dim
2507   \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
2508   \bool_if:NT \l_@@_block_auto_columns_width_bool
2509   {
2510     \cs_if_exist:cT { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
2511     {
2512       \exp_args:NNc \dim_set:Nn \l_@@_columns_width_dim
2513       { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
2514     }
2515   }
2516 }
```

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`).

```
2517 {
2518   \bool_if:NT \l_@@_block_auto_columns_width_bool
2519   {
2520     \iow_shipout:Nn \@mainaux \ExplSyntaxOn
2521     \iow_shipout:Nx \@mainaux
```



```

2522     {
2523         \cs_gset:cpn
2524         { @@ _ max _ cell _ width _ \int_use:N \g_@@_NiceMatrixBlock_int }

```

For technical reasons, we have to include the width of an eventual rule on the right side of the cells.

```

2525         { \dim_eval:n { \g_@@_max_cell_width_dim + \arrayrulewidth } }
2526     }
2527     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
2528 }
2529 }

```

The extra nodes

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

```

2530 \cs_generate_variant:Nn \dim_min:nn { v n }
2531 \cs_generate_variant:Nn \dim_max:nn { v n }

```

We have three macros of creation of nodes: `\@@_create_medium_nodes:`, `\@@_create_large_nodes:` and `\@@_create_medium_and_large_nodes:`.

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

The command `\@@_computations_for_medium_nodes:` must be used in a `{pgfpicture}`.

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`.

```

2532 \cs_new_protected:Npn \@@_computations_for_medium_nodes:
2533 {
2534     \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
2535     {
2536         \dim_zero_new:c { l_@@_row_\@@_i: _min_dim }
2537         \dim_set_eq:cN { l_@@_row_\@@_i: _min_dim } \c_max_dim
2538         \dim_zero_new:c { l_@@_row_\@@_i: _max_dim }
2539         \dim_set:cn { l_@@_row_\@@_i: _max_dim } { - \c_max_dim }
2540     }
2541     \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
2542     {
2543         \dim_zero_new:c { l_@@_column_\@@_j: _min_dim }
2544         \dim_set_eq:cN { l_@@_column_\@@_j: _min_dim } \c_max_dim
2545         \dim_zero_new:c { l_@@_column_\@@_j: _max_dim }
2546         \dim_set:cn { l_@@_column_\@@_j: _max_dim } { - \c_max_dim }
2547     }

```

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

```

2548     \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
2549     {
2550         \int_step_variable:nnNn
2551         \l_@@_first_col_int \g_@@_col_total_int \@@_j:

```

If the cell (i - j) is empty or an implicit cell (that is to say a cell after implicit ampersands &) we don't update the dimensions we want to compute.

```

2552     {
2553         \cs_if_exist:cT
2554         { pgf @ sh @ ns @ \@@_env: - \@@_i: - \@@_j: }

```

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

```

2555     {
2556         \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { south-west }
2557         \dim_set:cn { l_@@_row_ \@@_i: _min_dim }
2558         { \dim_min:vn { l_@@_row _ \@@_i: _min_dim } \pgf@y }
2559         \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
2560         {
2561             \dim_set:cn { l_@@_column _ \@@_j: _min_dim }
2562             { \dim_min:vn { l_@@_column _ \@@_j: _min_dim } \pgf@x }
2563         }

```

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.

```

2564         \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { north-east }
2565         \dim_set:cn { l_@@_row _ \@@_i: _max_dim }
2566         { \dim_max:vn { l_@@_row _ \@@_i: _max_dim } \pgf@y }
2567         \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
2568         {
2569             \dim_set:cn { l_@@_column _ \@@_j: _max_dim }
2570             { \dim_max:vn { l_@@_column _ \@@_j: _max_dim } \pgf@x }
2571         }
2572     }
2573 }
2574 }

```

Now, we have to deal with empty rows or empty columns since we don't have created nodes in such rows and columns.

```

2575     \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
2576     {
2577         \dim_compare:nNnT
2578         { \dim_use:c { l_@@_row _ \@@_i: _min _ dim } } = \c_max_dim
2579         {
2580             \@@_qpoint: { row - \@@_i: - base }
2581             \dim_set:cn { l_@@_row _ \@@_i: _max _ dim } \pgf@y
2582             \dim_set:cn { l_@@_row _ \@@_i: _min _ dim } \pgf@y
2583         }
2584     }
2585     \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
2586     {
2587         \dim_compare:nNnT
2588         { \dim_use:c { l_@@_column _ \@@_j: _min _ dim } } = \c_max_dim
2589         {
2590             \@@_qpoint: { col - \@@_j: }
2591             \dim_set:cn { l_@@_column _ \@@_j: _max _ dim } \pgf@x
2592             \dim_set:cn { l_@@_column _ \@@_j: _min _ dim } \pgf@x
2593         }
2594     }
2595 }

```

Here is the command \@@_create_medium_nodes:. When this command is used, the “medium nodes” are created.

```

2596 \cs_new_protected:Npn \@@_create_medium_nodes:
2597 {
2598     \pgfpicture
2599     \pgfrememberpicturepositiononpagetrue
2600     \pgf@relevantforpicturesizefalse
2601     \@@_computations_for_medium_nodes:

```

Now, we can create the “medium nodes”. We use a command `\@@_create_nodes:` because this command will also be used for the creation of the “large nodes”.

```

2602     \tl_set:Nn \l_@@_suffix_tl { -medium }
2603     \@@_create_nodes:
2604     \endpgfpicture
2605 }

```

The command `\@@_create_large_nodes:` must be used when we want to create only the “large nodes” and not the medium ones³⁸. However, the computation of the mathematical coordinates of the “large nodes” needs the computation of the mathematical coordinates of the “medium nodes”. Hence, we use first `\@@_computations_for_medium_nodes:` and then the command `\@@_computations_for_large_nodes:`.

```

2606 \cs_new_protected:Npn \@@_create_large_nodes:
2607 {
2608     \pgfpicture
2609     \pgfrememberpicturepositiononpagetrue
2610     \pgf@relevantforpicturesizefalse
2611     \@@_computations_for_medium_nodes:
2612     \@@_computations_for_large_nodes:
2613     \tl_set:Nn \l_@@_suffix_tl { - large }
2614     \@@_create_nodes:
2615     \endpgfpicture
2616 }
2617 \cs_new_protected:Npn \@@_create_medium_and_large_nodes:
2618 {
2619     \pgfpicture
2620     \pgfrememberpicturepositiononpagetrue
2621     \pgf@relevantforpicturesizefalse
2622     \@@_computations_for_medium_nodes:

```

Now, we can create the “medium nodes”. We use a command `\@@_create_nodes:` because this command will also be used for the creation of the “large nodes”.

```

2623     \tl_set:Nn \l_@@_suffix_tl { - medium }
2624     \@@_create_nodes:
2625     \@@_computations_for_large_nodes:
2626     \tl_set:Nn \l_@@_suffix_tl { - large }
2627     \@@_create_nodes:
2628     \endpgfpicture
2629 }

```

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

```

2630 \cs_new_protected:Npn \@@_computations_for_large_nodes:
2631 {
2632     \int_set:Nn \l_@@_first_row_int 1
2633     \int_set:Nn \l_@@_first_col_int 1

```

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

```

2634     \int_step_variable:nNn { \c@iRow - 1 } \@@_i:
2635     {
2636         \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim }
2637         {
2638             (
2639                 \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } +
2640                 \dim_use:c { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
2641             )
2642             / 2
2643         }
2644         \dim_set:eq:cc { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }

```

³⁸If we want to create both, we have to use `\@@_create_medium_and_large_nodes:`

```

2645     { l_@@_row_ \@@_i: _min_dim }
2646   }
2647   \int_step_variable:nnN { \c@jCol - 1 } \@@_j:
2648   {
2649     \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim }
2650     {
2651       (
2652         \dim_use:c { l_@@_column _ \@@_j: _ max _ dim } +
2653         \dim_use:c
2654           { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
2655       )
2656       / 2
2657     }
2658     \dim_set_eq:cc { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
2659     { l_@@_column _ \@@_j: _ max _ dim }
2660   }
2661   % \end{macrocode}
2662   % Here, we have to use |\dim_sub:cn| because of the number 1 in the name.
2663   % \begin{macrocode}
2664   \dim_sub:cn
2665     { l_@@_column _ 1 _ min _ dim }
2666   \l_@@_left_margin_dim
2667   \dim_add:cn
2668     { l_@@_column _ \int_use:N \c@jCol _ max _ dim }
2669   \l_@@_right_margin_dim
2670 }

```

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.

The function also uses `\l_@@_suffix_tl` (-medium or -large).

```

2671 \cs_new_protected:Npn \@@_create_nodes:
2672 {
2673   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
2674   {
2675     \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
2676     {

```

We draw the rectangular node for the cell (`\@@_i-\@@_j`).

```

2677     \@@_pgf_rect_node:nnnnn
2678     { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
2679     { \dim_use:c { l_@@_column _ \@@_j: _ min_dim } }
2680     { \dim_use:c { l_@@_row _ \@@_i: _ min_dim } }
2681     { \dim_use:c { l_@@_column _ \@@_j: _ max_dim } }
2682     { \dim_use:c { l_@@_row _ \@@_i: _ max_dim } }
2683     \str_if_empty:NF \l_@@_name_str
2684     {
2685       \pgfnodealias
2686       { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
2687       { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
2688     }
2689   }
2690 }

```

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 .

```

2691 \seq_mapthread_function:NNN
2692   \g_@@_multicolumn_cells_seq
2693   \g_@@_multicolumn_sizes_seq
2694   \@@_node_for_multicolumn:nn
2695 }

```

```

2696 \cs_new_protected:Npn \@@_extract_coords_values: #1 - #2 \q_stop
2697 {
2698   \cs_set:Npn \@@_i: { #1 }
2699   \cs_set:Npn \@@_j: { #2 }
2700 }

```

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”).

```

2701 \cs_new_protected:Npn \@@_node_for_multicolumn:nn #1 #2
2702 {
2703   \@@_extract_coords_values: #1 \q_stop
2704   \@@_pgf_rect_node:nnnnn
2705     { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
2706     { \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } }
2707     { \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } }
2708     { \dim_use:c { l_@@_column _ \int_eval:n { \@@_j: +#2-1 } _ max _ dim } }
2709     { \dim_use:c { l_@@_row _ \@@_i: _ max _ dim } }
2710   \str_if_empty:NF \l_@@_name_str
2711   {
2712     \pgfnodealias
2713       { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
2714       { \int_use:N \g_@@_env_int - \@@_i: - \@@_j: \l_@@_suffix_tl }
2715   }
2716 }

```

Block matrices

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

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

```

2717 \NewDocumentCommand \@@_Block: { 0 { } m D < > { } m }
2718 { \@@_Block_i #2 \q_stop { #1 } { #3 } { #4 } }

```

The first mandatory argument of `\@@_Block:` 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.

```

2719 \cs_new:Npn \@@_Block_i #1-#2 \q_stop { \@@_Block_ii:nnnnn { #1 } { #2 } }

```

Now, the arguments have been extracted: `#1` is i (the number of rows of the block), `#2` is j (the number of columns of the block), `#3` is the list of key-values, `#4` are the tokens to put before the math mode and `#5` is the label of the block.

```

2720 \cs_new_protected:Npn \@@_Block_ii:nnnnn #1 #2 #3 #4 #5
2721 {

```

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).

```

2722   \tl_gput_left:Nx \g_@@_internal_code_after_tl
2723   {
2724     \@@_Block_iii:nnnnnn
2725     { \int_use:N \c@iRow }
2726     { \int_use:N \c@jCol }
2727     { \int_eval:n { \c@iRow + #1 - 1 } }
2728     { \int_eval:n { \c@jCol + #2 - 1 } }
2729     { #3 }
2730     \exp_not:n { { #4 $ #5 $ } }
2731   }

```

It's not allowed to use the command `\Block` twice in the same cell of the array. That's why, at the first use, we link the command `\Block` to a special version. The scope of this link is the cell of the array.

```

2732     \cs_set_eq:NN \Block \@@_Block_error:nn
2733   }

2734 \cs_new:Npn \@@_Block_error:nn #1 #2
2735   {
2736     \@@_error:n { Second~Block }
2737     \cs_set_eq:NN \Block \use:nn
2738   }

2739 \keys_define:nn { NiceMatrix / Block }
2740   {
2741     tikz .tl_set:N = \l_@@_tikz_tl ,
2742     tikz .value_required:n = true ,
2743     white .bool_set:N = \l_@@_white_bool ,
2744     white .default:n = true ,
2745     white .value_forbidden:n = true ,
2746   }

```

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

```

2747 \cs_new_protected:Npn \@@_Block_iii:nnnnnn #1 #2 #3 #4 #5 #6
2748   {

```

The group is for the keys.

```

2749     \group_begin:
2750     \keys_set:nn { NiceMatrix / Block } { #5 }

2751     \bool_if:nTF
2752       {
2753         \int_compare_p:nNn { #3 } > \c@iRow
2754         || \int_compare_p:nNn { #4 } > \c@jCol
2755       }
2756       { \msg_error:nnnn { nicematrix } { Block~too~large } { #1 } { #2 } }
2757       {

```

We put the contents of the cell in the box `\l_@@_cell_box` because we want the command `\rotate` used in the content to be able to rotate the box.

```

2758     \hbox_set:Nn \l_@@_cell_box { #6 }

```

The construction of the node corresponding to the merged cells.

```

2759     \pgfpicture
2760     \pgfrememberpicturerepositiononpagetrue
2761     \pgf@relevantforpicturesizefalse
2762     \@@_qpoint: { row - #1 }
2763     \dim_set_eq:NN \l_tmpa_dim \pgf@y
2764     \@@_qpoint: { col - #2 }
2765     \dim_set_eq:NN \l_tmpb_dim \pgf@x
2766     \@@_qpoint: { row - \@@_succ:n { #3 } }
2767     \dim_set_eq:NN \l_tmpc_dim \pgf@y
2768     \@@_qpoint: { col - \@@_succ:n { #4 } }
2769     \dim_set_eq:NN \l_tmpd_dim \pgf@x

```

The following code doesn't work for the first vertical rule. You should allow the option `white` if and only if the option `vlines` and `hlines` has been used.

```

2770     \bool_if:NT \l_@@_white_bool
2771     {
2772       \begin { pgfscope }
2773       \pgfsetfillcolor { white }

```

Usually, the vertical rules are *before* the col-nodes. But there is an exception: if there is no “first col”, the first vertical rule is after the col node.³⁹ Since we don’t want the white rectangle to erase a part of this first rule, we have to do an adjustment in this case. *after* the “col node”.

```

2774         \int_compare:nNnT { #2 } = 1
2775         {
2776             \int_compare:nNnT \l_@@_first_col_int = 1
2777             { \dim_add:Nn \l_tmpb_dim \arrayrulewidth }
2778         }
2779         \pgfpathrectanglecorners
2780         { \pgfpoint \l_tmpb_dim { \l_tmpa_dim - \arrayrulewidth } }
2781         { \pgfpoint { \l_tmpd_dim - \arrayrulewidth } \l_tmpc_dim }
2782         \pgfusepathqfill
2783         \end { pgfscope }
2784     }

```

We construct the node for the block with the name (#1-#2-block).

The function `\@@_pgf_rect_node:nnnnn` takes as arguments the name of the node and the four coordinates of two opposite corner points of the rectangle.

```

2785         \begin { pgfscope }
2786         \exp_args:Nx \pgfset { \l_@@_tikz_tl }
2787         \@@_pgf_rect_node:nnnnn
2788         { \@@_env: - #1 - #2 - block }
2789         \l_tmpb_dim \l_tmpa_dim \l_tmpd_dim \l_tmpc_dim
2790         \end { pgfscope }

```

If the creation of the “medium nodes” is required, we create a “medium node” for the block. The function `\@@_pgf_rect_node:nnnnn` takes as arguments the name of the node and two PGF points.

```

2791         \bool_if:NT \l_@@_medium_nodes_bool
2792         {
2793             \@@_pgf_rect_node:nnn
2794             { \@@_env: - #1 - #2 - block - medium }
2795             { \pgfpointanchor { \@@_env: - #1 - #2 - medium } { north-west } }
2796             { \pgfpointanchor { \@@_env: - #3 - #4 - medium } { south-east } }
2797         }

```

Now, we will put the label of the block.

```

2798         \int_compare:nNnTF { #1 } = { #3 }
2799         {

```

If the block has only one row, we want the label of the block perfectly aligned on the baseline of the row. That’s why we have constructed a `\pgfcoordinate` on the baseline of the row, in the first column of the array. Now, we retrieve the *y*-value of that node and we store it in `\l_tmpa_dim`.

```

2800         \pgfextracty \l_tmpa_dim { \@@_qpoint: { row - #1 - base } }

```

We retrieve (in `\pgf@x`) the *x*-value of the center of the block.

```

2801         \@@_qpoint: { #1 - #2 - block }

```

We put the label of the block which has been composed in `\l_@@_cell_box`.

```

2802         \pgftransformshift { \pgfpoint \pgf@x \l_tmpa_dim }
2803         \pgfnode { rectangle } { base }
2804         { \box_use_drop:N \l_@@_cell_box } { } { }
2805     }

```

If the number of rows is different of 1, we put the label of the block in the center of the node (the label of the block has been composed in `\l_@@_cell_box`).

```

2806         {
2807             \pgftransformshift { \@@_qpoint: { #1 - #2 - block } }
2808             \pgfnode { rectangle } { center }
2809             { \box_use_drop:N \l_@@_cell_box } { } { }
2810         }

```

³⁹That’s true for the vertical rules drawn by “|” due to the conception of `{array}` (of `array`) and we have managed to have the same behaviour with `vlines`.

```

2811         \endpgfpicture
2812     }
2813     \group_end:
2814 }

```

How to draw the dotted lines transparently

```

2815 \cs_set_protected:Npn \@@_renew_matrix:
2816 {
2817     \RenewDocumentEnvironment { pmatrix } { } {
2818         { \pNiceMatrix }
2819         { \endpNiceMatrix }
2820     }
2821     \RenewDocumentEnvironment { vmatrix } { } {
2822         { \vNiceMatrix }
2823         { \endvNiceMatrix }
2824     }
2825     \RenewDocumentEnvironment { Vmatrix } { } {
2826         { \VNiceMatrix }
2827         { \endVNiceMatrix }
2828     }
2829     \RenewDocumentEnvironment { bmatrix } { } {
2830         { \bNiceMatrix }
2831         { \endbNiceMatrix }
2832     }
2833 }

```

Automatic arrays

```

2833 \cs_new_protected:Npn \@@_set_size:n #1-#2 \q_stop
2834 {
2835     \int_set:Nn \l_@@_nb_rows_int { #1 }
2836     \int_set:Nn \l_@@_nb_cols_int { #2 }
2837 }
2838 \NewDocumentCommand \AutoNiceMatrixWithDelims { m m O { } m O { } m ! O { } }
2839 {
2840     \int_zero_new:N \l_@@_nb_rows_int
2841     \int_zero_new:N \l_@@_nb_cols_int
2842     \@@_set_size:n #4 \q_stop
2843     \begin { NiceArrayWithDelims } { #1 } { #2 }
2844         { * { \l_@@_nb_cols_int } { C } } [ #3 , #5 , #7 ]
2845     \int_compare:nNnT \l_@@_first_row_int = 0
2846     {
2847         \int_compare:nNnT \l_@@_first_col_int = 0 { & }
2848         \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
2849         \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
2850     }
2851     \prg_replicate:nn \l_@@_nb_rows_int
2852     {
2853         \int_compare:nNnT \l_@@_first_col_int = 0 { & }

```

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

```

2854         \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { { } #6 & } #6
2855         \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
2856     }
2857     \int_compare:nNnT \l_@@_last_row_int > { -2 }
2858     {
2859         \int_compare:nNnT \l_@@_first_col_int = 0 { & }
2860         \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
2861         \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\

```



```

2862     }
2863     \end { NiceArrayWithDelims }
2864 }
2865 \cs_set_protected:Npn \@@_define_com:nnn #1 #2 #3
2866 {
2867     \cs_set_protected:cpn { #1 AutoNiceMatrix }
2868     {
2869         \str_gset:Nx \g_@@_name_env_str { #1 AutoNiceMatrix }
2870         \AutoNiceMatrixWithDelims { #2 } { #3 }
2871     }
2872 }
2873 \@@_define_com:nnn p ( )
2874 \@@_define_com:nnn b [ ]
2875 \@@_define_com:nnn v | |
2876 \@@_define_com:nnn V \ | \ |
2877 \@@_define_com:nnn B \{ \}

```

The command `\CodeAfter`

The command `\CodeAfter` catches everything until the end of the current environment (of `nicematrix`). First, we go until the next command `\end`.

```

2878 \cs_new_protected:Npn \@@_CodeAfter:n #1 \end
2879 {
2880     \tl_gput_right:Nn \g_@@_code_after_tl { #1 }
2881     \@@_CodeAfter_i:n
2882 }

```

We catch the argument of the command `\end` (in `#1`).

```

2883 \cs_new_protected:Npn \@@_CodeAfter_i:n #1
2884 {

```

If this is really the end of the current environment (of `nicematrix`), we put back the command `\end` and its argument in the TeX flow.

```

2885     \bool_if:NTF { \str_if_eq_p:Vn \g_@@_name_env_str { #1 } }
2886     { \end { #1 } }

```

If this is not the `\end` we are looking for, we put those tokens in `\g_@@_code_after_tl` and we go on searching for the next command `\end` with a recursive call to the command `\@@_CodeAfter:n`.

```

2887     {
2888         \tl_gput_right:Nn \g_@@_code_after_tl { \end { #1 } }
2889         \@@_CodeAfter:n
2890     }
2891 }

```

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.

```

2892 \bool_new:N \c_@@_obsolete_environments_bool
2893 \keys_define:nn { NiceMatrix / Package }
2894 {
2895     renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
2896     renew-dots .value_forbidden:n = true ,
2897     renew-matrix .code:n = \@@_renew_matrix: ,
2898     renew-matrix .value_forbidden:n = true ,
2899     transparent .meta:n = { renew-dots , renew-matrix } ,
2900     transparent .value_forbidden:n = true ,
2901     obsolete-environments .bool_set:N = \c_@@_obsolete_environments_bool ,

```

```

2902     obsolete-environments .value_forbidden:n = true ,
2903     obsolete-environments .default:n = true ,
2904     starred-commands .code:n =
2905         \@@_msg_redirect_name:nn { starred~commands } { none } ,
2906     starred-commands .value_forbidden:n = true ,
2907
2908 }
2909 \ProcessKeysOptions { NiceMatrix / Package }

```

Error messages of the package

The following command converts all the elements of a sequence (which are token lists) into strings.

```

2910 \cs_new_protected:Npn \@@_convert_to_str_seq:N #1
2911 {
2912     \seq_clear:N \l_tmpa_seq
2913     \seq_map_inline:Nn #1
2914     {
2915         \seq_put_left:Nx \l_tmpa_seq { \tl_to_str:n { ##1 } }
2916     }
2917     \seq_set_eq:NN #1 \l_tmpa_seq
2918 }

```

The following command creates a sequence of strings (str) from a clist.

```

2919 \cs_new_protected:Npn \@@_set_seq_of_str_from_clist:Nn #1 #2
2920 {
2921     \seq_set_from_clist:Nn #1 { #2 }
2922     \@@_convert_to_str_seq:N #1
2923 }
2924 \@@_set_seq_of_str_from_clist:Nn \c_@@_types_of_matrix_seq
2925 {
2926     NiceMatrix ,
2927     pNiceMatrix , bNiceMatrix , vNiceMatrix, BNiceMatrix, VNiceMatrix
2928 }

```

If the user uses too much columns, the command `\@@_error_too_much_cols:` is executed. This command raises an error but try to give the best information to the user in the error message. The command `\seq_if_in:NVTF` is not expandable and that's why we can't put it in the error message itself. We have to do the test before the `\@@_fatal:n`.

```

2929 \cs_new_protected:Npn \@@_error_too_much_cols:
2930 {
2931     \seq_if_in:NVTF \c_@@_types_of_matrix_seq \g_@@_name_env_str
2932     {
2933         \int_compare:nNnTF \l_@@_last_col_int = { -2 }
2934         { \@@_fatal:n { too~much~cols~for~matrix } }
2935         {
2936             \bool_if:NF \l_@@_last_col_without_value_bool
2937             { \@@_fatal:n { too~much~cols~for~matrix~with~last~col } }
2938         }
2939     }
2940     { \@@_fatal:n { too~much~cols~for~array } }
2941 }

```

The following command must *not* be protected since it's used in an error message.

```

2942 \cs_new:Npn \@@_message_hdotsfor:
2943 {
2944     \tl_if_empty:VF \g_@@_Hdotsfor_lines_tl
2945     { ~Maybe~your~use~of~\token_to_str:N \Hdotsfor\ is~incorrect.}
2946 }

```

```

2947 \@@_msg_new:nn { too-much-cols-for-matrix-with-last-col }
2948 {
2949   You-try-to-use-more-columns-than-allowed-by-your~
2950   \@@_full_name_env:.\@@_message_hdotsfor:\ The-maximal-number-of~
2951   columns-is~\int_eval:n { \l_@@_last_col_int - 1 }~(plus~the-potential~
2952   exterior-ones).~This-error-is-fatal.
2953 }
2954 \@@_msg_new:nn { too-much-cols-for-matrix }
2955 {
2956   You-try-to-use-more-columns-than-allowed-by-your~
2957   \@@_full_name_env:.\@@_message_hdotsfor:\ Recall-that-the-maximal~
2958   number-of-columns-for-a-matrix-is-fixed-by-the-LaTeX-counter~
2959   'MaxMatrixCols'.~Its-actual-value-is~\int_use:N \c@MaxMatrixCols.~
2960   This-error-is-fatal.
2961 }

```

For the following message, remind that the test is not done after the construction of the array but in each row. That's why we have to put `\c@jCol-1` and not `\c@jCol`.

```

2962 \@@_msg_new:nn { too-much-cols-for-array }
2963 {
2964   You-try-to-use-more-columns-than-allowed-by-your~
2965   \@@_full_name_env:.\@@_message_hdotsfor:\ The-maximal-number-of-columns-is~
2966   \int_eval:n { \c@jCol - 1 }~(plus~the~potential~exterior-ones).~
2967   This-error-is-fatal.
2968 }
2969 \@@_msg_new:nn { bad-option-for-line-style }
2970 {
2971   Since-you-haven't-loaded-Tikz,~the-only-value-you-can-give-to~'line-style'~
2972   is~'standard'.~If-you-go-on,~this-option-will-be-ignored.
2973 }
2974 \@@_msg_new:nn { Unknown-option-for-xdots }
2975 {
2976   As-for~now~there-is-only-three-options-available-here:~'color',~'line-style'~
2977   and~'shorten'~(and-you-try-to-use~'\l_keys_key_str').~If-you-go-on,~
2978   this-option-will-be-ignored.
2979 }
2980 \@@_msg_new:nn { ampersand-in-light-syntax }
2981 {
2982   You-can't-use-an-ampersand~(\token_to_str &)~to-separate-columns~because
2983   ~you-have-used-the-option~'light-syntax'.~This-error-is-fatal.
2984 }
2985 \@@_msg_new:nn { double-backslash-in-light-syntax }
2986 {
2987   You-can't-use~\token_to_str:N \~to-separate-rows~because-you-have-used~
2988   the-option~'light-syntax'.~You-must-use~the-character~'\l_@@_end_of_row_tl'~
2989   (set-by-the-option~'end-of-row').~This-error-is-fatal.
2990 }
2991 \@@_msg_new:nn { starred-commands }
2992 {
2993   The-starred-versions-of~\token_to_str:N \Cdots,~\token_to_str:N \Ldots,~
2994   \token_to_str:N \Vdots,~\token_to_str:N \Ddots\ and~\token_to_str:N \Iddots\
2995   are-deprecated.~However,~you-can-go-on-for-this-time.~If-you-don't-want-to~
2996   see~this-error~we-should-load~'nicematrix'~with~the~option~
2997   'starred-commands'.
2998 }
2999 \@@_msg_new:nn { bad-value-for-baseline }
3000 {
3001   The-value-given-to~'baseline'~(\int_use:N \l_tmpa_int)~is-not~
3002   valid.~The-value-must-be-between~\int_use:N \l_@@_first_row_int\ and~
3003   \int_use:N \g_@@_row_total_int\ or-equal-to~'t',~'c'~or~'b'.\\

```

```

3004     If-you-go-on,~a-value-of-1~will-be-used.
3005 }
3006 \@@_msg_new:nn { Second-Block }
3007 {
3008     You-can't-use~\token_to_str:N \Block\ twice-in-the-same-cell-of-the-array.\\
3009     If-you-go-on,~this-command-(and-the-other)-will-be-ignored.
3010 }
3011 \@@_msg_new:nn { empty-environment }
3012 { Your~\@@_full_name_env:\ is-empty.~This-error-is-fatal. }
3013 \@@_msg_new:nn { unknown-cell-for-line-in-code-after }
3014 {
3015     Your-command~\token_to_str:N \line\{#1\}\{#2\}~in-the-'code-after'~
3016     can't-be-executed-because~a-cell-doesn't-exist.\\
3017     If-you-go-on-this-command-will-be-ignored.
3018 }
3019 \@@_msg_new:nn { last-col-non-empty-for-NiceArray }
3020 {
3021     In-the~\@@_full_name_env:,~you-must-use-the-option~
3022     'last-col'~without-value.\\
3023     However,~you-can-go-on-for-this-time~
3024     (the-value~'\l_keys_value_tl'~will-be-ignored).
3025 }
3026 \@@_msg_new:nn { Block-too-large }
3027 {
3028     You-try-to-draw-a-block-in-the-cell~#1-#2~of-your-matrix-but-the-matrix-is~
3029     too-small-for-that-block. \\
3030     If-you-go-on,~this-command-will-be-ignored.
3031 }
3032 \@@_msg_new:nn { Wrong-last-row }
3033 {
3034     You-have-used~'last-row=\int_use:N \l_@@_last_row_int'~but-your~
3035     \@@_full_name_env:\ seems-to-have~\int_use:N \c@iRow \ rows.~
3036     If-you-go-on,~the-value-of~\int_use:N \c@iRow \ will-be-used-for~
3037     last-row.~You-can-avoid-this-problem-by-using~'last-row'~
3038     without-value~(more-compilations-might-be-necessary).
3039 }
3040 \@@_msg_new:nn { Yet-in-env }
3041 {
3042     Environments~\{NiceArray\}~(or~\{NiceMatrix\},~etc.)~can't-be-nested.\\
3043     This-error-is-fatal.
3044 }
3045 \@@_msg_new:nn { Outside-math-mode }
3046 {
3047     The~\@@_full_name_env:\ can-be-used-only-in-math-mode~
3048     (and-not-in~\token_to_str:N \vcenter).\\
3049     This-error-is-fatal.
3050 }
3051 \@@_msg_new:nn { Bad-value-for-letter-for-dotted-lines }
3052 {
3053     The-value-of-key~'\tl_use:N \l_keys_key_str'~must-be-of-length-1.\\
3054     If-you-go-on,~it-will-be-ignored.
3055 }
3056 \@@_msg_new:nnn { Unknown-key-for-NiceMatrixOptions }
3057 {
3058     The-key~'\tl_use:N \l_keys_key_str'~is-unknown-for-the-command~
3059     \token_to_str:N \NiceMatrixOptions. \\
3060     If-you-go-on,~it-will-be-ignored. \\
3061     For-a-list-of-the-available-keys,~type-H~<return>.
3062 }

```

```

3063 {
3064   The~available~options~are~(in~alphabetic~order):~
3065   allow-duplicate-names,~
3066   code-for-first-col,~
3067   code-for-first-row,~
3068   code-for-last-col,~
3069   code-for-last-row,~
3070   create-extra-nodes,~
3071   create-medium-nodes,~
3072   create-large-nodes,~
3073   end-of-row,~
3074   exterior-arraycolsep,~
3075   hlines,~
3076   hvlines,~
3077   left-margin,~
3078   letter-for-dotted-lines,~
3079   light-syntax,~
3080   nullify-dots,~
3081   parallelize-diags,~
3082   renew-dots,~
3083   renew-matrix,~
3084   right-margin,~
3085   small,~
3086   transparent,~
3087   vlines,~
3088   xdots/color,~
3089   xdots/shorten~and~
3090   xdots/line-style.
3091 }
3092 \@@_msg_new:nnn { Unknown~option~for~NiceArray }
3093 {
3094   The~option~'\tl_use:N\l_keys_key_str'~is~unknown~for~the~environment~
3095   \{NiceArray\}. \\
3096   If~you~go~on,~it~will~be~ignored. \\
3097   For~a~list~of~the~available~options,~type~H~<return>.
3098 }
3099 {
3100   The~available~options~are~(in~alphabetic~order):~
3101   b,~
3102   baseline,~
3103   c,~
3104   code-after,~
3105   code-for-first-col,~
3106   code-for-first-row,~
3107   code-for-last-col,~
3108   code-for-last-row,~
3109   columns-width,~
3110   create-extra-nodes,~
3111   create-medium-nodes,~
3112   create-large-nodes,~
3113   end-of-row,~
3114   extra-left-margin,~
3115   extra-right-margin,~
3116   first-col,~
3117   first-row,~
3118   hlines,~
3119   hvlines,~
3120   last-col,~
3121   last-row,~
3122   left-margin,~
3123   light-syntax,~
3124   name,~
3125   nullify-dots,~

```

```

3126 parallelize-diags,~
3127 renew-dots,~
3128 right-margin,~
3129 small,~
3130 t,~
3131 vlines,~
3132 xdots/color,~
3133 xdots/shorten~and~
3134 xdots/line-style.
3135 }

```

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).

```

3136 \@@_msg_new:nnn { Unknown~option~for~NiceMatrix }
3137 {
3138   The~option~'\tl_use:N\l_keys_key_str'~is~unknown~for~the~
3139   \@@_full_name_env:. \\\
3140   If~you~go~on,~it~will~be~ignored. \\\
3141   For~a~list~of~the~available~options,~type~H~<return>.
3142 }
3143 {
3144   The~available~options~are~(in~alphabetic~order):~
3145   b,~
3146   baseline,~
3147   c,~
3148   code-after,~
3149   code-for-first-col,~
3150   code-for-first-row,~
3151   code-for-last-col,~
3152   code-for-last-row,~
3153   columns-width,~
3154   create-extra-nodes,~
3155   create-medium-nodes,~
3156   create-large-nodes,~
3157   end-of-row,~
3158   extra-left-margin,~
3159   extra-right-margin,~
3160   first-col,~
3161   first-row,~
3162   hlines,~
3163   hvlines,~
3164   l~(=L),~
3165   last-col,~
3166   last-row,~
3167   left-margin,~
3168   light-syntax,~
3169   name,~
3170   nullify-dots,~
3171   parallelize-diags,~
3172   r~(=R),~
3173   renew-dots,~
3174   right-margin,~
3175   small,~
3176   t,~
3177   vlines,~
3178   xdots/color,~
3179   xdots/shorten~and~
3180   xdots/line-style.
3181 }
3182 \@@_msg_new:nnn { Duplicate~name }
3183 {
3184   The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
3185   the~same~environment~name~twice.~You~can~go~on,~but,~

```

```

3186 maybe,~you-will-have-incorrect-results-especially~
3187 if~you-use~'columns-width=auto'.~If~you-don't~want~to~see~this~
3188 message~again,~use~the~option~'allow-duplicate-names'.\\
3189 For~a~list~of~the~names~already~used,~type~H~<return>. \\
3190 }
3191 {
3192 The~names~already~defined~in~this~document~are:~
3193 \seq_use:Nnnn \g_@@_names_seq { ,~ } { ,~ } { ~and~ }.
3194 }
3195 \@@_msg_new:nn { Option-auto-for-columns-width }
3196 {
3197 You~can't~give~the~value~'auto'~to~the~option~'columns-width'~here.~
3198 If~you-go-on,~the~option~will~be~ignored.
3199 }
3200 \@@_msg_new:nn { Zero~row }
3201 {
3202 There~is~a~problem.~Maybe~you~have~used~l,~c~and~r~instead~of~L,~C~
3203 and~R~in~the~preamble~of~your~environment. \\
3204 This~error~is~fatal.
3205 }

```

Obsolete environments

The following environments are loaded only when the package `nicematrix` has been loaded with the option `obsolete-environments`. However, they will be completely deleted in a future version.

```

3206 \bool_if:NT \c_@@_obsolete_environments_bool
3207 {
3208   \NewDocumentEnvironment { pNiceArrayC } { }
3209   {
3210     \int_zero:N \l_@@_last_col_int
3211     \pNiceArray
3212   }
3213   { \endpNiceArray }
3214   \NewDocumentEnvironment { bNiceArrayC } { }
3215   {
3216     \int_zero:N \l_@@_last_col_int
3217     \bNiceArray
3218   }
3219   { \endbNiceArray }
3220   \NewDocumentEnvironment { BNiceArrayC } { }
3221   {
3222     \int_zero:N \l_@@_last_col_int
3223     \BNiceArray
3224   }
3225   { \endBNiceArray }
3226   \NewDocumentEnvironment { vNiceArrayC } { }
3227   {
3228     \int_zero:N \l_@@_last_col_int
3229     \vNiceArray
3230   }
3231   { \endvNiceArray }
3232   \NewDocumentEnvironment { VNiceArrayC } { }
3233   {
3234     \int_zero:N \l_@@_last_col_int
3235     \VNiceArray
3236   }
3237   { \endVNiceArray }
3238   \NewDocumentEnvironment { pNiceArrayRC } { }
3239   {
3240     \int_zero:N \l_@@_last_col_int

```

```

3241     \int_zero:N \l_@@_first_row_int
3242     \pNiceArray
3243   }
3244   { \endpNiceArray }
3245 \NewDocumentEnvironment { bNiceArrayRC } { }
3246 {
3247   \int_zero:N \l_@@_last_col_int
3248   \int_zero:N \l_@@_first_row_int
3249   \bNiceArray
3250 }
3251 { \endbNiceArray }
3252 \NewDocumentEnvironment { BNiceArrayRC } { }
3253 {
3254   \int_zero:N \l_@@_last_col_int
3255   \int_zero:N \l_@@_first_row_int
3256   \BNiceArray
3257 }
3258 { \endBNiceArray }
3259 \NewDocumentEnvironment { vNiceArrayRC } { }
3260 {
3261   \int_zero:N \l_@@_last_col_int
3262   \int_zero:N \l_@@_first_row_int
3263   \vNiceArray
3264 }
3265 { \endvNiceArray }
3266 \NewDocumentEnvironment { VNiceArrayRC } { }
3267 {
3268   \int_zero:N \l_@@_last_col_int
3269   \int_zero:N \l_@@_first_row_int
3270   \VNiceArray
3271 }
3272 { \endVNiceArray }
3273 \NewDocumentEnvironment { NiceArrayCwithDelims } { }
3274 {
3275   \int_zero:N \l_@@_last_col_int
3276   \NiceArrayWithDelims
3277 }
3278 { \endNiceArrayWithDelims }
3279 \NewDocumentEnvironment { NiceArrayRCwithDelims } { }
3280 {
3281   \int_zero:N \l_@@_last_col_int
3282   \int_zero:N \l_@@_first_row_int
3283   \NiceArrayWithDelims
3284 }
3285 { \endNiceArrayWithDelims }
3286 }

```

16 History

Changes between versions 1.0 and 1.1

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

Changes between versions 1.1 and 1.2

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

Changes between version 1.2 and 1.3

New environment `{pNiceArrayC}` and its variants.

Correction of a bug in the definition of `{BNiceMatrix}`, `{vNiceMatrix}` and `{VNiceMatrix}` (in fact, it was a typo).

Options are now available locally in `{pNiceMatrix}` and its variants.

The names of the options are changed. The old names were names in “camel style”.

Changes between version 1.3 and 1.4

The column types `w` and `W` can now be used in the environments `{NiceArray}`, `{pNiceArrayC}` and its variants with the same meaning as in the package `array`.

New option `columns-width` to fix the same width for all the columns of the array.

Changes between version 1.4 and 2.0

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

Changes between version 2.0 and 2.1

New implementation of the environment `{pNiceArrayRC}`. With this new implementation, there is no restriction on the width of the columns.

The package `nicematrix` no longer loads `mathtools` but only `amsmath`.

Creation of “medium nodes” and “large nodes”.

Changes between version 2.1 and 2.1.1

Small corrections: for example, the option `code-for-first-row` is now available in the command `\NiceMatrixOptions`.

Following a discussion on TeX StackExchange⁴⁰, Tikz externalization is now deactivated in the environments of the package `nicematrix`.⁴¹

Changes between version 2.1 and 2.1.2

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

Changes between version 2.1.2 and 2.1.3

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

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

⁴⁰cf. tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package

⁴¹Before 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 2.1.3 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 (excepted 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 StackExchange⁴², optimization of Tikz externalization is disabled in the environments of `nicematrix` when the class `standalone` or the package `standalone` is used.

Changes between version 3.4 and 3.5

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

Changes between version 3.5 and 3.6

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

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

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

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

Changes between version 3.6 and 3.7

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

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

Changes between version 3.7 and 3.8

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

Changes between version 3.8 and 3.9

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

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

Changes between version 3.9 and 3.10

New option `light-syntax` (and `end-of-row`).

New option `dotted-lines-margin` for fine tuning of the dotted lines.

⁴²cf. tex.stackexchange.com/questions/510841/nicematrix-and-tikz-external-optimize

Changes between versions 3.10 and 3.11

Correction of a bug linked to `first-row` and `last-row`.

Changes between versions 3.11 and 3.12

Command `\rotate` in the cells of the array.

Options `vlines`, `hlines` and `hvlines`.

Option `baseline` pour `{NiceArray}` (not for the other environments).

The name of the Tikz nodes created by the command `\Block` has changed: when the command has been issued in the cell $i-j$, the name is $i-j$ -block and, if the creation of the “medium nodes” is required, a node $i-j$ -block-medium is created.

If the user try to use more columns than allowed by its environment, an error is raised by `nicematrix` (instead of a low-level error).

The package must be loaded with the option `obsolete-environments` if we want to use the deprecated environments.

Changes between versions 3.12 and 3.13

The behaviour of the command `\rotate` is improved when used in the “last row”.

The option `dotted-lines-margin` has been renamed in `xdots/shorten` and the options `xdots/color` and `xdots/line-style` have been added for a complete customization of the dotted lines.

In the environments without preamble (`{NiceMatrix}`, `{pNiceMatrix}`, etc.), it’s possible to use the options `l` (=L) or `r` (=R) to specify the type of the columns.

The starred versions of the commands `\Cdots`, `\Ldots`, `\Vdots`, `\Ddots` and `\Iddots` are deprecated since the version 3.1 of `nicematrix`. Now, one should load `nicematrix` with the option `starred-commands` to avoid an error at the compilation.

The code of `nicematrix` no longer uses Tikz but only PGF. By default, Tikz is *not* loaded by `nicematrix`.

Changes between versions 3.13 and 3.14

Correction of a bug (question 60761504 on [stackoverflow](#)).

Better error messages when the user uses `&` or `\\` when `light-syntax` is in force.

Changes between versions 3.14 and 3.15

It’s possible to put labels on the dotted lines drawn by `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots`, `\Hdotsfor` and the command `\line` in the `code-after` with the tokens `_` and `^`.

The option `baseline` is now available in all the environments of `nicematrix`. Before, it was available only in `{NiceArray}`.

New command `\CodeAfter` (in the environments of `nicematrix`).

Index

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

	Symbols	
@@ commands:		<code>@@_Cell</code> : . . . 175 , 429 , 637 , 638 , 639 , 682 , 696
<code>@@_Block</code> :	656 , 2717	<code>@@_CodeAfter:n</code> 659 , 2878 , 2889
<code>@@_Block_error:nn</code>	2732 , 2734	<code>@@_CodeAfter_i:n</code> 2881 , 2883
<code>@@_Block_i</code>	2718 , 2719	<code>@@_Ddots</code> 650 , 665 , 2121
<code>@@_Block_ii:nnnnn</code>	2719 , 2720	<code>\g_@@_Ddots_lines_tl</code> 733 , 1465
<code>@@_Block_iii:nnnnnn</code>	2724 , 2747	<code>@@_Hdotsfor</code> : 654 , 668 , 2155
<code>@@_Cdots</code>	648 , 663 , 2105	<code>@@_Hdotsfor:nnnn</code> 2178 , 2188
<code>\g_@@_Cdots_lines_tl</code>	730 , 1467	<code>@@_Hdotsfor_i</code> 2158 , 2170 , 2174
		<code>\g_@@_Hdotsfor_lines_tl</code> 735 , 1463 , 2176 , 2944

<code>\@@_Hspace:</code>	653, 2138	<code>\@@_create_medium_and_large_nodes:</code> ..	1411, 2617
<code>\@@_Iddots</code>	651, 666, 2129	<code>\@@_create_medium_nodes:</code>	1412, 2596
<code>\g_@@_Iddots_lines_tl</code>	734, 1466	<code>\@@_create_nodes:</code> 2603, 2614, 2624, 2627, 2671	
<code>\@@_Ldots</code>	647, 662, 667, 2097	<code>\@@_ddots</code>	643, 2126
<code>\g_@@_Ldots_lines_tl</code>	731, 1468	<code>\g_@@_ddots_int</code>	1401, 1838, 1839
<code>\l_@@_NiceArray_bool</code>	79,	<code>\@@_define_com:nnn</code>	2865, 2873, 2874, 2875, 2876, 2877
<code>\g_@@_NiceMatrixBlock_int</code>	75, 2505, 2510, 2513, 2524	<code>\@@_define_env:n</code>	1323, 1347, 1348, 1349, 1350, 1351, 1352
<code>\@@_OnlyMainNiceMatrix:n</code> ..	658, 2324, 2342	<code>\g_@@_delta_x_one_dim</code>	1403, 1841, 1851
<code>\@@_OnlyMainNiceMatrix_i:n</code>	2327, 2334, 2337	<code>\g_@@_delta_x_two_dim</code>	1405, 1892, 1902
<code>\@@_Vdots</code>	649, 664, 2113	<code>\g_@@_delta_y_one_dim</code>	1404, 1843, 1851
<code>\g_@@_Vdots_lines_tl</code>	732, 1464	<code>\g_@@_delta_y_two_dim</code>	1406, 1894, 1902
<code>\l_@@_a_signature_tl</code>	2163, 2265	<code>\@@_double_int_eval:n</code>	2260, 2278, 2279
<code>\@@_actually_draw_Cdots:</code>	1699, 1703	<code>\g_@@_dp_ante_last_row_dim</code>	457, 629
<code>\@@_actually_draw_Ddots:</code>	1814, 1818	<code>\g_@@_dp_last_row_dim</code>	457, 458, 632, 633, 768, 769, 949, 2361
<code>\@@_actually_draw_Iddots:</code>	1865, 1869	<code>\g_@@_dp_row_zero_dim</code>	477, 478, 623, 624, 903, 932, 942
<code>\@@_actually_draw_Ldots:</code> ..	1657, 1661, 2239	<code>\c_@@_draft_bool</code>	17, 18, 56, 533, 2168, 2284, 2412, 2459
<code>\@@_actually_draw_Vdots:</code>	1751, 1755	<code>\@@_draw_Cdots:nnn</code>	1685
<code>\@@_adapt_S_column:</code>	150, 165, 744	<code>\@@_draw_Ddots:nnn</code>	1806
<code>\@@_after_array:</code>	979, 1353	<code>\@@_draw_Iddots:nnn</code>	1857
<code>\@@_analyze_end:Nn</code>	1059, 1103	<code>\@@_draw_Ldots:nnn</code>	1643
<code>\l_@@_argspec_tl</code>	2095, 2096, 2097, 2105, 2113, 2121,	<code>\@@_draw_Vdots:nnn</code>	1737
	2129, 2166, 2167, 2170, 2174, 2268, 2269, 2270	<code>\@@_draw_dotted_lines:</code>	1427, 1452
<code>\@@_array:</code>	548, 1060, 1081	<code>\@@_draw_dotted_lines_i:</code>	1455, 1459
<code>\l_@@_auto_columns_width_bool</code>	242, 324, 1134, 1138, 2500	<code>\@@_draw_line:</code>	1683,
<code>\l_@@_baseline_str</code> 234, 235, 317, 318, 319,			1735, 1804, 1855, 1906, 1910, 2321, 2457, 2491
320, 559, 892, 908, 911, 912, 913, 985, 991, 996		<code>\@@_draw_line_ii:nn</code>	2300, 2305
<code>\@@_begin_of_NiceMatrix:nn</code>	1330, 1334	<code>\@@_draw_line_iii:nn</code>	2308, 2312
<code>\@@_begin_of_row:</code>	433, 454, 1197	<code>\@@_draw_non_standard_dotted_line:</code> ..	1916, 1918
<code>\l_@@_block_auto_columns_width_bool</code> ..	755, 1139, 2493, 2498, 2508, 2518	<code>\@@_draw_non_standard_dotted_line:n</code> ..	1921, 1924
<code>\@@_cdots</code>	641, 2110	<code>\@@_draw_standard_dotted_line:</code> ..	1915, 1944
<code>\l_@@_cell_box</code> ..	435, 478, 480, 486, 495,	<code>\@@_draw_standard_dotted_line_i:</code> 2009, 2013	
499, 503, 505, 522, 598, 681, 689, 695, 704,		<code>\@@_draw_vlines:</code>	1428, 2344
767, 769, 1198, 1220, 1223, 1225, 1240,		<code>\g_@@_empty_cell_bool</code>	128, 497,
1261, 1265, 2249, 2252, 2256, 2758, 2804, 2809		500, 507, 2103, 2111, 2119, 2127, 2135, 2140	
<code>\@@_cell_with_light_syntax:n</code>	1095, 1102	<code>\@@_end_Cell:</code> 177, 490, 637, 638, 639, 686, 700	
<code>\g_@@_cells_seq</code>	1091, 1092, 1093, 1095	<code>\l_@@_end_of_row_tl</code>	252, 253, 281, 1077, 1078, 2988
<code>\g_@@_code_after_tl</code>	114, 337, 1444, 1445, 2880, 2888	<code>\c_@@_endpgfortikzpicture_tl</code>	40, 44, 1456, 2309, 2434
<code>\l_@@_code_for_first_col_tl</code>	283, 1209	<code>\@@_env:</code>	72, 74,
<code>\l_@@_code_for_first_row_tl</code>	287, 442	463, 468, 523, 529, 570, 575, 1119, 1121,	
<code>\l_@@_code_for_last_col_tl</code>	285, 1249	1129, 1131, 1149, 1151, 1165, 1170, 1180,	
<code>\l_@@_code_for_last_row_tl</code>	289, 449	1185, 1439, 1533, 1601, 1620, 1622, 2202,	
<code>\g_@@_col_total_int</code>	434, 674,	2220, 2293, 2295, 2315, 2318, 2554, 2556,	
881, 1156, 1157, 1180, 1185, 1186, 1239,		2564, 2678, 2687, 2705, 2788, 2794, 2795, 2796	
1357, 1360, 1365, 1372, 2541, 2551, 2585, 2675		<code>\g_@@_env_int</code>	71, 72,
<code>\c_@@_colortbl_loaded_bool</code>	86, 91, 588, 613, 2347, 2481	754, 777, 780, 795, 798, 1364, 1385, 1909, 2714	
<code>\l_@@_columns_width_dim</code>	76, 325, 380, 1135, 1141, 2506, 2512	<code>\@@_error:n</code>	24, 264, 273,
<code>\g_@@_com_or_env_str</code>	105, 106, 109	379, 388, 391, 409, 414, 416, 422, 427, 876,	
<code>\@@_computations_for_large_nodes:</code> ..	2612, 2625, 2630	920, 1003, 2100, 2108, 2116, 2124, 2132, 2736	
<code>\@@_computations_for_medium_nodes:</code> ..	2532, 2601, 2611, 2622	<code>\@@_error:nn</code>	25, 332
<code>\@@_convert_to_str_seq:N</code>	2910, 2922	<code>\@@_error:nnn</code>	26, 2298
<code>\@@_create_col_nodes:</code>	1063, 1085, 1109	<code>\@@_error_too_much_cols:</code>	847, 2929
<code>\@@_create_large_nodes:</code>	1414, 2606	<code>\@@_everycr:</code>	562, 618, 621
		<code>\@@_everycr_i:</code>	562, 563

\l_@@_exterior_arraycolsep_bool	236, 376, 832, 843	1582, 1590, 1594, 1603, 1611, 1620, 1665,	
\l_@@_extra_left_margin_dim	250, 309, 855, 1228	1707, 1781, 1783, 1824, 1875, 2196, 2206, 2208	
\l_@@_extra_right_margin_dim	251, 310, 867, 1269	\l_@@_initial_open_bool 1419, 1555, 1559,	
\@@_extract_coords_values:	2696, 2703	1562, 1569, 1575, 1579, 1595, 1663, 1705,	
\@@_fatal:n	27, 83, 746,	1720, 1730, 1758, 1765, 1777, 1820, 1871,	
1068, 1072, 1074, 1106, 1112, 2934, 2937, 2940		2015, 2062, 2190, 2197, 2209, 2289, 2430, 2464	
\@@_fatal:nn	28	\l_@@_initial_suffix_tl	118
\l_@@_final_anchor_tl	121	\@@_instruction_of_type:nn	
\@@_final_cell:	1621, 1640 534, 536, 2101, 2109, 2117, 2125, 2133	
\l_@@_final_i_int		\l_@@_inter_dots_dim	
1417, 1480, 1485, 1488, 1513, 1521, 1525,		... 98, 99, 1424, 2020, 2027, 2038, 2046,	
1534, 1542, 1622, 1677, 1830, 1881, 2193, 2221		2053, 2058, 2070, 2078, 2452, 2455, 2487, 2489	
\l_@@_final_j_int 1418, 1481, 1486, 1493,		\g_@@_internal_code_after_tl	
1498, 1504, 1514, 1522, 1526, 1535, 1543,	 113, 721, 1429, 1430, 2423, 2722	
1622, 1674, 1714, 1832, 1883, 2214, 2224, 2226		\@@_j:	2541, 2543,
\l_@@_final_open_bool		2544, 2545, 2546, 2551, 2554, 2556, 2559,	
.... 1420, 1487, 1491, 1494, 1501, 1507,		2561, 2562, 2564, 2567, 2569, 2570, 2585,	
1511, 1527, 1672, 1712, 1721, 1732, 1758,		2588, 2590, 2591, 2592, 2647, 2649, 2652,	
1771, 1779, 1790, 1828, 1879, 2017, 2032,		2654, 2658, 2659, 2675, 2678, 2679, 2681,	
2063, 2064, 2191, 2215, 2227, 2290, 2431, 2465		2686, 2687, 2699, 2705, 2706, 2708, 2713, 2714	
\l_@@_final_suffix_tl	120	\l_@@_l_dim	
\@@_find_extremities_of_line:nnnn	1475, 1647, 1689, 1742, 1810, 1861 1993, 1994, 2007, 2008, 2020, 2026,	
\l_@@_first_col_int		2037, 2045, 2053, 2058, 2070, 2071, 2078, 2079	
..... 134, 135, 339, 420, 433,		\l_@@_large_nodes_bool 245, 300, 1410, 1414	
825, 885, 1113, 2326, 2383, 2400, 2541,		\g_@@_last_col_found_bool	
2551, 2585, 2633, 2675, 2776, 2847, 2853, 2859		.. 142, 729, 882, 974, 1155, 1175, 1237, 1356	
\l_@@_first_row_int		\l_@@_last_col_int	140,
.. 132, 133, 340, 424, 672, 900, 916, 929,		141, 402, 404, 415, 423, 791, 797, 804, 836,	
940, 999, 2534, 2548, 2575, 2632, 2673,		1340, 1342, 1357, 1360, 1747, 2849, 2855,	
2845, 3002, 3241, 3248, 3255, 3262, 3269, 3282		2861, 2933, 2951, 3210, 3216, 3222, 3228,	
\@@_full_name_env:	107,	3234, 3240, 3247, 3254, 3261, 3268, 3275, 3281	
2950, 2957, 2965, 3012, 3021, 3035, 3047, 3139		\l_@@_last_col_without_value_bool ...	
\@@_hdottedline:	652, 2413, 2415 139, 401, 1358, 2936	
\@@_hdottedline:n	2424, 2428	\l_@@_last_row_int	
\@@_hdottedline:i:	2418, 2421 136, 137, 341, 425, 447, 586, 715,	
\@@_hdottedline:i:n	2433, 2437	762, 772, 779, 786, 870, 874, 877, 884, 946,	
\l_@@_hlines_bool	239, 292, 580	1079, 1080, 1205, 1206, 1246, 1247, 1379,	
\g_@@_ht_last_row_dim		1652, 1694, 2250, 2332, 2340, 2359, 2857, 3034	
..... 459, 630, 631, 766, 767, 948, 2362		\l_@@_last_row_without_value_bool ...	
\g_@@_ht_row_one_dim 485, 486, 627, 628	 138, 774, 872, 1377	
\g_@@_ht_row_zero_dim		\g_@@_last_vdotted_col_int 718, 720, 726, 728	
..... 479, 480, 625, 626, 903, 932, 943		\@@_ldots	640, 2102
\@@_i:	2534, 2536,	\l_@@_left_delim_dim 810, 814, 819, 1050, 1226	
2537, 2538, 2539, 2548, 2554, 2556, 2557,		\l_@@_left_delim_tl	739, 2451
2558, 2559, 2564, 2565, 2566, 2567, 2575,		\l_@@_left_margin_dim	
2578, 2580, 2581, 2582, 2634, 2636, 2639,	 246, 303, 854, 1227, 2446, 2666	
2640, 2644, 2645, 2673, 2678, 2680, 2682,		\l_@@_letter_for_dotted_lines_str ...	
2686, 2687, 2698, 2705, 2707, 2709, 2713, 2714	 387, 393, 394, 708, 709	
\@@_iddots	644, 2134	\l_@@_light_syntax_bool	
\g_@@_iddots_int	1402, 1889, 1890 233, 279, 857, 862, 1380	
\l_@@_in_env_bool	78, 746, 747	\@@_line	1443, 2270
\l_@@_initial_anchor_tl	119	\@@_line_i:nn	2277, 2285, 2287
\@@_initial_cell:	1619, 1635	\@@_line_with_light_syntax:n ... 1084, 1097	
\l_@@_initial_i_int ... 1415, 1478, 1553,		\@@_line_with_light_syntax:i:n	1083, 1089, 1100
1556, 1581, 1589, 1593, 1602, 1610, 1620,		\g_@@_max_cell_width_dim	
1668, 1723, 1725, 1822, 1873, 2192, 2193, 2203	 494, 495, 756, 1140, 2499, 2525	
\l_@@_initial_j_int		\l_@@_max_delimiter_width_bool 255, 278, 970	
..... 1416, 1479, 1554, 1561, 1566, 1572,		\c_@@_max_l_dim	2007, 2012
		\l_@@_medium_nodes_bool 244, 299, 1408, 2791	
		\@@_message_hdotsfor: 2942, 2950, 2957, 2965	
		\@@_msg_new:nn	
		... 29, 54, 2947, 2954, 2962, 2969, 2974,	

2980, 2985, 2991, 2999, 3006, 3011, 3013, 3019, 3026, 3032, 3040, 3045, 3051, 3195, 3200	\@@_rotate: 657, 2244
\@@_msg_new:nnn .. 30, 3056, 3092, 3136, 3182	\@@_rotate_i: 2244, 2245
\@@_msg_redirect_name:nn 31, 382, 2905	\@@_rotate_ii: 2245, 2246
\@@_multicolumn:nnn 655, 2144	\@@_rotate_iii: 2246, 2247
\g_@@_multicolumn_cells_seq 670, 2149, 2559, 2567, 2692	\g_@@_row_of_col_done_bool 117, 584, 741, 1126
\g_@@_multicolumn_sizes_seq 671, 2151, 2693	\g_@@_row_total_int 673, 883, 917, 1000, 1379, 1386, 1393, 2234, 2365, 2534, 2548, 2575, 2673, 3003
\g_@@_name_env_str 104, 110, 111, 742, 743, 1105, 1278, 1279, 1285, 1286, 1293, 1294, 1301, 1302, 1309, 1310, 1317, 1318, 1327, 1447, 2869, 2885, 2931	\g_@@_rows_seq . 1076, 1078, 1080, 1082, 1084
\l_@@_name_str 243, 334, 465, 469, 525, 528, 572, 576, 775, 784, 787, 793, 802, 805, 1120, 1121, 1130, 1131, 1150, 1151, 1167, 1171, 1182, 1186, 1367, 1371, 1388, 1392, 2683, 2686, 2710, 2713	\l_@@_save_iRow_int 115, 600, 1472
\g_@@_names_seq 77, 331, 333, 3193	\l_@@_save_jCol_int 116, 603, 1473
\l_@@_nb_cols_int 2836, 2841, 2844, 2848, 2854, 2860	\@@_set_final_coords: 1628, 1641
\l_@@_nb_rows_int 2835, 2840, 2851	\@@_set_final_coords_from_anchor:n 1638, 1680, 1718, 1761, 1776, 1835, 1886
\@@_node_for_multicolumn:nn 2694, 2701	\@@_set_initial_coords: 1623, 1636
\@@_node_for_the_cell: 504, 509, 1224, 1270	\@@_set_initial_coords_from_anchor:n 1633, 1671, 1711, 1760, 1770, 1827, 1878
\l_@@_nullify_dots_bool 241, 298, 2102, 2110, 2118, 2126, 2134	\@@_set_seq_of_str_from_clist:Nn 2919, 2924
\c_@@_obsolete_environments_bool 2892, 2901, 3206	\@@_set_size:n 2833, 2842
\@@_old_multicolumn 2143, 2146	\c_@@_siunitx_loaded_bool 143, 147, 152, 727
\l_@@_parallelize_diags_bool 237, 238, 295, 1399, 1836, 1887	\l_@@_small_bool 291, 437, 606, 1200, 1242, 1421
\@@_pgf_rect_node:nnn 209, 2793	\@@_standard_ialign: 561, 634
\@@_pgf_rect_node:nnnnn 184, 2677, 2704, 2787	\c_@@_standard_tl 130, 131, 1914, 2456, 2490
\c_@@_pgfortikzpicture_tl 39, 43, 1454, 2307, 2432	\l_@@_stop_loop_bool 1482, 1483, 1515, 1528, 1537, 1550, 1551, 1583, 1596, 1605
\@@_pre_array: 596, 809	\@@_succ:n 182, 570, 1009, 1165, 1170, 1171, 1180, 1185, 1186, 1674, 1714, 1725, 1773, 1783, 1830, 1832, 1875, 1881, 2355, 2363, 2365, 2371, 2447, 2488, 2640, 2644, 2654, 2658, 2766, 2768
\c_@@_preamble_first_col_tl 826, 1193	\l_@@_suffix_tl 2602, 2613, 2623, 2626, 2678, 2686, 2687, 2705, 2713, 2714
\c_@@_preamble_last_col_tl 837, 1233	\c_@@_table_collect_begin_tl . 160, 162, 175
\@@_pred:n 183, 1342	\c_@@_table_print_tl 163, 164, 177
\@@_put_box_in_flow: 972, 981, 1052	\@@_test_if_math_mode: 80, 745, 1287, 1295, 1303, 1311, 1319
\@@_put_box_in_flow_bis:nn 971, 1019	\l_@@_the_array_box 823, 848, 905, 909, 935, 962
\@@_put_box_in_flow_i: 987, 989	\c_@@_tikz_loaded_bool .. 33, 38, 1431, 2466
\@@_qpoint: 73, 895, 897, 924, 926, 1007, 1009, 1012, 1665, 1668, 1674, 1677, 1707, 1714, 1723, 1725, 1767, 1773, 1781, 1783, 1822, 1824, 1830, 1832, 1873, 1875, 1881, 1883, 2315, 2318, 2352, 2355, 2363, 2365, 2376, 2393, 2440, 2444, 2447, 2483, 2486, 2488, 2580, 2590, 2762, 2764, 2766, 2768, 2800, 2801, 2807	\l_@@_tikz_tl 2741, 2786
\l_@@_radius_dim 102, 103, 716, 1423, 1681, 1682, 2087, 2417, 2442, 2484, 2485	\l_@@_type_of_col_tl 405, 406, 407, 408, 1328, 1330
\l_@@_real_left_delim_dim 1021, 1036, 1051	\c_@@_types_of_matrix_seq 2924, 2931
\l_@@_real_right_delim_dim 1022, 1048, 1054	\@@_update_for_first_and_last_row: 473, 496, 764, 1218, 1259
\@@_renew_NC@rewrite@S: 168, 727	\@@_vdots 642, 2118
\l_@@_renew_dots_bool 296, 660, 2895	\@@_vdottedline:n 722, 2460, 2462
\@@_renew_matrix: 372, 2815, 2897	\@@_vdottedline_i:n 2469, 2474, 2479
\@@_restore_iRow_jCol: 1448, 1470	\@@_vline: 757, 2342
\c_@@_revtex_bool 47, 49, 52, 550	\@@_vline_i: 92, 2342, 2343
\l_@@_right_delim_dim 811, 815, 821, 1053, 1267	\l_@@_vlines_bool 240, 293, 831, 842, 849, 868, 1428
\l_@@_right_delim_tl 740, 2454	\l_@@_white_bool 2743, 2770
\l_@@_right_margin_dim 247, 305, 866, 1268, 2449, 2669	\g_@@_width_first_col_dim 249, 888, 1219, 1220
	\g_@@_width_last_col_dim 248, 976, 1260, 1261
	\l_@@_x_final_dim 124, 1630, 1675, 1676, 1715, 1716, 1763, 1785, 1787, 1791, 1793, 1798, 1800, 1833, 1842, 1850, 1884, 1893, 1901, 1941, 1955, 1964, 2000, 2052, 2068, 2319, 2448, 2455, 2485

<code>\l_@@_x_initial_dim</code>	122, 1625, 1666, 1667, 1708, 1709, 1763, 1784, 1785, 1787, 1791, 1793, 1795, 1798, 1800, 1825, 1842, 1850, 1876, 1893, 1901, 1932, 1954, 1964, 2000, 2052, 2066, 2068, 2086, 2088, 2316, 2445, 2452, 2484
<code>\l_@@_xdots_color_tl</code>	254, 267, 1656, 1698, 1739, 1813, 1864, 1922, 2238, 2274
<code>\l_@@_xdots_down_tl</code> ...	271, 1938, 1948, 1983
<code>\l_@@_xdots_line_style_tl</code>	129, 131, 263, 1914, 1922, 2456, 2490
<code>\l_@@_xdots_shorten_dim</code>	100, 101, 269, 1425, 1929, 1930, 2026, 2037, 2045
<code>\l_@@_xdots_up_tl</code>	272, 1934, 1947, 1973
<code>\l_@@_y_final_dim</code>	125, 1631, 1678, 1682, 1727, 1731, 1733, 1774, 1831, 1844, 1847, 1882, 1895, 1898, 1941, 1955, 1963, 2002, 2057, 2076, 2320, 2443, 2489
<code>\l_@@_y_initial_dim</code>	123, 1626, 1669, 1681, 1726, 1727, 1731, 1733, 1768, 1823, 1844, 1849, 1874, 1895, 1900, 1932, 1954, 1963, 2002, 2057, 2074, 2076, 2086, 2089, 2317, 2441, 2442, 2443, 2487
<code>\</code>	1073, 1100, 2849, 2855, 2861, 2987, 3003, 3008, 3016, 3022, 3029, 3042, 3048, 3053, 3059, 3060, 3095, 3096, 3139, 3140, 3188, 3189, 3203
<code>\{</code>	111, 1304, 2877, 3015, 3042, 3095
<code>\}</code>	111, 1304, 2877, 3015, 3042, 3095
<code>\ </code>	1320, 2876
<code>\sqcup</code>	2945, 2950, 2957, 2965, 2994, 3002, 3003, 3008, 3012, 3035, 3036, 3047
A	
<code>\array</code>	558
<code>\arraycolsep</code>	304, 306, 308, 609, 814, 815, 851, 887, 961, 963, 977, 1116, 1142, 1177, 1189, 1230, 1262, 1667, 1676, 1709, 1716, 2446, 2449
<code>\arrayrulewidth</code>	589, 590, 851, 852, 868, 2351, 2380, 2384, 2397, 2401, 2525, 2777, 2780, 2781
<code>\arraystretch</code>	608
<code>\AtBeginDocument</code>	34, 87, 144, 1450, 2093, 2164, 2266, 2303, 2426
<code>\AutoNiceMatrixWithDelims</code>	2838, 2870
B	
<code>\begin</code>	186, 211, 858, 859, 1336, 1920, 2663, 2772, 2785, 2843
<code>\Block</code>	656, 2732, 2737, 3008
<code>\BNiceArray</code>	3223, 3256
<code>\bNiceArray</code>	3217, 3249
<code>\BNiceMatrix</code>	2830
<code>\bNiceMatrix</code>	2827
bool commands:	
<code>\bool_do_until:Nn</code>	1483, 1551
<code>\bool_gset_false:N</code>	500, 507, 729, 741
<code>\bool_gset_true:N</code>	1126, 1237, 2103, 2111, 2119, 2127, 2135, 2140
<code>\bool_if:NTF</code>	56, 152, 437, 497, 533, 550, 580, 584, 588, 606, 613, 660, 727, 746, 755, 758, 812, 849, 857, 862, 868, 872, 890, 970, 974, 1134, 1155, 1175, 1200,
1242, 1356, 1358, 1377, 1380, 1399, 1410, 1414, 1421, 1428, 1431, 1511, 1579, 1663, 1672, 1705, 1712, 1730, 1732, 1765, 1771, 1777, 1779, 1790, 1797, 1820, 1828, 1836, 1871, 1879, 1887, 2015, 2017, 2032, 2062, 2063, 2064, 2102, 2110, 2118, 2126, 2134, 2168, 2284, 2347, 2370, 2371, 2412, 2466, 2481, 2508, 2518, 2770, 2791, 2885, 2936, 3206	
<code>\bool_if:nTF</code>	882, 914, 997, 1408, 2099, 2107, 2115, 2123, 2131, 2291, 2459, 2751
<code>\bool_lazy_all:nTF</code>	828, 839
<code>\bool_lazy_and:nnTF</code> ..	1137, 1201, 1719, 1946
<code>\bool_lazy_or:nnTF</code> ...	260, 1245, 1758, 2006
<code>\bool_lazy_or_p:nn</code>	1204
<code>\bool_new:N</code>	17, 33, 47, 78, 79, 86, 117, 128, 138, 139, 142, 143, 233, 236, 237, 239, 240, 241, 242, 244, 245, 255, 2493, 2892
<code>\bool_not_p:n</code>	831, 832, 842, 843, 1139
<code>\bool_set:Nn</code>	1762
<code>\bool_set_false:N</code>	1419, 1420, 1482, 1487, 1550, 1555, 1757, 2190, 2191, 2289, 2290
<code>\bool_set_true:N</code>	18, 38, 49, 52, 91, 147, 238, 324, 401, 747, 774, 1277, 1491, 1494, 1501, 1507, 1515, 1527, 1528, 1537, 1559, 1562, 1569, 1575, 1583, 1595, 1596, 1605, 2197, 2209, 2215, 2227, 2430, 2431, 2464, 2465, 2498, 2500
<code>\l_tmpa_bool</code>	1757, 1762, 1797
box commands:	
<code>\box_clear_new:N</code>	598, 823
<code>\box_dp:N</code>	458, 478, 624, 633, 769, 984, 1030, 1043
<code>\box_ht:N</code>	459, 480, 486, 626, 628, 631, 767, 983, 1030, 1043, 2255
<code>\box_move_up:nn</code> ...	62, 64, 66, 905, 934, 1016
<code>\box_rotate:Nn</code>	2249
<code>\box_set_dp:Nn</code>	984
<code>\box_set_ht:Nn</code>	983
<code>\box_use:N</code>	2256
<code>\box_use_drop:N</code>	499, 505, 522, 689, 704, 905, 909, 935, 962, 986, 1016, 1017, 1225, 2804, 2809
<code>\box_wd:N</code>	495, 503, 819, 821, 1037, 1049, 1220, 1223, 1261, 1265
<code>\l_tmpa_box</code>	818, 819, 820, 821, 952, 983, 984, 986, 1016, 1017, 1030, 1043
<code>\l_tmpb_box</code>	1023, 1037, 1038, 1049
C	
<code>\Cdots</code>	648, 2993
<code>\cdots</code>	641, 663
<code>\CodeAfter</code>	659
<code>\color</code>	1650, 1653, 1656, 1692, 1695, 1698, 1739, 1745, 1748, 1813, 1864, 2232, 2235, 2238, 2274
<code>\colorlet</code>	96, 97, 443, 450, 1210, 1250
<code>\cr</code>	1191
<code>\crr</code>	1111
cs commands:	
<code>\cs_generate_variant:Nn</code>	2530, 2531
<code>\cs_gset:Npn</code> ...	1364, 1371, 1385, 1392, 2523
<code>\cs_gset_eq:NN</code>	165, 617
<code>\cs_if_exist:NTF</code>	15, 599, 602, 748, 751, 777, 784, 795, 802, 1472, 1473, 1518, 1531, 1586, 1599, 2200, 2218, 2510, 2553

`\exp_args:Nnx` 1330
`\exp_args:No` 1921
`\exp_args:NV` 709, 1060, 1081, 1083
`\exp_args:Nx` 2786
`\exp_not:N` 39, 40, 43, 44, 2430, 2431
`\exp_not:n` 544, 2730
`\ExplSyntaxOff` 1375, 1396, 2160, 2262, 2527
`\ExplSyntaxOn` 1361, 1382, 2162, 2264, 2520

F

fi commands:

`\fi:` 84
`\firstline` 645
`\fontdimen` 1014

fp commands:

`\fp_eval:n` 1959
`\fp_to_dim:n` 1996

G

group commands:

`\group_begin:` 154, 1355, 1648, 1690,
1743, 1811, 1862, 1992, 2230, 2272, 2346, 2749
`\group_end:` 159, 1446, 1658, 1700,
1752, 1815, 1866, 2010, 2240, 2281, 2410, 2813
`\group_insert_after:N` 2244, 2245, 2246

H

`\halign` 635
`\hbox` 566, 959, 1145

hbox commands:

`\hbox:n` 62, 64, 67
`\hbox_overlap_left:n` 1221
`\hbox_overlap_right:n` 1263
`\hbox_set:Nn` . 818, 820, 952, 1023, 1038, 2758
`\hbox_set:Nw` .. 435, 681, 695, 848, 1198, 1240
`\hbox_set_end:` 493, 687, 701, 869, 1217, 1258
`\hbox_to_wd:nn` 202, 227
`\Hdotsfor` 654, 2945
`\hdotsfor` 668
`\hdottedline` 652
`\hfil` 702
`\hline` 645, 646
`\hrule` 589, 590
`\Hspace` 653
`\hspace` 2141
`\hss` 702

I

`\ialign` 561, 611, 634
`\Iddots` 651, 2994
`\iddots` 57, 644, 666

if commands:

`\if_mode_math:` 82
`\ifstandalone` 751

int commands:

`\int_add:Nn` 1485, 1486, 1581, 1582, 1593, 1594
`\int_compare:nNnTF`
..... 432, 433, 438, 440, 447, 475,
483, 582, 586, 713, 715, 718, 762, 772, 791,
825, 836, 870, 874, 884, 885, 900, 929, 940,
946, 1079, 1112, 1113, 1243, 1340, 1488,
1490, 1493, 1498, 1500, 1504, 1506, 1556,
1558, 1561, 1566, 1568, 1572, 1574, 1649,
1652, 1691, 1694, 1744, 1747, 1839, 1890,

2147, 2194, 2212, 2231, 2234, 2250, 2326,
2329, 2331, 2332, 2339, 2340, 2359, 2381,
2383, 2398, 2400, 2774, 2776, 2798, 2845,
2847, 2849, 2853, 2855, 2857, 2859, 2861, 2933

`\int_compare:nTF` 386

`\int_compare_p:nNn` 916, 917, 999,
1000, 1202, 1205, 1206, 1246, 1247, 2753, 2754

`\int_eval:n` 2150, 2204,
2222, 2261, 2483, 2708, 2727, 2728, 2951, 2966

`\int_gadd:Nn` 2153

`\int_gdecr:N` 882, 884

`\int_gincr:N`
. 431, 456, 754, 1161, 1238, 1838, 1889, 2505

`\int_gset:Nn` 434, 672, 728, 1154

`\int_gset_eq:NN`
..... 720, 877, 881, 883, 1239, 1472, 1473

`\int_gzero:N` 565

`\int_gzero_new:N`
..... 601, 604, 673, 674, 726, 1401, 1402

`\int_max:nn` 434

`\int_new:N` 71, 75, 115, 116, 132, 134, 136, 140

`\int_set:Nn` 133, 135, 137, 141, 402,
404, 779, 786, 797, 804, 913, 921, 993, 996,
1004, 1080, 1478, 1479, 1480, 1481, 2019,
2023, 2034, 2042, 2060, 2192, 2196, 2206,
2208, 2214, 2224, 2226, 2632, 2633, 2835, 2836

`\int_set_eq:NN` 600, 603, 994, 1357, 2193

`\int_step_inline:nnn` 2083, 2241, 2369

`\int_step_variable:nNn` 2634, 2647

`\int_step_variable:nnNn`
2534, 2541, 2548, 2550, 2575, 2585, 2673, 2675

`\int_sub:Nn` 1513, 1514, 1525, 1526, 1553, 1554

`\int_use:N` 72, 463, 468,
469, 523, 528, 529, 542, 543, 575, 576, 722,
777, 780, 795, 798, 897, 926, 1012, 1364,
1365, 1372, 1385, 1386, 1393, 1521, 1522,
1534, 1535, 1542, 1543, 1589, 1590, 1602,
1603, 1610, 1611, 1620, 1622, 1665, 1668,
1677, 1707, 1723, 1781, 1822, 1824, 1873,
1883, 1909, 2150, 2179, 2180, 2203, 2221,
2424, 2510, 2513, 2524, 2668, 2714, 2725,
2726, 2959, 3001, 3002, 3003, 3034, 3035, 3036

`\int_zero:N` 339,
340, 415, 420, 423, 424, 3210, 3216, 3222,
3228, 3234, 3240, 3241, 3247, 3248, 3254,
3255, 3261, 3262, 3268, 3269, 3275, 3281, 3282

`\int_zero_new:N`
..... 1415, 1416, 1417, 1418, 2840, 2841

`\g_tmpa_int` 1154, 1161, 1165, 1170, 1171

`\l_tmpa_int` 913, 916, 917, 921,
926, 993, 994, 996, 999, 1000, 1004, 1012,
2019, 2023, 2034, 2042, 2070, 2078, 2083, 3001

`\l_tmpb_int` 2060, 2072, 2080

iow commands:

`\iow_shipout:Nn` 1361, 1362, 1369,
1375, 1382, 1383, 1390, 1396, 2520, 2521, 2527

K

`\kern` 67

keys commands:

`\keys_define:nn` 256, 275,
315, 344, 370, 397, 411, 418, 2494, 2739, 2893

`\l_keys_key_str`
..... 15, 2977, 3053, 3058, 3094, 3138

<code>\keys_set:nn</code> 277, 396, 759, 760, 1329, 1655, 1697, 1750, 1812, 1863, 2237, 2273, 2507, 2750	<code>\pgfnode</code> ... 194, 221, 519, 1968, 1978, 2803, 2808
<code>\l_keys_value_tl</code> ... 3024, 3184	<code>\pgfnodealias</code> ... 467, 527, 574, 1121, 1131, 1151, 1169, 1184, 2685, 2712
L	<code>\pgfpathcircle</code> ... 2085
<code>\lasthline</code> ... 646	<code>\pgfpathlineto</code> ... 2390
<code>\ldots</code> ... 647, 2993	<code>\pgfpathmoveto</code> ... 2373
<code>\ldots</code> ... 640, 662	<code>\pgfpathrectanglecorners</code> ... 2779
<code>\left</code> ... 955, 1026, 1041	<code>\pgfpicture</code> ... 43, 460, 511, 568, 894, 923, 1006, 1117, 1127, 1147, 1163, 1178, 2348, 2473, 2598, 2608, 2619, 2759
legacy commands:	<code>\pgfpoint</code> ... 193, 1954, 1955, 2086, 2378, 2395, 2780, 2781, 2802
<code>\legacy_if:nTF</code> ... 328	<code>\pgfpointadd</code> ... 218, 2375, 2392
<code>\line</code> ... 1443, 3015	<code>\pgfpointanchor</code> ... 74, 1635, 1640, 2556, 2564, 2795, 2796
M	<code>\pgfpointdiff</code> ... 219
<code>\makebox</code> ... 688, 703	<code>\pgfpointlineattime</code> ... 1953
math commands:	<code>\pgfpointorigin</code> ... 464, 571, 1119, 1129, 1149, 1166, 1181
<code>\c_math_toggle_token</code> ... 436, 492, 856, 865, 954, 968, 1025, 1034, 1040, 1046, 1199, 1216, 1241, 1257, 1934, 1937, 1939, 1972, 1974, 1982, 1984	<code>\pgfpointscale</code> ... 218
<code>\mathinner</code> ... 59	<code>\pgfpointshapeborder</code> ... 2315, 2318
msg commands:	<code>\pgfrememberpicturepositiononpagetrue</code> ... 461, 513, 569, 1118, 1128, 1148, 1164, 1179, 1461, 1912, 1990, 2314, 2349, 2439, 2482, 2599, 2609, 2620, 2760
<code>\msg_error:nn</code> ... 16, 24	<code>\pgfscope</code> ... 1950
<code>\msg_error:nnn</code> ... 25	<code>\pgfset</code> ... 187, 212, 514, 2786
<code>\msg_error:nnnn</code> ... 26, 2756	<code>\pgfsetbaseline</code> ... 512
<code>\msg_fatal:nn</code> ... 27	<code>\pgfsetfillcolor</code> ... 2773
<code>\msg_fatal:nnn</code> ... 28	<code>\pgfsetlinewidth</code> ... 2351
<code>\msg_new:nnn</code> ... 9, 29	<code>\pgftransformrotate</code> ... 1957
<code>\msg_new:nnnn</code> ... 30	<code>\pgftransformshift</code> .. 193, 218, 1951, 2802, 2807
<code>\msg_redirect_name:nnn</code> ... 32	<code>\pgfusepath</code> ... 1977, 1987
<code>\msg_warning:nn</code> ... 56	<code>\pgfusepathqfill</code> ... 2091, 2354, 2357, 2782
<code>\multicolumn</code> ... 655, 2143, 2157, 2171, 2184	<code>\pgfusepathqstroke</code> ... 2408
<code>\myfiledate</code> ... 6	<code>\phantom</code> ... 2102, 2110, 2118, 2126, 2134
<code>\myfileversion</code> ... 7	<code>\pNiceArray</code> ... 3211, 3242
N	<code>\pNiceMatrix</code> ... 2818
<code>\newcolumntype</code> ... 637, 638, 639, 678, 692, 709	prg commands:
<code>\NewDocumentCommand</code> ... 395, 2097, 2105, 2113, 2121, 2129, 2170, 2174, 2270, 2717, 2838	<code>\prg_do_nothing:</code> ... 156, 165, 617
<code>\NewDocumentEnvironment</code> ... 737, 1056, 1066, 1275, 1283, 1291, 1299, 1307, 1315, 1325, 2503, 3208, 3214, 3220, 3226, 3232, 3238, 3245, 3252, 3259, 3266, 3273, 3279	<code>\prg_replicate:nn</code> ... 1156, 1157, 2171, 2184, 2848, 2851, 2854, 2860
<code>\NewExpandableDocumentCommand</code> ... 1908	<code>\ProcessKeysOptions</code> ... 2909
<code>\NiceArrayWithDelims</code> ... 1280, 1288, 1296, 1304, 1312, 1320, 3276, 3283	<code>\ProcessOptions</code> ... 20
<code>\NiceMatrixLastEnv</code> ... 1908	<code>\ProvideDocumentCommand</code> ... 57
<code>\NiceMatrixOptions</code> ... 395, 3059	<code>\ProvidesExplPackage</code> ... 4
<code>\noalign</code> ... 562, 617, 2417	Q
<code>\normalbaselines</code> ... 605	quark commands:
<code>\nulldelimiterspace</code> ... 1037, 1049	<code>\q_stop</code> ... 2260, 2278, 2279, 2696, 2703, 2718, 2719, 2833, 2842
<code>\numexpr</code> ... 182, 183	R
O	<code>\relax</code> ... 20, 182, 183, 676, 677
<code>\omit</code> ... 1115, 1125, 1160	<code>\renewcommand</code> ... 170
<code>\OnlyMainNiceMatrix</code> ... 658, 2323	<code>\RenewDocumentEnvironment</code> ... 2817, 2820, 2823, 2826, 2829
P	<code>\RequirePackage</code> ... 1, 3, 21, 22, 23
peek commands:	<code>\right</code> ... 967, 1033, 1045
<code>\peek_meaning_ignore_spaces:NTF</code> ... 1058	<code>\rotate</code> ... 657
<code>\pgfcoordinate</code> ... 462, 570, 1119, 1129, 1149, 1165, 1180	S
<code>\pgfextracty</code> ... 2800	<code>\scriptstyle</code> ... 437, 1200, 1242, 1934, 1938, 1973, 1983
<code>\pgfgetlastxy</code> ... 220	

seq commands:	
\seq_clear:N	2912
\seq_count:N	1080
\seq_gclear_new:N	670, 671, 1076, 1091
\seq_gpop_left:NN	1082, 1093
\seq_gput_left:Nn	333, 2149, 2151
\seq_gset_split:Nnn	1078, 1092
\seq_if_in:NnTF	331, 2559, 2567, 2931
\seq_map_function:NN	1084, 1095
\seq_map_inline:Nn	2913
\seq_mapthread_function:NNN	2691
\seq_new:N	77
\seq_put_left:Nn	2915
\seq_set_eq:NN	2917
\seq_set_from_clist:Nn	2921
\seq_use:Nnnn	3193
\l_tmpa_seq	2912, 2915, 2917
skip commands:	
\skip_gadd:Nn	1142
\skip_gset:Nn	1133
\skip_gset_eq:NN	1140, 1141
\skip_horizontal:N	716, 852, 854, 855, 866, 867, 868, 887, 888, 961, 963, 976, 977, 1050, 1051, 1053, 1054, 1116, 1144, 1162, 1177, 1189, 1226, 1227, 1228, 1230, 1262, 1267, 1268, 1269
\skip_vertical:N	958, 965, 2417
\skip_vertical:n	2255
\g_tmpa_skip	1133, 1140, 1141, 1142, 1144, 1162
\c_zero_skip	622
\space	110, 111
str commands:	
\c_backslash_str	110
\c_colon_str	394
\str_case:nnTF	991
\str_gclear:N	1447
\str_gset:Nn	743, 1279, 1286, 1294, 1302, 1310, 1318, 1327, 2869
\str_if_empty:NTF	465, 525, 572, 742, 775, 793, 1120, 1130, 1150, 1167, 1182, 1278, 1285, 1293, 1301, 1309, 1317, 1367, 1388, 2683, 2710
\str_if_eq:nnTF	109, 323, 378, 559, 892, 908, 911, 985, 1105
\str_if_eq_p:nn	262, 2885
\str_lowercase:n	688, 703
\str_new:N	104, 105, 234, 243, 393
\str_set:Nn	106, 235, 317, 318, 319, 330, 387, 912
\str_set_eq:NN	334, 394
\l_tmpa_str	330, 331, 333, 334
T	
\tabskip	622
T _E X and L ^A T _E X 2 _ε commands:	
\@acol	554
\@acoll	552
\@acolr	553
\@adddtopreamble	757
\@array@array	556
\@arrayacol	552, 553, 554
\@arrayrule	757
\@arstrutbox	458, 459, 624, 626, 628, 631, 633, 2255
\@halignto 555	
\@ifclassloaded 48, 51	
\@ifnextchar 675	
\@ifpackageloaded 36, 89, 146	
\@mainaux 1361, 1362, 1369, 1375, 1382, 1383, 1390, 1396, 2520, 2521, 2527	
\@tempa 2161, 2163, 2263, 2265	
\@temptokena 155, 158, 172, 174	
\bBigg@ 818, 820	
\c@iRow 438, 447, 456, 463, 468, 469, 475, 483, 523, 528, 529, 542, 570, 575, 576, 582, 586, 600, 601, 672, 713, 715, 874, 877, 883, 884, 897, 994, 1009, 1112, 1202, 1206, 1243, 1247, 1472, 1488, 1773, 2150, 2179, 2250, 2331, 2332, 2339, 2340, 2355, 2363, 2424, 2488, 2634, 2725, 2727, 2753, 3035, 3036	
\c@jCol 431, 432, 434, 440, 523, 528, 529, 543, 565, 603, 604, 718, 720, 722, 881, 882, 1238, 1239, 1473, 1493, 1504, 1572, 2150, 2153, 2180, 2212, 2329, 2371, 2447, 2647, 2668, 2726, 2728, 2754, 2966	
\c@MaxMatrixCols 1341, 2959	
\CT@arc@ 92, 589, 2347, 2481	
\CT@everycr 615	
\CT@row@color 617	
\NC@find 156, 179	
\NC@find@W 677	
\NC@find@w 676	
\NC@rewrite@S 157, 170	
\new@ifnextchar 675	
\pgf@relevantforpicturesizefalse 1462, 1913, 1991, 2082, 2350, 2600, 2610, 2621, 2761	
\pgf@x 1625, 1630, 1666, 1675, 1708, 1715, 1782, 1784, 1825, 1833, 1876, 1884, 2316, 2319, 2446, 2449, 2484, 2485, 2562, 2570, 2765, 2769, 2802	
\pgf@y 896, 898, 925, 927, 1008, 1010, 1013, 1626, 1631, 1669, 1678, 1724, 1726, 1768, 1774, 1823, 1831, 1874, 1882, 2317, 2320, 2353, 2356, 2364, 2366, 2441, 2487, 2489, 2558, 2566, 2581, 2582, 2591, 2592, 2763, 2767	
\tikz@library@external@loaded 748	
tex commands:	
\tex_mkern:D 61, 63, 65, 68	
\tex_the:D 174	
\textfont 1014	
\the 182, 183	
\theiRow 599, 1472	
\thejCol 602, 1473	
\tikzpicture 39, 261, 2468	
\tikzset 750, 752, 1433	
tl commands:	
\tl_const:Nn 39, 40, 43, 44, 130, 1193, 1233	
\tl_count:n 386	
\tl_gclear:N 1430, 1445	
\tl_gclear_new:N 730, 731, 732, 733, 734, 735	
\tl_gput_left:Nn 2722	
\tl_gput_right:Nn 538, 721, 2176, 2423, 2880, 2888	
\tl_gset:Nn 158, 162, 164	
\tl_if_empty:nnTF 399, 413, 421, 1068, 1099, 1656, 1698, 1739, 1813,	

<code>\tl_if_eq:NNTF</code>	1914, 2451, 2454	
<code>\tl_if_eq:nnTF</code>	1071, 1073	
<code>\tl_item:Nn</code>	161, 162, 164	
<code>\tl_map_inline:nn</code>	1069	
<code>\tl_new:N</code>	113,	
	114, 118, 119, 120, 121, 129, 160, 163, 252, 254	
<code>\tl_put_left:Nn</code>	826, 834	
<code>\tl_put_right:Nn</code>	764, 837, 845, 847	
<code>\tl_set:Nn</code>	161, 253, 263, 405,	
	406, 407, 408, 739, 740, 824, 1328, 2095,	
	2166, 2268, 2450, 2453, 2602, 2613, 2623, 2626	
<code>\tl_set_eq:NN</code>	131, 2163, 2265, 2456, 2490	
<code>\tl_set_rescan:Nnn</code>	707, 1077, 2096, 2167, 2269	
<code>\tl_to_str:n</code>	2915	
<code>\tl_use:N</code>	3053, 3058, 3094, 3138	
<code>\g_tmpa_tl</code>	158, 161, 164	
<code>\l_tmpa_tl</code>	161, 162,	
	824, 826, 834, 837, 845, 847, 1060, 1081,	
	1082, 1083, 1093, 1094, 2450, 2451, 2453, 2454	
token commands:		
<code>\token_to_str</code>	2982	
<code>\token_to_str:N</code>		
	2945, 2987, 2993, 2994, 3008, 3015, 3048, 3059	
		U
use commands:		
<code>\use:N</code>	541, 780, 787, 798, 805	
<code>\use:n</code>	2275, 2323	
<code>\use:nn</code>	2737	
<code>\usepgfmodule</code>	2	
		V
vbox commands:		
<code>\vbox:n</code>	67	
<code>\vbox_set_top:Nn</code>	2252	
<code>\vbox_to_ht:nn</code>	198, 225, 1029, 1042	
<code>\vbox_to_zero:n</code>	2254	
<code>\vcenter</code>	956, 1027, 3048	
<code>\Vdots</code>	649, 2994	
<code>\vdots</code>	642, 664	
<code>\vfill</code>	201, 227	
<code>\vline</code>	92, 2343	
<code>\VNiceArray</code>	3235, 3270	
<code>\vNiceArray</code>	3229, 3263	
<code>\VNiceMatrix</code>	2824	
<code>\vNiceMatrix</code>	2821	
		X
<code>\xglobal</code>	443, 450, 1210, 1250	

Contents

1	Presentation	1
2	The environments of this package	2
3	The continuous dotted lines	2
3.1	The option <code>nullify-dots</code>	3
3.2	The command <code>\Hdotsfor</code>	4
3.3	How to generate the continuous dotted lines transparently	4
3.4	The labels of the dotted lines	5
3.5	Customization of the dotted lines	5
4	The PGF/Tikz nodes created by <code>nicematrix</code>	6
5	The code-after	7
6	The environment <code>{NiceArray}</code> and its variants	8
7	The vertical position of the arrays	9
8	The exterior rows and columns	10
9	The dotted lines to separate rows or columns	11
10	The width of the columns	12
11	Block matrices	13

12	Advanced features	14
12.1	Alignement option in NiceMatrix	14
12.2	The command <code>\rotate</code>	14
12.3	The option <code>small</code>	15
12.4	The counters <code>iRow</code> and <code>jCol</code>	15
12.5	The options <code>hlines</code> , <code>vlines</code> and <code>hlines</code>	16
12.6	The option <code>light-syntax</code>	17
12.7	Use of the column type <code>S</code> of <code>siunitx</code>	17
13	Technical remarks	17
13.1	Definition of new column types	17
13.2	The names of the PGF nodes created by <code>nicematrix</code>	18
13.3	Diagonal lines	18
13.4	The “empty” cells	19
13.5	The option <code>exterior-arraycolsep</code>	19
13.6	The class option <code>draft</code>	19
13.7	A technical problem with the argument of <code>\\</code>	19
13.8	Obsolete environments	20
14	Examples	20
14.1	Dotted lines	20
14.2	Dotted lines which are no longer dotted	22
14.3	Width of the columns	22
14.4	How to highlight cells of the matrix	23
14.5	Direct use of the Tikz nodes	26
15	Implementation	27
16	History	104
	Index	108