

The package `nicematrix`*

F. Pantigny
fpantigny@wanadoo.fr

May 8, 2020

Abstract

The LaTeX package `nicematrix` provides new environments similar to the classical environments `{tabular}`, `{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 lines;
- exterior rows and columns for labels;
- a control of the width of the columns.

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

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 4.0 of `nicematrix`, at the date of 2020/05/08.

2 The environments of this package

The package `nicematrix` defines the following new environments.

<code>{NiceMatrix}</code>	<code>{NiceArray}</code>	<code>{NiceTabular}</code>
<code>{pNiceMatrix}</code>	<code>{pNiceArray}</code>	
<code>{bNiceMatrix}</code>	<code>{bNiceArray}</code>	
<code>{BNiceMatrix}</code>	<code>{BNiceArray}</code>	
<code>{vNiceMatrix}</code>	<code>{vNiceArray}</code>	
<code>{VNiceMatrix}</code>	<code>{VNiceArray}</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 environments `{NiceArray}` and `{NiceTabular}` are similar to the environments `{array}` and `{tabular}` of the package `{array}`. However, for technical reasons, in the preamble of these environments, 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{...}{...}`, `l`, `>{...}`, `<{...}`, `@{...}`, `!{...}` and `*{n}{...}` but the letters `p`, `m` and `b` should not be used.

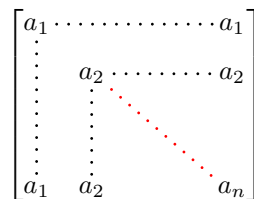
See p. 8 the section relating to `{NiceArray}` and `{NiceTabular}`.

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`.¹

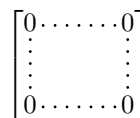
Each of them must be used alone in the cell of the array and it draws a dotted line between the first non-empty cells² on both sides of the current cell. Of course, for `\Ldots` and `\Cdots`, it's an horizontal line; for `\Vdots`, it's a vertical line and for `\Ddots` and `\Iddots` diagonal ones. It's possible to change the color of these lines with the option `color`.³

```
\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}
```



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}
```



¹The 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`.

²The precise definition of a “non-empty cell” is given below (cf. p. 20).

³It'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 PDF⁴).

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.⁵

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.

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

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

⁴ And it's not possible to draw a `\Ldots` and a `\Cdots` line between the same cells.

⁵ In `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`.⁶

- The option `renew-dots`

With this option, the commands `\ldots`, `\cdots`, `\vdots`, `\ddots`, `\iddots`¹ 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.

⁶The 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. 9.

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).⁷

⁷The 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`:

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

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

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 loaded⁸). First, the user have to give a name to the array (with the key called `name`). Then, the nodes are accessible through the names “`name-i-j`” where `name` is the name given to the array and `i` and `j` the numbers of the row and the column of the considered cell.

```
$\begin{pNiceMatrix}[name=mymatrix]
```

```
1 & 2 & 3 \\
```

```
4 & 5 & 6 \\
```

```
7 & 8 & 9
```

```
\end{pNiceMatrix}$
```

```
\tikz[remember picture,overlay]
```

```
\draw (mymatrix-2-2) circle (2mm) ;
```

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

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

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

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

In fact, the package `nicematrix` can create “extra nodes”: the “medium nodes” and the “large nodes”. The first ones are created with the option `create-medium-nodes` and the second ones with the option `create-large-nodes`.⁹

⁸We remind that, since the version 3.13, `nicematrix` doesn't load Tikz by default by only PGF (Tikz is a layer over PGF).

⁹There 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.¹⁰

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

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`.¹¹

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

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. 24).

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 loaded¹², 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.

¹⁰There is no “large nodes” created in the exterior rows and columns (for these rows and columns, cf. p. 9).

¹¹The 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.

¹²We 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}

```

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

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`. For an example, cf. p. 25.

6 The environments `{NiceArray}` and `{NiceTabular}`

The environments `{NiceArray}` and `{NiceTabular}` are similar to the environments `{array}` and `{tabular}`. 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 R¹³ instead of l, c and r. In a command `\multicolumn`, one should also use the letters L, C, R.

It's possible to use the constructions `p{...}`, `m{...}`, `b{...}`, `w{...}{...}`, `W{...}{...}`, `l{...}`, `>{...}`, `<{...}`, `@{...}`, `!{...}` and `*{n}{...}`. However, in the environment `{NiceArray}` (and its variants), the content of the columns of type `w` or `W` is composed in math mode (in `{array}` of `array`, they are composed in text mode).

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:

```

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

```

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

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.

```

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

```

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

¹³The column types L, C and R are defined locally inside `{NiceArray}` with `\newcolumnstype` of `array`. This definition overrides an eventual previous definition.

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`¹⁴).

```
\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
2.

n	0	1	2	3	4	5
u_n	1	2	4	8	16	32

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
2.

n	0	1	2	3	4	5
u_n	1	2	4	8	16	32

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{pNiceMatrix}[first-row,last-row,first-col,last-col]
$\begin{pNiceMatrix}[first-row,last-row,first-col,last-col,nullify-dots]
& C_1 & & \Cdots & & C_4 & & \\
L_1 & a_{11} & a_{12} & a_{13} & a_{14} & L_1 & & \end{pNiceMatrix}
```

¹⁴It's also possible to use `\firsthline` with `{NiceArray}`.

```

\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{pNiceMatrix}$
\end{pNiceMatrix}$

```

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

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 `{NiceTabular}` and `{pNiceArray}`, the number of columns can obviously be computed from the preamble.
 - When the option `light-syntax` (cf. p. 18) 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}}
$\begin{pNiceArray}{CC|CC}[first-row,last-row=5,first-col,last-col,nullify-dots]
    & C_1    & & \Cdots & & C_4    & & \\\
L_1  & a_{11} & & a_{12} & & a_{13} & & a_{14} & & L_1 & \\\
\Vdots & a_{21} & & a_{22} & & a_{23} & & a_{24} & & \Vdots & \\\
\hline
    & a_{31} & & a_{32} & & a_{33} & & a_{34} & & \\\
L_4  & a_{41} & & a_{42} & & a_{43} & & a_{44} & & L_4 & \\\
    & C_1    & & \Cdots & & C_4    & & & & &
\end{pNiceArray}$

```

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

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.¹⁵

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 19.

- 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)

```

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

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

¹⁵The 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.

```

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

```

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`.

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 rule¹⁶. 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.

```

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

```

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}

```

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

It's possible to give the special value `auto` to the option `columns-width`: all the columns of the array will have a width equal to the widest cell of the array.¹⁷

```

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

```

¹⁶In fact, this is true only for `\hline` and “|” but not for `\cline`.

¹⁷The 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).

$$\backslash \text{NiceMatrixOptions}\{\text{columns-width}=10\text{mm}\}$$
$$\end{pNiceMatrix}$$
$$=$$

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

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$
$$1 \quad \& \quad 1245 \quad \backslash \backslash \quad 345 \quad \& \quad 2$$
$$\end{pNiceMatrix}$$

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`¹⁸. The environment `{NiceMatrixBlock}` has no direct link with the command `\Block` presented just below (cf. p. 13).

[illegible] $\$\\begin{pNiceMatrix}$

```
a & b \\ c & d
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
$$=$$

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

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$
$$1 \quad \& \quad 1245 \quad \backslash \backslash \quad 345 \quad \& \quad 2$$
$$\end{pNiceMatrix}$$
$$\end{NiceMatrixBlock}$$

Several compilations may be necessary to achieve the job.

11 Block matrices

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

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

$$\arrayrulecolor{blue}$$
$$\begin{array}{ccc|c} & & & \\ & & & \\ & & & \end{array}$$
$$\text{Block}\{3-3\}\{A\} \quad \& \quad \& \quad \& \quad 0 \quad \backslash \backslash$$
$$\& \hspace*{1cm} \& \& \text{\Vdots} \&$$
$$\& \& \& 0 \quad \backslash \backslash$$

\hline

$$0 \quad \& \quad \backslash \text{Cdots} \quad \& \quad 0 \quad \& \quad 0$$
$$\end{bNiceArray}$$
$$\arrayrulecolor{black}$$

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

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`

¹⁸At this time, this is the only usage of the environment `{NiceMatrixBlock}` but it may have other usages in the future.

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

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

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

12 The color of the rows and columns

With the classical package `colortbl`, it's possible to color the cells, rows and columns of a tabular. However, the resulting PDF is not always perfectly displayed by the PDF viewers, in particular in conjunction with rules. With some PDF viewers, some vertical rules seem to vanish. On the other side, some thin horizontal white lines may appear in some circumstances.

The version 4.0 of `nicematrix` provide similar tools which do not present these drawbacks. This version provides a key `code-before` for some code which will be executed *before* the drawing of the tabular. In this `code-before`, new commands are available: `\cellcolor`, `\rectanglecolor`, `\rowcolor`, `\columncolor`, `\rowcolors` and `\chessboardcolors`.

- The command `\cellcolor` takes its name from the command `\cellcolor` of `colortbl`.

This command takes in arguments a color and a list of cells, each of which with the format $i-j$ where i is the number of row and j the number of column of the cell.

```
\begin{NiceTabular}{|C|C|C|}[code-before = \cellcolor{red!15}{3-1,2-2,1-3}]
\hline
a & b & c \\ \hline
e & f & g \\ \hline
h & i & j \\ \hline
\end{NiceTabular}
```

a	b	c
e	f	g
h	i	j

- The command `\rectanglecolor` takes three arguments. The first is the color. The second is the upper-left cell of the rectangle and the third is the lower-right cell of the rectangle.

```
\begin{NiceTabular}{|C|C|C|}[code-before = \rectanglecolor{red!15}{2-2}{3-3}]
\hline
a & b & c \\ \hline
e & f & g \\ \hline
h & i & j \\ \hline
\end{NiceTabular}
```

a	b	c
e	f	g
h	i	j

- The command `\rowcolor` takes its name from the command `\rowcolor` of `colortbl`. Its first argument is the color and the second is a comma-separated list of rows or interval of rows with the form $a-b$ (an interval of the form $a-$ represent all the rows from the row a until the end).

```

 $\begin{NiceArray}{LLL}[hvlines, code-before = \rowcolor{red!15}{1,3-5,8-}]
a_1 & b_1 & c_1 \\
a_2 & b_2 & c_2 \\
a_3 & b_3 & c_3 \\
a_4 & b_4 & c_4 \\
a_5 & b_5 & c_5 \\
a_6 & b_6 & c_6 \\
a_7 & b_7 & c_7 \\
a_8 & b_8 & c_8 \\
a_9 & b_9 & c_9 \\
a_{10} & b_{10} & c_{10} \\
\end{NiceArray}$ 

```

a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3
a_4	b_4	c_4
a_5	b_5	c_5
a_6	b_6	c_6
a_7	b_7	c_7
a_8	b_8	c_8
a_9	b_9	c_9
a_{10}	b_{10}	c_{10}

- The command `\columncolor` takes its name from the command `\columncolor` of `colortbl`. Its syntax is similar to the syntax of `\rowcolor`.
- The command `\rowcolors` (with a *s*) takes its name from the command `\rowcolors` of `xcolor`. The *s* emphasizes the fact that there is *two* colors. This command colors alternately the rows of the tabular, beginning with the row whose number is given in first argument. The two other arguments are the colors.

```

\begin{NiceTabular}{LR}%
[code-before = \rowcolor{red!15}{1} \rowcolors{3}{blue!10}{}]
\toprule
Town & Habitants \\
\midrule
Pau & 80000 \\
Paris & 2000000 \\
Toulouse & 700000 \\
Reims & 40000 \\
\bottomrule
\end{NiceTabular}

```

Ville	habitants
Pau	80000
Paris	2000000
Toulouse	700000
Reims	40000

- The command `\chessboardcolors` take in arguments two colors and colors the cells of the tabular in quincunx with these colors.

```

 $\begin{pNiceMatrix}[R,margin, code-before=\chessboardcolors{red!15}{blue!15}]
1 & -1 & 1 \\
-1 & 1 & -1 \\
1 & -1 & 1 \\
\end{pNiceMatrix}$ 

```

1	-1	1
-1	1	-1
1	-1	1

13 Advanced features

13.1 Aligement 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).¹⁹

```

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

```

$\cos x$	$-\sin x$
$\sin x$	$\cos x$

¹⁹This is a part of the functionality provided by the environments `{pmatrix*}`, `{bmatrix*}`, etc. of `mathtools`.

13.2 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 }
$A = \begin{pNiceMatrix}[first-row,last-col=4]
e_1 & e_2 & e_3 & \\
1 & 2 & 3 & e_1 \\
4 & 5 & 6 & e_2 \\
7 & 8 & 9 & e_3 \\
\end{pNiceMatrix}$
```

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

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

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

13.3 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`).

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

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

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.

13.4 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 column²⁰. 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).

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

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

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 `\AutoNiceMatrix`, `\pAutoNiceMatrix`, `\bAutoNiceMatrix`, `\vAutoNiceMatrix`, `\VAutoNiceMatrix` and `\BAutoNiceMatrix`.

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

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

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

13.5 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>

²⁰We 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.

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`.

```

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

```

a	b	c	d
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}
 $\begin{pNiceMatrix}[vlines]
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 2 & 3 & 4 & 5 & 6 \\
\end{pNiceMatrix}$ 

```

$$\left(\begin{array}{c|c|c|c|c|c} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{array} \right)$$

13.6 The option `light-syntax`

The option `light-syntax`²¹ 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.²²

13.7 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`.

```

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

```

$$\left(\begin{array}{cccccc} C_1 & \dots & \dots & \dots & \dots & C_n \\ 2.3 & 0 & \dots & \dots & \dots & 0 \\ 12.4 & \vdots & & & & \vdots \\ 1.45 & \vdots & & & & \vdots \\ 7.2 & 0 & \dots & \dots & \dots & 0 \end{array} \right)$$

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

²¹This option is inspired by the package `spalign` of Joseph Rabinoff.

²²The 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.

14 Technical remarks

14.1 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 job²³:

```
\newcolumnstype{?}{!\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}
```

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

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

14.2 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`.

14.3 Diagonal lines

By default, all the diagonal lines²⁴ 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 &      & \Vdots \\
a+b    & \Cdots & a+b  & 1      \\
\end{pNiceMatrix}
```

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

²³The command `\vrule` is a TeX (and not LaTeX) command.

²⁴We speak of the lines created by `\Ddots` and not the lines created by a command `\line` in `code-after`.

```

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

```

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

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

The same example without parallelization:

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

14.4 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`.

14.5 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 idea²⁵. The environment `{matrix}` of `amsmath` and its variants (`{pmatrix}`, `{vmatrix}`, etc.) of `amsmath` prefer to delete these spaces with explicit instructions `\hspace -\arraycolsep`²⁶. 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).

²⁵In 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).*

²⁶And 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

14.6 A technical problem with the argument of `\`

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

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

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

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

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

<pre>\begin{pNiceMatrix} a & \multicolumn1C{\frac AB} \\\[2mm] b & c \end{pNiceMatrix}</pre>	$\begin{pmatrix} a & \frac{A}{B} \\ b & c \end{pmatrix}$
--	--

14.7 Obsolete environments which have been deleted

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 have been deleted from `nicematrix`.

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

However, the definition of those environments is still available (temporarily) at the end of the file `nicematrix.sty` after a command `\file_input_stop:`.

15 Examples

15.1 Dotted lines

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

<pre>\$\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}\$</pre>	$\begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ & & & \ddots & 0 \\ & & & \ddots & 1 \\ 0 & 0 & & \cdots & 0 \\ 1 & 0 & & \cdots & 0 \end{pmatrix}$
---	--

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

```

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

An example with `\Hdotsfor`:

```

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

```

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

An example for the resultant of two polynomials:

```

 $\setlength{\extrarowheight}{1mm}$ 
 $\begin{vNiceArray}{CCC:CCC}[columns-width=6mm]$ 
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}

```

$$\left| \begin{array}{ccc} a_0 & & \\ & \ddots & \\ a_1 & & a_0 \\ & \ddots & \\ a_p & & a_1 \\ & \ddots & \\ & & a_p \end{array} \right| \quad \left| \begin{array}{ccc} b_0 & & \\ & \ddots & \\ b_1 & & b_0 \\ & \ddots & \\ b_q & & b_1 \\ & \ddots & \\ & & b_q \end{array} \right|$$

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}
```

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

15.2 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 not longer dotted.

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

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

15.3 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
{ last-col,code-for-last-col = \color{blue}\scriptstyle,light-syntax}
\setlength{\extrarowheight}{1mm}
$\begin{pNiceArray}{CCCC:C}
  1  1  1  1  1 ;
  2  4  8 16  9 ;
  3  9 27 81 36 ;
  4 16 64 256 100
\end{pNiceArray}$
\medskip
$\begin{pNiceArray}{CCCC:C}
  1  1  1  1  1 ;
  0  2  6 14  7      { L_2 \gets -2 L_1 + L_2 } ;
  0  6 24 78 33      { L_3 \gets -3 L_1 + L_3 } ;
  0 12 60 252 96      { L_4 \gets -4 L_1 + L_4 }
\end{pNiceArray}$
...
\end{NiceMatrixBlock}
```

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

15.4 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 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`).

```

 $\begin{pNiceArray}{>{\strut}CCCC}[create-large-nodes,margin,extra-margin = 2pt]
  a_{11} & a_{12} & a_{13} & a_{14} \\
  a_{21} & a_{22} & a_{23} & a_{24} \\
  a_{31} & a_{32} & a_{33} & a_{34} \\
  a_{41} & a_{42} & a_{43} & a_{44}
\end{pNiceArray}$ 
 $\begin{tikzpicture}[name suffix = -large,
  every node/.style = {draw,inner sep = 0 pt}]
  \node [fit = (1-1)] {} ;
  \node [fit = (2-2)] {} ;
  \node [fit = (3-3)] {} ;
  \node [fit = (4-4)] {} ;
\end{tikzpicture}$ 
 $\end{pNiceArray}$ 

```

$$\left(\begin{array}{|c|c|c|c|} \hline a_{11} & a_{12} & a_{13} & a_{14} \\ \hline a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ \hline a_{41} & a_{42} & a_{43} & a_{44} \\ \hline \end{array} \right)$$

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).²⁷

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}}

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

```

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

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

²⁷On the other side, the command `\cline` doesn't spread the rows of the array.

```

\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

```

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`.

```

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

```

$$\begin{pmatrix} \text{A} & 0 \\ \vdots & \vdots \\ 0 & 0 \end{pmatrix}$$

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}
```

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

15.5 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}
& &
```

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

```

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

```

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

```

\begin{bNiceArray}{CC>{\strut}CCC}[name=A,first-col]
    & a_{11} & \Cdots & & & a_{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}

```

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

16 Implementation

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

However, when the option `renew-dots` is used, the commands `\cdots`, `\ldots`, `\dots`, `\vdots`, `\ddots` and `\iddots` are redefined in the environments provided by `nicematrix` as explained previously.

In the same way, if the option `renew-matrix` is used, the environment `{matrix}` of `amsmath` is redefined.

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

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

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

We load some packages.

```
17 \RequirePackage { array }
18 \RequirePackage { amsmath }
19 \RequirePackage { xparse }

20 \cs_new_protected:Npn @@_error:n { \msg_error:nn { nicematrix } }
21 \cs_new_protected:Npn @@_error:nn { \msg_error:nnn { nicematrix } }
22 \cs_new_protected:Npn @@_error:nnn { \msg_error:nnnn { nicematrix } }
23 \cs_new_protected:Npn @@_fatal:n { \msg_fatal:nn { nicematrix } }
24 \cs_new_protected:Npn @@_fatal:nn { \msg_fatal:nnn { nicematrix } }
25 \cs_new_protected:Npn @@_msg_new:nn { \msg_new:nnn { nicematrix } }
26 \cs_new_protected:Npn @@_msg_new:nnn { \msg_new:nnnn { nicematrix } }

27 \cs_new_protected:Npn @@_msg_redirect_name:nn
28 { \msg_redirect_name:nnn { nicematrix } }
```

Technical definitions

```

29 \bool_new:N \c_@@_booktabs_loaded_bool
30 \bool_new:N \c_@@_tikz_loaded_bool
31 \AtBeginDocument
32 {
33   \ifpackageloaded { booktabs }
34     { \bool_set_true:N \c_@@_booktabs_loaded_bool }
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 { }

```

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.

```

54 \ProvideDocumentCommand \iddots { }
55 {
56   \mathinner
57   {
58     \tex_mkern:D 1 mu
59     \box_move_up:nn { 1 pt } { \hbox:n { . } }
60     \tex_mkern:D 2 mu
61     \box_move_up:nn { 4 pt } { \hbox:n { . } }
62     \tex_mkern:D 2 mu
63     \box_move_up:nn { 7 pt }
64     { \vbox:n { \kern 7 pt \hbox:n { . } } }
65     \tex_mkern:D 1 mu
66   }
67 }

```

This definition is a variant of the standard definition of `\ddots`.

In the `aux` file, we will have the references of the PGF/Tikz nodes created by `nicematrix`. However, when `booktabs` is used, some nodes (more precisely, some `row` nodes) will be defined twice because their position will be modified. In order to avoid an error message in this case, we will redefine `\pgfutil@check@rerun` in the `aux` file.

```

68 \AtBeginDocument
69 {
70   \@ifpackageloaded { booktabs }
71     { \iow_now:Nn \mainaux \nicematrix@redefine@check@rerun }
72     { }
73   }
74   \cs_set_protected:Npn \nicematrix@redefine@check@rerun
75     {
76       \cs_set_eq:NN \@@_old_pgful@check@rerun \pgful@check@rerun

```

The new version of `\pgful@check@rerun` will not check the PGF nodes whose names start with `nm-` (which is the prefix for the nodes creates by `nicematrix`).

```

77   \cs_set_protected:Npn \pgful@check@rerun ##1 ##2
78     {
79       \str_set:Nx \l_tmpa_str { \tl_range:nnn { ##1 } 1 3 }
80       \str_if_eq:VnF \l_tmpa_str { nm- }
81         { \@@_old_pgful@check@rerun { ##1 } { ##2 } }
82     }
83   }

```

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

```

84 \bool_new:N \c_@@_colortbl_loaded_bool
85 \AtBeginDocument
86 {
87   \@ifpackageloaded { colortbl }
88     { \bool_set_true:N \c_@@_colortbl_loaded_bool }
89     {

```

The command `\CT@arc@` is a command of `colortbl` which sets the color of the rules in the array. We will use it to store the instruction of color for the rules even if `colortbl` is not loaded.

```

90   \cs_set_protected:Npn \CT@arc@ { }
91   \NewDocumentCommand { \arrayrulecolor } { m }
92     { \tl_gset:Nn \CT@arc@ { \color { #1 } } }

```

The following line are from the redefinition of `\hline` of standard LaTeX by `colortbl` (array does not redefine `\hline`).

```

93   \cs_set:Npn \hline
94     {
95       \noalign { \ifnum 0 = ` } \fi
96       \let \hskip \vskip
97       \let \vrule \hrule
98       \let \@width \@height
99       { \CT@arc@ \vline }
100       \futurelet \reserved@a
101       \@xhline
102     }
103   }
104 }

```

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

```

105 \cs_new:Npn \@@_succ:n #1 { \the \numexpr #1 + 1 \relax }
106 \cs_new:Npn \@@_pred:n #1 { \the \numexpr #1 - 1 \relax }

```

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`.

```

107 \bool_new:N \c_@@_siunitx_loaded_bool
108 \AtBeginDocument
109 {
110   \@ifpackageloaded { siunitx }
111     { \bool_set_true:N \c_@@_siunitx_loaded_bool }
112     { }

```

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:`.

```
114 \cs_set_protected:Npn \@@_adapt_S_column:
115 {
116   \bool_if:NT \c_@@_siunitx_loaded_bool
117   {
118     \group_begin:
119     \@temptokena = { }
```

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

```
120   \cs_set_eq:NN \NC@find \prg_do_nothing:
121   \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`).

```
122   \tl_gset:NV \g_tmpa_tl \@temptokena
123   \group_end:
124   \tl_new:N \c_@@_table_collect_begin_tl
125   \tl_set:Nx \l_tmpa_tl { \tl_item:Nn \g_tmpa_tl 2 }
126   \tl_gset:Nx \c_@@_table_collect_begin_tl { \tl_item:Nn \l_tmpa_tl 1 }
127   \tl_new:N \c_@@_table_print_tl
128   \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).

```
129   \cs_gset_eq:NN \@@_adapt_S_column: \prg_do_nothing:
```



```

130     }
131 }

```

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).

```

132 \cs_new_protected:Npn \@@_renew_NC@rewrite@S:
133 {
134   \renewcommand*{\NC@rewrite@S}[1] []
135   {
136     \@temptokena \exp_after:wN
137     {
138       \tex_the:D \@temptokena
139       > { \@@_Cell: \c_@@_table_collect_begin_tl S {##1} }
140       c
141       < { \c_@@_table_print_tl \@@_end_Cell: }
142     }
143     \NC@find
144   }
145 }

```

Parameters

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.

```

146 \int_new:N \g_@@_env_int

```

The following command is only a syntactic shortcut. It must *not* be protected (it will be used in names of PGF nodes).

```

147 \cs_new:Npn \@@_env: { nm - \int_use:N \g_@@_env_int }

```

The following command is only a syntactic shortcut. The `q` in `qpoint` means *quick*.

```

148 \cs_new_protected:Npn \@@_qpoint:n #1
149 { \pgfpointanchor { \@@_env: - #1 } { center } }

```

the following counter will count the environments `{NiceMatrixBlock}`.

```

150 \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).

```

151 \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`.

```

152 \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.

```

153 \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).

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

If the user uses `{NiceTabular}`, we will raise the following flag.

```
155 \bool_new:N \l_@@_NiceTabular_bool
```

```
156 \cs_new_protected:Npn \@@_test_if_math_mode:
157 {
158   \if_mode_math: \else:
159     \@@_fatal:n { Outside-math-mode }
160   \fi:
161 }
```

The following colors will be used to memorize the color of the potential “first col” and the potential “first row”.

```
162 \colorlet { nicematrix-last-col } { . }
163 \colorlet { nicematrix-last-row } { . }
```

The following string is the name of the current environment or the current command of `nicematrix` (despite its name which contains `env`).

```
164 \str_new:N \g_@@_name_env_str
```

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

```
165 \str_new:N \g_@@_com_or_env_str
166 \str_set:Nn \g_@@_com_or_env_str { environment }
```

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

```
167 \cs_new:Npn \@@_full_name_env:
168 {
169   \str_if_eq:VnTF \g_@@_com_or_env_str { command }
170     { command \space \c_backslash_str \g_@@_name_env_str }
171     { environment \space \{ \g_@@_name_env_str \} }
172 }
```

The following token list corresponds to the option `code-after` (it's also possible to set the value of that parameter with the command `\CodeAfter`).

```
173 \tl_new:N \g_@@_code_after_tl
```

The following token list has a function similar to `\g_@@_code_after_tl` but it is used internally by `nicematrix`. In fact, we have to distinguish between `\g_@@_code_after_tl` and `\g_@@_internal_code_after_tl` because we must take care of the order in which instructions stored in that parameters are executed.

```
174 \tl_new:N \g_@@_internal_code_after_tl
```

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

```
175 \int_new:N \l_@@_old_iRow_int
176 \int_new:N \l_@@_old_jCol_int
```

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

A kind of false row will be inserted at the end of the array for the construction of the `col` nodes (and also to fix the width of the columns when `columns-width` is used). When this special row will be created, we will raise the flag `\g_@@_row_of_col_done_bool` in order to avoid some actions set in the redefinition of `\everycr` when the last `\cr` of the `\halign` will occur (after that row of `col` nodes).

```
177 \bool_new:N \g_@@_row_of_col_done_bool
```

The following flag will be raised when the key `code-before` is used in the environment. Indeed, if there is a `code-before` in the environment, we will manage to have the `row` nodes and the `col` nodes available *before* the creation of the array.

```
178 \bool_new:N \l_@@_code_before_bool
```

The following dimensions will be used when drawing the dotted lines.

```
179 \dim_new:N \l_@@_x_initial_dim
180 \dim_new:N \l_@@_y_initial_dim
181 \dim_new:N \l_@@_x_final_dim
182 \dim_new:N \l_@@_y_final_dim
```

`expl3` provides scratch dimension `\l_tmpa_dim` and `\l_tmpd_dim`. We create two other in the same spirit (if they don't exist yet : that's why we use `\dim_zero_new:N`).

```
183 \dim_zero_new:N \l_tmpc_dim
184 \dim_zero_new:N \l_tmpd_dim
```

Some cells will be declared as “empty” (for example a cell with the instruction `\Cdot`).

```
185 \bool_new:N \g_@@_empty_cell_bool
```

The following dimension will be used to save the current value of `\arraycolsep`.

```
186 \dim_new:N \@@_old_arraycolsep_dim
```

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

```
187 \dim_new:N \g_@@_width_last_col_dim
188 \dim_new:N \g_@@_width_first_col_dim
```

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.

```
189 \int_new:N \l_@@_first_row_int
190 \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.

```
191 \int_new:N \l_@@_first_col_int
192 \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).

```
193 \int_new:N \l_@@_last_row_int
194 \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”.²⁸

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

Idem for `\l_@@_last_col_without_value_bool`

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

- **Last column**

For the potential “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}`).

```
197 \int_new:N \l_@@_last_col_int
198 \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}
```

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.

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

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

Command for creation of rectangle nodes

The following command should be used in a `{pgfpicture}`. It creates a rectangle (empty but with a name).

#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.

```
200 \cs_new_protected:Npn \@@_pgf_rect_node:nnnnn #1 #2 #3 #4 #5
201 {
202   \begin { pgfscope }
203   \pgfset
204   {
205     outer-sep = \c_zero_dim ,
```

²⁸We 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.

```

206         inner~sep = \c_zero_dim ,
207         minimum~size = \c_zero_dim
208     }
209     \pgftransformshift { \pgfpoint { 0.5 * ( #2 + #4 ) } { 0.5 * ( #3 + #5 ) } }
210     \pgfnode
211     { rectangle }
212     { center }
213     {
214         \vbox_to_ht:nn
215         { \dim_abs:n { #5 - #3 } }
216         {
217             \vfill
218             \hbox_to_wd:nn { \dim_abs:n { #4 - #2 } } { }
219         }
220     }
221     { #1 }
222     { }
223     \end { pgfscope }
224 }

```

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.

```

225 \cs_new_protected:Npn \@@pgf_rect_node:nnn #1 #2 #3
226 {
227     \begin { pgfscope }
228     \pgfset
229     {
230         outer~sep = \c_zero_dim ,
231         inner~sep = \c_zero_dim ,
232         minimum~size = \c_zero_dim
233     }
234     \pgftransformshift { \pgfpointscale { 0.5 } { \pgfpointadd { #2 } { #3 } } }
235     \pgfpointdiff { #3 } { #2 }
236     \pgfgetlastxy \l_tmpa_dim \l_tmpb_dim
237     \pgfnode
238     { rectangle }
239     { center }
240     {
241         \vbox_to_ht:nn
242         { \dim_abs:n \l_tmpb_dim }
243         { \vfill \hbox_to_wd:nn { \dim_abs:n \l_tmpa_dim } { } }
244     }
245     { #1 }
246     { }
247     \end { pgfscope }
248 }

```

The options

The following dimensions correspond to the options `cell-space-top-limit` and `co` (these parameters are inspired by the package `cellspace`).

```

249 \dim_new:N \l_@@_cell_space_top_limit_dim
250 \dim_new:N \l_@@_cell_space_bottom_limit_dim

```

The following dimension is the distance between two dots for the dotted lines (when `line-style` is equal to `standard`, which is the initial value). The initial value is 0.45 em but it will be changed if the option `small` is used.

```

251 \dim_new:N \l_@@_inter_dots_dim
252 \dim_set:Nn \l_@@_inter_dots_dim { 0.45 em }

```

The following dimension 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).

```
253 \dim_new:N \l_@@_xdots_shorten_dim
254 \dim_set:Nn \l_@@_xdots_shorten_dim { 0.3 em }
```

The following dimension is the radius of the dots for the 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.

```
255 \dim_new:N \l_@@_radius_dim
256 \dim_set:Nn \l_@@_radius_dim { 0.53 pt }
```

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.

```
257 \tl_new:N \l_@@_xdots_line_style_tl
258 \tl_const:Nn \c_@@_standard_tl { standard }
259 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
```

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

```
260 \bool_new:N \l_@@_light_syntax_bool
```

The string `\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).

```
261 \str_new:N \l_@@_baseline_str
262 \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`).

```
263 \bool_new:N \l_@@_exterior_arraycolsep_bool
```

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

```
264 \bool_new:N \l_@@_parallelize_diags_bool
265 \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`.

```
266 \bool_new:N \l_@@_hlines_bool
267 \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.).

```
268 \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).

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

The string `\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.

```
270 \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”.

```
271 \bool_new:N \l_@@_medium_nodes_bool
272 \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`).

```
273 \dim_new:N \l_@@_left_margin_dim
274 \dim_new:N \l_@@_right_margin_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`.

```
275 \dim_new:N \l_@@_extra_left_margin_dim
276 \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.

```
277 \tl_new:N \l_@@_end_of_row_tl
278 \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 “:”.

```
279 \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 fonction 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.

```
280 \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.

```
281 \keys_define:nn { NiceMatrix / xdots }
282 {
283   line-style .code:n =
284   {
285     \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.

```
286     { \cs_if_exist_p:N \tikzpicture }
287     { \str_if_eq_p:nn { #1 } { standard } }
288     { \tl_set:Nn \l_@@_xdots_line_style_tl { #1 } }
289     { \@@_error:n { bad~option~for~line~style } }
290   } ,
291   line-style .value_required:n = true ,
292   color .tl_set:N = \l_@@_xdots_color_tl ,
293   color .value_required:n = true ,
294   shorten .dim_set:N = \l_@@_xdots_shorten_dim ,
295   shorten .value_required:n = true ,
```

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

```

296   down .tl_set:N = \l_@@_xdots_down_tl ,
297   up .tl_set:N = \l_@@_xdots_up_tl ,
298   unknown .code:n = \@@_error:n { Unknown-option-for-~xdots }
299 }

300 \keys_define:nn { NiceMatrix / Global }
301 {
302   cell-space-top-limit .dim_set:N = \l_@@_cell_space_top_limit_dim ,
303   cell-space-top-limit .value_required:n = true ,
304   cell-space-bottom-limit .dim_set:N = \l_@@_cell_space_bottom_limit_dim ,
305   cell-space-bottom-limit .value_required:n = true ,
306   xdots .code:n = \keys_set:nn { NiceMatrix / xdots } { #1 } ,
307   max-delimiter-width .bool_set:N = \l_@@_max_delimiter_width_bool ,
308   light-syntax .bool_set:N = \l_@@_light_syntax_bool ,
309   light-syntax .default:n = true ,
310   end-of-row .tl_set:N = \l_@@_end_of_row_tl ,
311   end-of-row .value_required:n = true ,
312   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
313   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
314   last-row .int_set:N = \l_@@_last_row_int ,
315   last-row .default:n = -1 ,
316   code-for-first-col .tl_set:N = \l_@@_code_for_first_col_tl ,
317   code-for-first-col .value_required:n = true ,
318   code-for-last-col .tl_set:N = \l_@@_code_for_last_col_tl ,
319   code-for-last-col .value_required:n = true ,
320   code-for-first-row .tl_set:N = \l_@@_code_for_first_row_tl ,
321   code-for-first-row .value_required:n = true ,
322   code-for-last-row .tl_set:N = \l_@@_code_for_last_row_tl ,
323   code-for-last-row .value_required:n = true ,
324   hlines .bool_set:N = \l_@@_hlines_bool ,
325   vlines .bool_set:N = \l_@@_vlines_bool ,
326   hvlines .meta:n = { hlines , vlines } ,
327   parallelize-diags .bool_set:N = \l_@@_parallelize_diags_bool ,

```

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

```

328   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
329   renew-dots .value_forbidden:n = true ,
330   nullify-dots .bool_set:N = \l_@@_nullify_dots_bool ,
331   create-medium-nodes .bool_set:N = \l_@@_medium_nodes_bool ,
332   create-large-nodes .bool_set:N = \l_@@_large_nodes_bool ,
333   create-extra-nodes .meta:n =
334     { create-medium-nodes , create-large-nodes } ,
335   left-margin .dim_set:N = \l_@@_left_margin_dim ,
336   left-margin .default:n = \arraycolsep ,
337   right-margin .dim_set:N = \l_@@_right_margin_dim ,
338   right-margin .default:n = \arraycolsep ,
339   margin .meta:n = { left-margin = #1 , right-margin = #1 } ,
340   margin .default:n = \arraycolsep ,
341   extra-left-margin .dim_set:N = \l_@@_extra_left_margin_dim ,
342   extra-right-margin .dim_set:N = \l_@@_extra_right_margin_dim ,
343   extra-margin .meta:n =
344     { extra-left-margin = #1 , extra-right-margin = #1 } ,
345   extra-margin .value_required:n = true
346 }

```

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

```

347 \keys_define:nn { NiceMatrix / Env }
348 {

```



```

349 code-before .code:n =
350 {
351   \tl_if_empty:nF { #1 }
352   {
353     \tl_set:Nn \l_@@_code_before_tl { #1 }
354     \bool_set_true:N \l_@@_code_before_bool
355   }
356 } ,

```

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

```

357 c .code:n = \str_set:Nn \l_@@_baseline_str c ,
358 t .code:n = \str_set:Nn \l_@@_baseline_str t ,
359 b .code:n = \str_set:Nn \l_@@_baseline_str b ,
360 baseline .tl_set:N = \l_@@_baseline_str ,
361 baseline .value_required:n = true ,
362 columns-width .code:n =
363   \str_if_eq:nnTF { #1 } { auto }
364   { \bool_set_true:N \l_@@_auto_columns_width_bool }
365   { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,
366 columns-width .value_required:n = true ,
367 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.

```

368 \legacy_if:nF { measuring@ }
369 {
370   \str_set:Nn \l_tmpa_str { #1 }
371   \seq_if_in:NVTf \g_@@_names_seq \l_tmpa_str
372   { \@@_error:nn { Duplicate-name } { #1 } }
373   { \seq_gput_left:Nv \g_@@_names_seq \l_tmpa_str }
374   \str_set_eq:NN \l_@@_name_str \l_tmpa_str
375 } ,
376 name .value_required:n = true ,
377 code-after .tl_gset:N = \g_@@_code_after_tl ,
378 code-after .value_required:n = true ,
379 }

```

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

```

380 \keys_define:nn { NiceMatrix }
381 {
382   NiceMatrixOptions .inherit:n =
383   {
384     NiceMatrix / Global ,
385   } ,
386   NiceMatrixOptions / xdots .inherit:n = NiceMatrix / xdots ,
387   NiceMatrix .inherit:n =
388   {
389     NiceMatrix / Global ,
390     NiceMatrix / Env ,
391   } ,
392   NiceMatrix / xdots .inherit:n = NiceMatrix / xdots ,
393   NiceTabular .inherit:n =
394   {
395     NiceMatrix / Global ,
396     NiceMatrix / Env
397   } ,
398   NiceTabular / xdots .inherit:n = NiceMatrix / xdots ,
399   NiceArray .inherit:n =
400   {
401     NiceMatrix / Global ,

```

```

402     NiceMatrix / Env ,
403   } ,
404   NiceArray / xdots .inherit:n = NiceMatrix / xdots ,
405   pNiceArray .inherit:n =
406   {
407     NiceMatrix / Global ,
408     NiceMatrix / Env ,
409   } ,
410   pNiceArray / xdots .inherit:n = NiceMatrix / xdots
411 }

```

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

```

412 \keys_define:nn { NiceMatrix / NiceMatrixOptions }
413 {
414   last-col .code:n = \tl_if_empty:nF { #1 }
415                 { \@@_error:n { last-col-non-empty-for-NiceMatrixOptions } }
416                 \int_zero:N \l_@@_last_col_int ,
417   small .bool_set:N = \l_@@_small_bool ,
418   small .value_forbidden:n = true ,

```

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

```

419   renew-matrix .code:n = \@@_renew_matrix: ,
420   renew-matrix .value_forbidden:n = true ,
421   transparent .meta:n = { renew-dots , renew-matrix } ,
422   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}`.

```

423   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.

```

424   columns-width .code:n =
425   \str_if_eq:nnTF { #1 } { auto }
426   { \@@_error:n { Option-auto-for-columns-width } }
427   { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,

```

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

```

428   allow-duplicate-names .code:n =
429   \@@_msg_redirect_name:nn { Duplicate-name } { none } ,
430   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`).

```

431   letter-for-dotted-lines .code:n =
432   {
433     \int_compare:nTF { \tl_count:n { #1 } = 1 }
434     { \str_set:Nx \l_@@_letter_for_dotted_lines_str { #1 } }
435     { \@@_error:n { Bad-value-for-letter-for-dotted-lines } }
436   } ,
437   letter-for-dotted-lines .value_required:n = true ,
438   unknown .code:n = \@@_error:n { Unknown-key-for-NiceMatrixOptions }
439 }

```

```

440 \str_new:N \l_@@_letter_for_dotted_lines_str
441 \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.

```

442 \NewDocumentCommand \NiceMatrixOptions { m }
443 { \keys_set:nn { NiceMatrix / NiceMatrixOptions } { #1 } }

```

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

```

444 \keys_define:nn { NiceMatrix / NiceMatrix }
445 {
446   last-col .code:n = \tl_if_empty:nTF {#1}
447   {
448     \bool_set_true:N \l_@@_last_col_without_value_bool
449     \int_set:Nn \l_@@_last_col_int { -1 }
450   }
451   { \int_set:Nn \l_@@_last_col_int { #1 } } ,
452   l .code:n = \tl_set:Nn \l_@@_type_of_col_tl L ,
453   r .code:n = \tl_set:Nn \l_@@_type_of_col_tl R ,
454   L .code:n = \tl_set:Nn \l_@@_type_of_col_tl L ,
455   R .code:n = \tl_set:Nn \l_@@_type_of_col_tl R ,
456   S .code:n = \bool_if:NTF \c_@@_siunitx_loaded_bool
457   { \tl_set:Nn \l_@@_type_of_col_tl S }
458   { \@@_error:n { option~S~without~siunitx } } ,
459   small .bool_set:N = \l_@@_small_bool ,
460   small .value_forbidden:n = true ,
461   unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix }
462 }

```

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

```

463 \keys_define:nn { NiceMatrix / NiceArray }
464 {

```

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.

```

465   small .bool_set:N = \l_@@_small_bool ,
466   small .value_forbidden:n = true ,
467   last-col .code:n = \tl_if_empty:nF { #1 }
468   { \@@_error:n { last-col~non-empty~for~NiceArray } }
469   \int_zero:N \l_@@_last_col_int ,
470   unknown .code:n = \@@_error:n { Unknown~option~for~NiceArray }
471 }
472 \keys_define:nn { NiceMatrix / pNiceArray }
473 {
474   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
475   last-col .code:n = \tl_if_empty:nF {#1}
476   { \@@_error:n { last-col~non-empty~for~NiceArray } }
477   \int_zero:N \l_@@_last_col_int ,
478   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
479   small .bool_set:N = \l_@@_small_bool ,
480   small .value_forbidden:n = true ,
481   unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix }
482 }

```

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

```

483 \keys_define:nn { NiceMatrix / NiceTabular }

```

```

484 {
485   unknown .code:n = \@@_error:n { Unknown-option-for-NiceTabular }
486 }

```

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}`).

```

487 \cs_new_protected:Npn \@@_Cell:
488 {

```

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

```

489   \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.

```

490   \int_compare:nNnT \c@jCol = 1
491     { \int_compare:nNnT \l_@@_first_col_int = 1 \@@_begin_of_row: }
492   \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 potential `\c_math_toggle_token` also).

```

493   \hbox_set:Nw \l_@@_cell_box
494   \bool_if:NF \l_@@_NiceTabular_bool
495   {
496     \c_math_toggle_token
497     \bool_if:NT \l_@@_small_bool \scriptstyle
498   }

```

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.

```

499   \int_compare:nNnTF \c@iRow = 0
500   {
501     \int_compare:nNnT \c@jCol > 0
502     {
503       \l_@@_code_for_first_row_tl
504       \xglobal \colorlet { nicematrix-first-row } { . }
505     }
506   }
507   {
508     \int_compare:nNnT \c@iRow = \l_@@_last_row_int
509     {
510       \l_@@_code_for_last_row_tl
511       \xglobal \colorlet { nicematrix-last-row } { . }
512     }
513   }
514 }

```

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.

```

515 \cs_new_protected:Npn \@@_begin_of_row:
516 {
517   \int_gincr:N \c@iRow
518   \dim_gset_eq:NN \g_@@_dp_ante_last_row_dim \g_@@_dp_last_row_dim
519   \dim_gset:Nn \g_@@_dp_last_row_dim { \box_dp:N \@arstrutbox }
520   \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }

```

```

521 \pgfpicture
522 \pgfrememberpicturepositiononpagetrue
523 \pgfcoordinate
524 { \l_@@_env: - row - \int_use:N \c@iRow - base }
525 { \pgfpoint \c_zero_dim { 0.5 \arrayrulewidth } }
526 \str_if_empty:NF \l_@@_name_str
527 {
528   \pgfnodealias
529   { \l_@@_name_str - row - \int_use:N \c@iRow - base }
530   { \l_@@_env: - row - \int_use:N \c@iRow - base }
531 }
532 \endpgfpicture
533 }

```

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.

```

534 \cs_new_protected:Npn \@@_update_for_first_and_last_row:
535 {
536   \int_compare:nNnTF \c@iRow = 0
537   {
538     \dim_gset:Nn \g_@@_dp_row_zero_dim
539     { \dim_max:nn \g_@@_dp_row_zero_dim { \box_dp:N \l_@@_cell_box } }
540     \dim_gset:Nn \g_@@_ht_row_zero_dim
541     { \dim_max:nn \g_@@_ht_row_zero_dim { \box_ht:N \l_@@_cell_box } }
542   }
543   {
544     \int_compare:nNnT \c@iRow = 1
545     {
546       \dim_gset:Nn \g_@@_ht_row_one_dim
547       { \dim_max:nn \g_@@_ht_row_one_dim { \box_ht:N \l_@@_cell_box } }
548     }
549   }
550 }
551 \cs_new_protected:Npn \@@_end_Cell:
552 {
553   \bool_if:NF \l_@@_NiceTabular_bool \c_math_toggle_token
554   \hbox_set_end:
555   \box_set_ht:Nn \l_@@_cell_box
556   { \box_ht:N \l_@@_cell_box + \l_@@_cell_space_top_limit_dim }
557   \box_set_dp:Nn \l_@@_cell_box
558   { \box_dp:N \l_@@_cell_box + \l_@@_cell_space_bottom_limit_dim }

```

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”).

```

559 \dim_gset:Nn \g_@@_max_cell_width_dim
560 { \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”.

```

561 \@@_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.

```

562 \bool_if:NTF \g_@@_empty_cell_bool
563 { \box_use_drop:N \l_@@_cell_box }
564 {
565   \dim_compare:nNnTF { \box_wd:N \l_@@_cell_box } > \c_zero_dim
566   \@@_node_for_the_cell:
567   { \box_use_drop:N \l_@@_cell_box }
568 }
569 \bool_gset_false:N \g_@@_empty_cell_bool
570 }

```

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

```

571 \cs_new_protected:Npn \@@_node_for_the_cell:
572 {
573   \pgfpicture
574   \pgfsetbaseline \c_zero_dim
575   \pgfrememberpicturepositiononpagetrue
576   \pgfset
577   {
578     inner~sep = \c_zero_dim ,
579     minimum~width = \c_zero_dim
580   }
581   \pgfnode
582   { rectangle }
583   { base }
584   { \box_use_drop:N \l_@@_cell_box }
585   { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
586   { }
587   \str_if_empty:NF \l_@@_name_str
588   {
589     \pgfnodealias
590     { \l_@@_name_str - \int_use:N \c@iRow - \int_use:N \c@jCol }
591     { \@@_env: - \int_use:N \c@iRow - \int_use:N \c@jCol }
592   }
593   \endpgfpicture
594 }

```

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}

```

```

595 \cs_new_protected:Npn \@@_instruction_of_type:nn #1 #2
596 {

```

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).

```

597   \tl_gput_right:cx
598   { g_@@_ #1 _ lines _ tl }
599   {
600     \use:c { @@ _ draw _ #1 : nnn }
601     { \int_use:N \c@iRow }
602     { \int_use:N \c@jCol }
603     { \exp_not:n { #2 } }
604   }
605 }

```

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`.

```

606 \cs_new_protected:Npn \@@_array:
607 {
608   \bool_if:NTF \c_@@_revtex_bool
609   {
610     \cs_set_eq:NN \@acol1 \@arrayacol
611     \cs_set_eq:NN \@acolr \@arrayacol
612     \cs_set_eq:NN \@acol \@arrayacol
613     \cs_set:Npn \@halignto { }
614     \@array@array
615   }
616   \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.

```

617   [ \str_if_eq:VnTF \l_@@_baseline_str c c t ]
618 }

```

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.

```

619 \cs_set_eq:NN \@@_old_ialign: \ialign

```

The following command creates a row node (and not a row of nodes!).

```

620 \cs_new_protected:Npn \@@_create_row_node:
621 {

```

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

```

622   \hbox
623   {
624     \bool_if:NT \l_@@_code_before_bool
625     {
626       \vtop
627       {
628         \skip_vertical:N 0.5\arrayrulewidth
629         \pgfsys@markposition { \@@_env: - row - \@@_succ:n \c@iRow }
630         \skip_vertical:N -0.5\arrayrulewidth
631       }
632     }
633     \pgfpicture
634     \pgfrememberpicturepositiononpagetrue
635     \pgfcoordinate { \@@_env: - row - \@@_succ:n \c@iRow }
636     { \pgfpoint \c_zero_dim { - 0.5 \arrayrulewidth } }
637     \str_if_empty:NF \l_@@_name_str
638     {
639       \pgfnodealias
640       { \l_@@_name_str - row - \int_use:N \c@iRow }

```

```

641         { \@@_env: - row - \int_use:N \c@iRow }
642     }
643     \endpgfpicture
644 }
645 }

```

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

```

646 \cs_new:Npn \@@_everycr: { \noalign { \@@_everycr_i: } }
647 \cs_new_protected:Npn \@@_everycr_i:
648 {
649     \int_gzero:N \c@jCol
650     \bool_if:NF \g_@@_row_of_col_done_bool
651     {
652         \@@_create_row_node:

```

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

```

653         \bool_if:NT \l_@@_hlines_bool
654     {

```

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.

```

655         \int_compare:nNnT \c@iRow > { -1 }
656     {
657         \int_compare:nNnF \c@iRow = \l_@@_last_row_int

```

The command `\CT@arc@` is a command of `colortbl` which sets the color of the rules in the array. The package `nicematrix` uses it even if `colortbl` is not loaded. We use a TeX group in order to limit the scope of `\CT@arc@`.

```

658         { { \CT@arc@ \hrule height \arrayrulewidth } }
659     }
660 }
661 }
662 }

```

The command `\@@_newcolumntype` is the command `\newcolumntype` of `array` without the warnings for redefinitions of columns types (we will use it to redefine the columns types `w`, `W`, `p`, `m` and `b`).

```

663 \cs_set_protected:Npn \@@_newcolumntype #1
664 {
665     \cs_if_free:cT { NC @ find @ #1 }
666     { \NC@list \expandafter { \the \NC@list \NC@do #1 } }
667     \cs_set:cpn {NC @ find @ #1 } ##1 #1 { \NC@ { ##1 } }
668     \peek_meaning:NTF [
669         { \newcol@ #1 }
670         { \newcol@ #1 [ 0 ] }
671     }

```

The following command will be used to redefine the column types `p`, `m` and `b`. That means that it will be used three times. The first argument is the letter of the column type (`p`, `m` or `b`). The second is the letter of position for the environment `{minipage}` (`t`, `c` or `b`).

```

672 \cs_new_protected:Npn \@@_define_columntype:nn #1 #2
673 {

```

We don’t want a warning for redefinition of the column type. That’s why we use `\@@_newcolumntype` and not `\newcolumntype`.

```

674     \@@_newcolumntype #1 [ 1 ]
675     {
676         > {
677             \@@_Cell:
678             \begin { minipage } [ #2 ] { ##1 }
679             \mode_leave_vertical: \box_use:N \@arstrutbox
680         }

```


Here, we put `c` but we would have the result with `l` or `r`.

```

681         c
682     < { \box_use:N \@arstrutbox \end { minipage } \@_end_Cell: }
683 }
684 }

```

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

```

685 \cs_new_protected:Npn \@_pre_array:
686 {

```

If `booktabs` is loaded, we have to patch the macro `\@BTnormal` which is a macro of `booktabs`. The macro `\@BTnormal` draws an horizontal rule but it occurs after a vertical skip done by a low level TeX command. When this macro `\@BTnormal` occurs, the `row` node has yet been inserted by `nicematrix` *before* the vertical skip (and thus, at a wrong place). That why we decide to create a new `row` node (for the same row). We patch the macro `\@BTnormal` to create this `row` node. This new `row` node will overwrite the previous definition of that `row` node and we have managed to avoid the error messages of that redefinition ²⁹.

```

687     \bool_if:NT \c_@@_booktabs_loaded_bool
688     { \tl_put_left:Nn \@BTnormal \@_create_row_node: }
689     \box_clear_new:N \l_@@_cell_box
690     \cs_if_exist:NT \theiRow
691     { \int_set_eq:NN \l_@@_old_iRow_int \c@iRow }
692     \int_gzero_new:N \c@iRow
693     \cs_if_exist:NT \thejCol
694     { \int_set_eq:NN \l_@@_old_jCol_int \c@jCol }
695     \int_gzero_new:N \c@jCol
696     \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}`).

```

697     \bool_if:NT \l_@@_small_bool
698     {
699         \cs_set:Npn \arraystretch { 0.47 }
700         \dim_set:Nn \arraycolsep { 1.45 pt }
701     }

```

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`.

```

702     \cs_set:Npn \ialign
703     {
704         \bool_if:NT \l_@@_NiceTabular_bool
705         { \dim_set_eq:NN \arraycolsep \@_old_arraycolsep_dim }
706         \bool_if:NTF \c_@@_colortbl_loaded_bool
707         {
708             \CT@everycr
709             {
710                 \noalign { \cs_gset_eq:NN \CT@row@color \prg_do_nothing: }
711                 \@_everycr:
712             }
713         }
714         { \everycr { \@_everycr: } }
715         \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`³⁰ and `\extrarowheight`

²⁹cf. `\nicematrix@redefine@check@rerun`

³⁰The option `small` of `nicematrix` changes (among other) the value of `\arraystretch`. This is done, of course, before the call of `{array}`.

(of array). That box is inserted (via `\@arstrut`) in the beginning of each row of the array. That's why we use the dimensions of that box to initialize the variables which will be the dimensions of the potential first and last row of the environment. This initialization must be done after the creation of `\@arstrutbox` and that's why we do it in the `\ialign`.

```

716 \dim_gzero_new:N \g_@@_dp_row_zero_dim
717 \dim_gset:Nn \g_@@_dp_row_zero_dim { \box_dp:N \@arstrutbox }
718 \dim_gzero_new:N \g_@@_ht_row_zero_dim
719 \dim_gset:Nn \g_@@_ht_row_zero_dim { \box_ht:N \@arstrutbox }
720 \dim_gzero_new:N \g_@@_ht_row_one_dim
721 \dim_gset:Nn \g_@@_ht_row_one_dim { \box_ht:N \@arstrutbox }
722 \dim_gzero_new:N \g_@@_dp_ante_last_row_dim
723 \dim_gzero_new:N \g_@@_ht_last_row_dim
724 \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
725 \dim_gzero_new:N \g_@@_dp_last_row_dim
726 \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`.³¹

```

727 \cs_set_eq:NN \ialign \@_old_ialign:
728 \halign
729 }

```

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).

```

730 \cs_set_eq:NN \@_old_ldots \ldots
731 \cs_set_eq:NN \@_old_cdots \cdots
732 \cs_set_eq:NN \@_old_vdots \vdots
733 \cs_set_eq:NN \@_old_ddots \ddots
734 \cs_set_eq:NN \@_old_iddots \iddots
735 \cs_set_eq:NN \firsthline \hline
736 \cs_set_eq:NN \lasthline \hline
737 \cs_set_eq:NN \Ldots \@_Ldots
738 \cs_set_eq:NN \Cdots \@_Cdots
739 \cs_set_eq:NN \Vdots \@_Vdots
740 \cs_set_eq:NN \Ddots \@_Ddots
741 \cs_set_eq:NN \Iddots \@_Iddots
742 \cs_set_eq:NN \hdottedline \@_hdottedline:
743 \cs_set_eq:NN \Hspace \@_Hspace:
744 \cs_set_eq:NN \Hdotsfor \@_Hdotsfor:
745 \cs_set_eq:NN \Vdotsfor \@_Vdotsfor:
746 \cs_set_eq:NN \multicolumn \@_multicolumn:nnn
747 \cs_set_eq:NN \Block \@_Block:
748 \cs_set_eq:NN \rotate \@_rotate:
749 \cs_set_eq:NN \OnlyMainNiceMatrix \@_OnlyMainNiceMatrix:n
750 \cs_set_eq:NN \dotfill \@_dotfill:
751 \cs_set_eq:NN \CodeAfter \@_CodeAfter:n
752 \cs_set_eq:NN \slashbox \@_slashbox:nn
753 \bool_if:NT \l_@@_renew_dots_bool
754 {
755 \cs_set_eq:NN \ldots \@_Ldots
756 \cs_set_eq:NN \cdots \@_Cdots
757 \cs_set_eq:NN \vdots \@_Vdots
758 \cs_set_eq:NN \ddots \@_Ddots
759 \cs_set_eq:NN \iddots \@_Iddots
760 \cs_set_eq:NN \dots \@_Ldots
761 \cs_set_eq:NN \hdotsfor \@_Hdotsfor:
762 }

```

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`,

³¹The user will probably not use 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).

```
763 \seq_gclear_new:N \g_@@_multicolumn_cells_seq
764 \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).

```
765 \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`).

```
766 \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.

```
767 \int_gzero_new:N \g_@@_col_total_int
768 \cs_set_eq:NN \@ifnextchar \new@ifnextchar
```

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}`. We use `\@@_newcolumntype` because it will be slightly quicker than `\newcolumntype`.

```
769 \@@_newcolumntype L { > \@@_Cell: l < \@@_end_Cell: }
770 \@@_newcolumntype C { > \@@_Cell: c < \@@_end_Cell: }
771 \@@_newcolumntype R { > \@@_Cell: r < \@@_end_Cell: }
```

We redefine the column types p, m and b. The command `\@@_define_columntype:nn` is only used here.

```
772 \@@_define_columntype:nn p t
773 \@@_define_columntype:nn m c
774 \@@_define_columntype:nn b b
```

We redefine the column types w and W. We use `\@@_newcolumntype` instead of `\newcolumntype` because we don’t want warnings for column types already defined.

```
775 \@@_newcolumntype w [ 2 ]
776 {
777   > {
778     \hbox_set:Nw \l_@@_cell_box
779     \@@_Cell:
780   }
781   c
782   < {
783     \@@_end_Cell:
784     \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.

```
785 \makebox [ ##2 ] [ \str_lowercase:n { ##1 } ]
786 { \box_use_drop:N \l_@@_cell_box }
787 }
788 }
789 \@@_newcolumntype W [ 2 ]
790 {
791   > {
792     \hbox_set:Nw \l_@@_cell_box
793     \@@_Cell:
794   }
795   c
796   < {
797     \@@_end_Cell:
798     \hbox_set_end:
```

```

799         \cs_set_eq:NN \hss \hfil
800         \makebox [ ##2 ] [ \str_lowercase:n { ##1 } ]
801         { \box_use_drop:N \l_@@_cell_box }
802     }
803 }

```

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).

```

804     \tl_set_rescan:Nno
805     \l_@@_letter_for_dotted_lines_str { } \l_@@_letter_for_dotted_lines_str
806     \exp_args:NV \newcolumntype \l_@@_letter_for_dotted_lines_str
807     {
808         !
809     }

```

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).

```

810     \int_compare:nNnF \c@iRow = 0
811     {
812         \int_compare:nNnF \c@iRow = \l_@@_last_row_int
813         { \skip_horizontal:N 2\l_@@_radius_dim }
814     }

```

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`.

```

815     \int_compare:nNnT \c@jCol > \g_@@_last_vdotted_col_int
816     {
817         \int_gset_eq:NN \g_@@_last_vdotted_col_int \c@jCol
818         \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`.

```

819         { \@@_vdottedline:n { \int_use:N \c@jCol } }
820     }
821 }
822 }
823 \int_gzero_new:N \g_@@_last_vdotted_col_int
824 \bool_if:NT \c_@@_siunitx_loaded_bool \@@_renew_NC@rewrite@S:
825 \int_gset:Nn \g_@@_last_vdotted_col_int { -1 }
826 \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.

```

827     \tl_gclear_new:N \g_@@_Cdots_lines_tl
828     \tl_gclear_new:N \g_@@_Ldots_lines_tl
829     \tl_gclear_new:N \g_@@_Vdots_lines_tl

```

```

830 \tl_gclear_new:N \g_@@Ddots_lines_tl
831 \tl_gclear_new:N \g_@@Iddots_lines_tl
832 \tl_gclear_new:N \g_@@HVDotsfor_lines_tl
833 }

```

The environment {NiceArrayWithDelims}

```

834 \NewDocumentEnvironment { NiceArrayWithDelims } { m m O { } m ! O { } }
835 {
836   \tl_set:Nn \l_@@_left_delim_tl { #1 }
837   \tl_set:Nn \l_@@_right_delim_tl { #2 }
838   \bool_gset_false:N \g_@@_row_of_col_done_bool
839   \str_if_empty:NT \g_@@_name_env_str
840     { \str_gset:Nn \g_@@_name_env_str { NiceArrayWithDelims } }
841   \@@_adapt_S_column:
842   \bool_if:NTF \l_@@_NiceTabular_bool
843     \mode_leave_vertical:
844     \@@_test_if_math_mode:
845   \bool_if:NT \l_@@_in_env_bool { \@@_fatal:n { Yet~in~env } }
846   \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).

```

847   \cs_if_exist:NT \tikz@library@external@loaded
848   {
849     \tikzset { external / export = false }
850     \cs_if_exist:NT \ifstandalone
851       { \tikzset { external / optimize = false } }
852   }

```

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

```

853   \int_gincr:N \g_@@_env_int
854   \bool_if:NF \l_@@_block_auto_columns_width_bool
855     { \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.

```

856   \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`.

```

857   \bool_if:NTF \l_@@_NiceArray_bool
858     { \keys_set:nn { NiceMatrix / NiceArray } }
859     { \keys_set:nn { NiceMatrix / pNiceArray } }
860   { #3 , #5 }

```

If the key `code-before` is used, we have to create the `col` nodes and the `row` nodes before the creation of the array. First, we have to test whether the size of the array has been written in the `aux` file in a previous run. In this case, a command `\@@_size_nb_of_env:` has been created.

```

861   \bool_if:NT \l_@@_code_before_bool
862   {
863     \seq_if_exist:cT { @@_size_ \int_use:N \g_@@_env_int _ seq }
864     {

```

First, we give values to the LaTeX counters `iRow` and `jCol`. We remind that, in the `code-before` (and in the `code-after`) they represent the numbers of rows and columns of the array (without the potential last row and last column).

```

865       \int_zero_new:N \c@iRow
866       \int_set:Nn \c@iRow
867         { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 2 }
868       \int_zero_new:N \c@jCol
869       \int_set:Nn \c@jCol

```

```

870      { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 4 }

```

We have to adjust the values of `\c@iRow` and `\c@jCol` to take into account the potential last row and last column. A value of -2 for `\l_@@_last_row_int` means that there is no last row. Idem for the columns.

```

871      \int_compare:nNnF \l_@@_last_row_int = { -2 }
872      { \int_decr:N \c@iRow }
873      \int_compare:nNnF \l_@@_last_col_int = { -2 }
874      { \int_decr:N \c@jCol }

```

Now, we will create all the `col` nodes and `row` nodes with the informations written in the `aux` file. You use the technique described in the page 1229 of `pgfmanual.pdf`, version 3.1.4b.

```

875      \pgfsys@markposition { \@@_env: - position }
876      \pgfsys@getposition { \@@_env: - position } \@@_picture_position:
877      \pgfpicture

```

First, the creation of the `row` nodes.

```

878      \int_step_inline:nnn
879      { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 1 }
880      { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 2 + 1 }
881      {
882        \pgfsys@getposition { \@@_env: - row - ##1 } \@@_node_position:
883        \pgfcoordinate { \@@_env: - row - ##1 }
884        { \pgfpointdiff \@@_picture_position: \@@_node_position: }
885      }

```

Now, the creation of the `col` nodes.

```

886      \int_step_inline:nnn
887      { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 3 }
888      { \seq_item:cn { @@_size_ \int_use:N \g_@@_env_int _ seq } 4 + 1 }
889      {
890        \pgfsys@getposition { \@@_env: - col - ##1 } \@@_node_position:
891        \pgfcoordinate { \@@_env: - col - ##1 }
892        { \pgfpointdiff \@@_picture_position: \@@_node_position: }
893      }
894      \endpgfpicture
895      \group_begin:
896      \bool_if:NT \c_@@_tikz_loaded_bool
897      {

```

Be careful: we must *not* put “remember picture” in the `\pgfset`.

```

898      \pgfset
899      {
900        every-picture / .style =
901        { overlay , name-prefix = \@@_env: - }
902      }
903      }
904      \cs_set_eq:NN \cellcolor \@@_cellcolor:nn
905      \cs_set_eq:NN \rectanglecolor \@@_rectanglecolor:nnn
906      \cs_set_eq:NN \rowcolor \@@_rowcolor:nn
907      \cs_set_eq:NN \rowcolors \@@_rowcolors:nnn
908      \cs_set_eq:NN \columncolor \@@_columncolor:nn
909      \cs_set_eq:NN \chessboardcolors \@@_chessboardcolors:nn

```

We compose the `code-before` in math mode in order to nullify the spaces put by the user between instructions in the `code-before`.

```

910      \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
911      \l_@@_code_before_tl
912      \bool_if:NT \l_@@_NiceTabular_bool \c_math_toggle_token
913      \group_end:
914    }
915  }

```

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).

```

916      \int_compare:nNnT \l_@@_last_row_int > { -2 }
917      {

```

```

918 \tl_put_right:Nn \@@_update_for_first_and_last_row:
919 {
920   \dim_gset:Nn \g_@@_ht_last_row_dim
921   { \dim_max:nn \g_@@_ht_last_row_dim { \box_ht:N \l_@@_cell_box } }
922   \dim_gset:Nn \g_@@_dp_last_row_dim
923   { \dim_max:nn \g_@@_dp_last_row_dim { \box_dp:N \l_@@_cell_box } }
924 }
925 }
926 \int_compare:nNnT \l_@@_last_row_int = { -1 }
927 {
928   \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.

```

929 \str_if_empty:NTF \l_@@_name_str
930 {
931   \cs_if_exist:cT { @@_last_row_ \int_use:N \g_@@_env_int }
932   {
933     \int_set:Nn \l_@@_last_row_int
934     { \use:c { @@_last_row_ \int_use:N \g_@@_env_int } }
935   }
936 }
937 {
938   \cs_if_exist:cT { @@_last_row_ \l_@@_name_str }
939   {
940     \int_set:Nn \l_@@_last_row_int
941     { \use:c { @@_last_row_ \l_@@_name_str } }
942   }
943 }
944 }

```

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).

```

945 \int_compare:nNnT \l_@@_last_col_int = { -1 }
946 {
947   \str_if_empty:NTF \l_@@_name_str
948   {
949     \cs_if_exist:cT { @@_last_col_ \int_use:N \g_@@_env_int }
950     {
951       \int_set:Nn \l_@@_last_col_int
952       { \use:c { @@_last_col_ \int_use:N \g_@@_env_int } }
953     }
954   }
955   {
956     \cs_if_exist:cT { @@_last_col_ \l_@@_name_str }
957     {
958       \int_set:Nn \l_@@_last_col_int
959       { \use:c { @@_last_col_ \l_@@_name_str } }
960     }
961   }
962 }

```

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

```

963 \@@_pre_array:

```

We compute the width of the two delimiters.

```

964 \dim_zero_new:N \l_@@_left_delim_dim
965 \dim_zero_new:N \l_@@_right_delim_dim
966 \bool_if:NTF \l_@@_NiceArray_bool
967 {
968   \dim_gset:Nn \l_@@_left_delim_dim { 2 \arraycolsep }
969   \dim_gset:Nn \l_@@_right_delim_dim { 2 \arraycolsep }
970 }
971 {

```

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

```

972 \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 #1 $ }
973 \dim_set:Nn \l_@@_left_delim_dim { \box_wd:N \l_tmpa_box }
974 \hbox_set:Nn \l_tmpa_box { $ \bBigg@ 5 #2 $ }
975 \dim_set:Nn \l_@@_right_delim_dim { \box_wd:N \l_tmpa_box }
976 }

```

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

```

977 \box_clear_new:N \l_@@_the_array_box

```

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

```

978 \tl_set:Nn \l_tmpa_tl { #4 }
979 \int_compare:nNnTF \l_@@_first_col_int = 0
980 { \tl_put_left:NV \l_tmpa_tl \c_@@_preamble_first_col_tl }
981 {
982   \bool_lazy_all:nT
983   {
984     \l_@@_NiceArray_bool
985     { \bool_not_p:n \l_@@_NiceTabular_bool }
986     { \bool_not_p:n \l_@@_vlines_bool }
987     { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
988   }
989   { \tl_put_left:Nn \l_tmpa_tl { @ { } } }
990 }
991 \int_compare:nNnTF \l_@@_last_col_int > { -1 }
992 { \tl_put_right:NV \l_tmpa_tl \c_@@_preamble_last_col_tl }
993 {
994   \bool_lazy_all:nT
995   {
996     \l_@@_NiceArray_bool
997     { \bool_not_p:n \l_@@_NiceTabular_bool }
998     { \bool_not_p:n \l_@@_vlines_bool }
999     { \bool_not_p:n \l_@@_exterior_arraycolsep_bool }
1000   }
1001   { \tl_put_right:Nn \l_tmpa_tl { @ { } } }
1002 }
1003 \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).

```

1004 \hbox_set:Nw \l_@@_the_array_box

```

Here is a trick. We will call `\array` and, at the beginning, `\array` will set `\col@sep` equal to the current value of `\arraycolsep`. In we are in an environment `{NiceTabular}`, we would like that `\array` sets `\col@sep` equal to the current value of `\tabcolsep`. That's why we set `\arraycolsep` equal to `\tabcolsep`. However, the value of `\tabcolsep` in each cell of the array should be equal to the current value of `\tabcolsep` outside `{NiceTabular}`. That's why we save the current value of `\arraycolsep` and we will restore the value just before the `\halign`. It's possible because we do a redefinition of `\ialign` (see just below).

```

1005 \bool_if:NT \l_@@_NiceTabular_bool
1006 {
1007   \dim_set_eq:NN \@@_old_arraycolsep_dim \arraycolsep
1008   \dim_set_eq:NN \arraycolsep \tabcolsep
1009 }

```

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.

```

1010 \bool_if:NT \l_@@_vlines_bool
1011 {
1012   \dim_add:Nn \arraycolsep { 0.5 \arrayrulewidth }
1013   \skip_horizontal:N 0.5\arrayrulewidth

```



```

1014     }
1015     \skip_horizontal:N \l_@@_left_margin_dim
1016     \skip_horizontal:N \l_@@_extra_left_margin_dim
1017     \c_math_toggle_token
1018     \bool_if:NTF \l_@@_light_syntax_bool
1019       { \use:c { @@-light-syntax } }
1020       { \use:c { @@-normal-syntax } }
1021   }
1022   {
1023     \bool_if:NTF \l_@@_light_syntax_bool
1024       { \use:c { end @@-light-syntax } }
1025       { \use:c { end @@-normal-syntax } }
1026     \c_math_toggle_token
1027     \skip_horizontal:N \l_@@_right_margin_dim
1028     \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.

```

1029     \bool_if:NT \l_@@_vlines_bool { \skip_horizontal:N 0.5\arrayrulewidth }
1030     \hbox_set_end:

```

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

If 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).

```

1031     \int_compare:nNnT \l_@@_last_row_int > { -2 }
1032     {
1033       \bool_if:NF \l_@@_last_row_without_value_bool
1034       {
1035         \int_compare:nNnF \l_@@_last_row_int = \c@iRow
1036         {
1037           \@@_error:n { Wrong~last~row }
1038           \int_gset_eq:NN \l_@@_last_row_int \c@iRow
1039         }
1040       }
1041     }

```

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”.³²

```

1042     \int_gset_eq:NN \c@jCol \g_@@_col_total_int
1043     \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.

```

1044     \int_gset_eq:NN \g_@@_row_total_int \c@iRow
1045     \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. 66).

```

1046     \int_compare:nNnT \l_@@_first_col_int = 0
1047     {
1048       \skip_horizontal:N \arraycolsep
1049       \skip_horizontal:N \g_@@_width_first_col_dim
1050     }

```

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}`.

```

1051     \bool_if:NTF \l_@@_NiceArray_bool
1052     {

```

³²We remind that the potential “first column” (exterior) has the number 0.

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`.

```

1053   \str_if_eq:VnTF \l_@@_baseline_str { b }
1054   {
1055     \pgfpicture
1056     \@@_qpoint:n { row - 1 }
1057     \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1058     \@@_qpoint:n { row - \int_use:N \c@iRow - base }
1059     \dim_gsub:Nn \g_tmpa_dim \pgf@y
1060     \endpgfpicture
1061     \int_compare:nNnT \l_@@_first_row_int = 0
1062     {
1063       \dim_gadd:Nn \g_tmpa_dim
1064       { \g_@@_ht_row_zero_dim + \g_@@_dp_row_zero_dim }
1065     }
1066     \box_move_up:nn \g_tmpa_dim { \box_use_drop:N \l_@@_the_array_box }
1067   }
1068   {
1069     \str_if_eq:VnTF \l_@@_baseline_str { c }
1070     { \box_use_drop:N \l_@@_the_array_box }
1071     {

```

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

```

1072     \str_if_eq:VnT \l_@@_baseline_str { t }
1073     { \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`.

```

1074     \int_set:Nn \l_tmpa_int \l_@@_baseline_str
1075     \bool_if:nT
1076     {
1077       \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int
1078       || \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int
1079     }
1080     {
1081       \@@_error:n { bad-value-for-baseline }
1082       \int_set:Nn \l_tmpa_int 1
1083     }
1084     \pgfpicture
1085     \@@_qpoint:n { row - 1 }
1086     \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1087     \@@_qpoint:n { row - \int_use:N \l_tmpa_int - base }
1088     \dim_gsub:Nn \g_tmpa_dim \pgf@y
1089     \endpgfpicture
1090     \int_compare:nNnT \l_@@_first_row_int = 0
1091     {
1092       \dim_gadd:Nn \g_tmpa_dim
1093       { \g_@@_ht_row_zero_dim + \g_@@_dp_row_zero_dim }
1094     }
1095     \box_move_up:nn \g_tmpa_dim
1096     { \box_use_drop:N \l_@@_the_array_box }
1097   }
1098 }
1099 }

```

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).

```

1100   {
1101     \int_compare:nNnTF \l_@@_first_row_int = 0
1102     {
1103       \dim_set_eq:NN \l_tmpa_dim \g_@@_dp_row_zero_dim
1104       \dim_add:Nn \l_tmpa_dim \g_@@_ht_row_zero_dim
1105     }
1106     { \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”.³³

```

1107   \int_compare:nNnTF \l_@@_last_row_int > { -2 }
1108   {
1109     \dim_set_eq:NN \l_tmpb_dim \g_@@_ht_last_row_dim
1110     \dim_add:Nn \l_tmpb_dim \g_@@_dp_last_row_dim
1111   }
1112   { \dim_zero:N \l_tmpb_dim }
1113   \hbox_set:Nn \l_tmpa_box
1114   {
1115     \c_math_toggle_token
1116     \left #1
1117     \vcenter
1118     {

```

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.

```

1119     \skip_vertical:N -\l_tmpa_dim
1120     \hbox
1121     {
1122       \bool_if:NTF \l_@@_NiceTabular_bool
1123       { \skip_horizontal:N -\tabcolsep }
1124       { \skip_horizontal:N -\arraycolsep }
1125       \box_use_drop:N \l_@@_the_array_box
1126       \bool_if:NTF \l_@@_NiceTabular_bool
1127       { \skip_horizontal:N -\tabcolsep }
1128       { \skip_horizontal:N -\arraycolsep }
1129     }

```

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

```

1130     \skip_vertical:N -\l_tmpb_dim
1131   }
1132   \right #2
1133   \c_math_toggle_token
1134 }

```

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.

```

1135   \bool_if:NTF \l_@@_max_delimiter_width_bool
1136   { \@@_put_box_in_flow_bis:nn { #1 } { #2 } }
1137   \@@_put_box_in_flow:
1138 }

```

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. 66).

```

1139   \bool_if:NT \g_@@_last_col_found_bool
1140   {
1141     \skip_horizontal:N \g_@@_width_last_col_dim
1142     \skip_horizontal:N \arraycolsep
1143   }
1144   \@@_after_array:
1145 }

```

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`).

```

1146 \cs_new_protected:Npn \@@_put_box_in_flow:
1147 {
1148   \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }

```

³³A 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).

```

1149 \box_set_dp:Nn \l_tmpa_box { \box_dp:N \l_tmpa_box + \l_tmpb_dim }
1150 \str_if_eq:VnTF \l_@@_baseline_str { c }
1151 { \box_use_drop:N \l_tmpa_box }
1152 \@@_put_box_in_flow_i:
1153 }

```

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).

```

1154 \cs_new_protected:Npn \@@_put_box_in_flow_i:
1155 {
1156   \str_case:VnF \l_@@_baseline_str
1157   {
1158     { t } { \int_set:Nn \l_tmpa_int 1 }
1159     { b } { \int_set_eq:NN \l_tmpa_int \c@iRow }
1160   }
1161   { \int_set:Nn \l_tmpa_int \l_@@_baseline_str }
1162   \bool_if:nT
1163   {
1164     \int_compare_p:nNn \l_tmpa_int < \l_@@_first_row_int
1165     || \int_compare_p:nNn \l_tmpa_int > \g_@@_row_total_int
1166   }
1167   {
1168     \@@_error:n { bad-value-for-baseline }
1169     \int_set:Nn \l_tmpa_int 1
1170   }
1171   \pgfpicture
1172   \@@_qpoint:n { row - 1 }
1173   \dim_gset_eq:NN \g_tmpa_dim \pgf@y
1174   \@@_qpoint:n { row - \@@_succ:n \c@iRow }
1175   \dim_gadd:Nn \g_tmpa_dim \pgf@y
1176   \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).

```

1177   \@@_qpoint:n { row - \int_use:N \l_tmpa_int - base }
1178   \dim_gsub:Nn \g_tmpa_dim \pgf@y

```

We take into account the position of the mathematical axis.

```

1179   \dim_gsub:Nn \g_tmpa_dim { \fontdimen22 \textfont2 }

```

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

```

1180   \endpgfpicture
1181   \box_move_up:nn \g_tmpa_dim { \box_use_drop:N \l_tmpa_box }
1182   \box_use_drop:N \l_tmpa_box
1183 }

```

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.

```

1184 \cs_new_protected:Npn \@@_put_box_in_flow_bis:nn #1 #2
1185 {

```

We will compute the real width of both delimiters used.

```

1186   \dim_zero_new:N \l_@@_real_left_delim_dim
1187   \dim_zero_new:N \l_@@_real_right_delim_dim
1188   \hbox_set:Nn \l_tmpb_box
1189   {
1190     \c_math_toggle_token
1191     \left #1
1192     \vcenter
1193     {
1194       \vbox_to_ht:nn
1195       { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1196       { }
1197     }
1198     \right .

```

```

1199     \c_math_toggle_token
1200   }
1201   \dim_set:Nn \l_@@_real_left_delim_dim
1202     { \box_wd:N \l_tmpb_box - \nullldelimiterspace }
1203   \hbox_set:Nn \l_tmpb_box
1204     {
1205       \c_math_toggle_token
1206       \left .
1207       \vbox_to_ht:nn
1208         { \box_ht:N \l_tmpa_box + \box_dp:N \l_tmpa_box }
1209         { }
1210       \right #2
1211       \c_math_toggle_token
1212     }
1213   \dim_set:Nn \l_@@_real_right_delim_dim
1214     { \box_wd:N \l_tmpb_box - \nullldelimiterspace }

```

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

```

1215   \skip_horizontal:N \l_@@_left_delim_dim
1216   \skip_horizontal:N -\l_@@_real_left_delim_dim
1217   \@@_put_box_in_flow:
1218   \skip_horizontal:N \l_@@_right_delim_dim
1219   \skip_horizontal:N -\l_@@_real_right_delim_dim
1220 }

```

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).

```

1221 \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).

```

1222 {
1223   \peek_meaning_ignore_spaces:NTF \end
1224   { \@@_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`).

```

1225   { \exp_args:NV \@@_array: \l_tmpa_tl }
1226 }
1227 {
1228   \@@_create_col_nodes:
1229   \endarray
1230 }

```

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`).

```

1231 \NewDocumentEnvironment { @@-light-syntax } { b }
1232 {

```

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**.

```

1233   \tl_if_empty:nT { #1 } { \@@_fatal:n { empty-environment } }
1234   \tl_map_inline:nn { #1 }
1235   {
1236     \tl_if_eq:nnT { ##1 } { & }
1237     { \@@_fatal:n { ampersand-in-light-syntax } }
1238     \tl_if_eq:nnT { ##1 } { \ }
1239     { \@@_fatal:n { double-backslash-in-light-syntax } }
1240   }

```

Now, you extract the `code-after` or the body of the environment. Maybe, there is no command `\CodeAfter` in the body. That's why you put a marker `\CodeAfter` after `#1`. If there is yet a `\CodeAfter` in `#1`, this second (or third...) `\CodeAfter` will be caught in the value of `\g_@@_code_after_tl`. That doesn't matter because `\CodeAfter` will be set to *no-op* before the execution of `\g_@@_code_after_tl`.

```
1241 \@@_light_syntax_i #1 \CodeAfter \q_stop
1242 }
```

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`.

```
1243 { }
1244 \cs_new_protected:Npn \@@_light_syntax_i #1\CodeAfter #2\q_stop
1245 {
1246 \tl_gput_right:Nn \g_@@_code_after_tl { #2 }
```

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

```
1247 \seq_gclear_new:N \g_@@_rows_seq
1248 \tl_set_rescan:Nno \l_@@_end_of_row_tl { } \l_@@_end_of_row_tl
1249 \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.

```
1250 \int_compare:nNnT \l_@@_last_row_int = { -1 }
1251 { \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`).

```
1252 \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.

```
1253 \seq_gpop_left:NN \g_@@_rows_seq \l_tmpa_tl
1254 \exp_args:NV \@@_line_with_light_syntax_i:n \l_tmpa_tl
1255 \seq_map_function:NN \g_@@_rows_seq \@@_line_with_light_syntax:n
1256 \@@_create_col_nodes:
1257 \endarray
1258 }
1259 \cs_new_protected:Npn \@@_line_with_light_syntax:n #1
1260 { \tl_if_empty:nF { #1 } { \ \ \@@_line_with_light_syntax_i:n { #1 } } }
1261 \cs_new_protected:Npn \@@_line_with_light_syntax_i:n #1
1262 {
1263 \seq_gclear_new:N \g_@@_cells_seq
1264 \seq_gset_split:Nnn \g_@@_cells_seq { ~ } { #1 }
1265 \seq_gpop_left:NN \g_@@_cells_seq \l_tmpa_tl
1266 \l_tmpa_tl
1267 \seq_map_inline:Nn \g_@@_cells_seq { & ##1 }
1268 }
```

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).

```
1269 \cs_new_protected:Npn \@@_analyze_end:Nn #1 #2
1270 {
1271 \str_if_eq:VnT \g_@@_name_env_str { #2 }
1272 { \@@_fatal:n { empty~environment } }
```

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

```
1273 \end { #2 }
1274 }
```

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).

```

1275 \cs_new:Npn \@@_create_col_nodes:
1276 {
1277   \crcr
1278   \int_compare:nNnT \c@iRow = 0 { \@@_fatal:n { Zero~row } }
1279   \int_compare:nNnT \l_@@_first_col_int = 0
1280   {
1281     \omit
1282     \skip_horizontal:N -2\col@sep
1283     \bool_if:NT \l_@@_code_before_bool
1284       { \pgfsys@markposition { \@@_env: - col - 0 } }
1285     \pgfpicture
1286     \pgfrememberpicturerepositiononpagetrue
1287     \pgfcoordinate { \@@_env: - col - 0 } \pgfpointorigin
1288     \str_if_empty:NF \l_@@_name_str
1289       { \pgfnodealias { \l_@@_name_str - col - 0 } { \@@_env: - col - 0 } }
1290     \endpgfpicture
1291     &
1292   }
1293   \omit

```

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

```

1294   \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`).

```

1295   \int_compare:nNnTF \l_@@_first_col_int = 0
1296   {
1297     \bool_if:NT \l_@@_code_before_bool
1298     {
1299       \hbox
1300       {
1301         \skip_horizontal:N -0.5\arrayrulewidth
1302         \pgfsys@markposition { \@@_env: - col - 1 }
1303         \skip_horizontal:N 0.5\arrayrulewidth
1304       }
1305     }
1306     \pgfpicture
1307     \pgfrememberpicturerepositiononpagetrue
1308     \pgfcoordinate { \@@_env: - col - 1 }
1309     { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
1310     \str_if_empty:NF \l_@@_name_str
1311     { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
1312     \endpgfpicture
1313   }
1314   {
1315     \bool_if:NT \l_@@_code_before_bool
1316     {
1317       \hbox
1318       {
1319         \skip_horizontal:N 0.5 \arrayrulewidth
1320         \pgfsys@markposition { \@@_env: - col - 1 }
1321         \skip_horizontal:N -0.5\arrayrulewidth
1322       }
1323     }
1324     \pgfpicture
1325     \pgfrememberpicturerepositiononpagetrue
1326     \pgfcoordinate { \@@_env: - col - 1 }
1327     { \pgfpoint { 0.5 \arrayrulewidth } \c_zero_dim }
1328     \str_if_empty:NF \l_@@_name_str
1329     { \pgfnodealias { \l_@@_name_str - col - 1 } { \@@_env: - col - 1 } }
1330     \endpgfpicture
1331   }

```

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.

```

1332 \skip_gset:Nn \g_tmpa_skip { 0 pt+plus 1 fill }
1333 \bool_if:NF \l_@@_auto_columns_width_bool
1334 { \dim_compare:nNnT \l_@@_columns_width_dim > \c_zero_dim }
1335 {
1336   \bool_lazy_and:nnTF
1337     \l_@@_auto_columns_width_bool
1338     { \bool_not_p:n \l_@@_block_auto_columns_width_bool }
1339     { \skip_gset_eq:NN \g_tmpa_skip \g_@@_max_cell_width_dim }
1340     { \skip_gset_eq:NN \g_tmpa_skip \l_@@_columns_width_dim }
1341     \skip_gadd:Nn \g_tmpa_skip { 2 \col@sep }
1342 }
1343 \skip_horizontal:N \g_tmpa_skip
1344 \hbox
1345 {
1346   \bool_if:NT \l_@@_code_before_bool
1347   {
1348     \hbox
1349     {
1350       \skip_horizontal:N -0.5\arrayrulewidth
1351       \pgfsys@markposition { \@@_env: - col - 2 }
1352       \skip_horizontal:N 0.5\arrayrulewidth
1353     }
1354   }
1355   \pgfpicture
1356   \pgfrememberpicturepositiononpagetrue
1357   \pgfcoordinate { \@@_env: - col - 2 }
1358   { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
1359   \str_if_empty:NF \l_@@_name_str
1360   { \pgfnodealias { \l_@@_name_str - col - 2 } { \@@_env: - col - 2 } }
1361   \endpgfpicture
1362 }

```

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.

```

1363 \int_gset:Nn \g_tmpa_int 1
1364 \bool_if:NTF \g_@@_last_col_found_bool
1365 { \prg_replicate:nn { \g_@@_col_total_int - 2 } }
1366 { \prg_replicate:nn { \g_@@_col_total_int - 1 } }
1367 {
1368   &
1369   \omit

```

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

```

1370 \int_gincr:N \g_tmpa_int
1371 \skip_horizontal:N \g_tmpa_skip
1372 \bool_if:NT \l_@@_code_before_bool
1373 {
1374   \hbox
1375   {
1376     \skip_horizontal:N -0.5\arrayrulewidth
1377     \pgfsys@markposition { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1378     \skip_horizontal:N 0.5\arrayrulewidth
1379   }
1380 }

```

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

```

1381 \pgfpicture
1382 \pgfrememberpicturepositiononpagetrue
1383 \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1384 { \pgfpoint { - 0.5 \arrayrulewidth } \c_zero_dim }
1385 \str_if_empty:NF \l_@@_name_str

```



```

1386         {
1387             \pgfnodealias
1388             { \l_@@_name_str - col - \@@_succ:n \g_tmpa_int }
1389             { \@@_env: - col - \@@_succ:n \g_tmpa_int }
1390         }
1391     \endpgfpicture
1392 }
1393 \bool_if:NT \g_@@_last_col_found_bool
1394 {
1395     \bool_if:NT \l_@@_code_before_bool
1396     {
1397         \pgfsys@markposition { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
1398     }
1399     \skip_horizontal:N 2\col@sep
1400     \pgfpicture
1401     \pgfrememberpicturepositiononpagetrue
1402     \pgfcoordinate { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
1403     \pgfpointorigin
1404     \str_if_empty:NF \l_@@_name_str
1405     {
1406         \pgfnodealias
1407         { \l_@@_name_str - col - \@@_succ:n \g_@@_col_total_int }
1408         { \@@_env: - col - \@@_succ:n \g_@@_col_total_int }
1409     }
1410     \endpgfpicture
1411     \skip_horizontal:N -2\col@sep
1412 }
1413 \cr
1414 }

```

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

```

1415 \tl_const:Nn \c_@@_preamble_first_col_tl
1416 {
1417     >
1418     {
1419         \@@_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.

```

1420     \hbox_set:Nw \l_@@_cell_box
1421     \bool_if:NF \l_@@_NiceTabular_bool \c_math_toggle_token
1422     \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”.

```

1423     \bool_lazy_and:nnT
1424     { \int_compare_p:nNn \c@iRow > 0 }
1425     {
1426         \bool_lazy_or_p:nn
1427         { \int_compare_p:nNn \l_@@_last_row_int < 0 }
1428         { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
1429     }
1430     {
1431         \l_@@_code_for_first_col_tl
1432         \xglobal \colorlet { nicematrix-first-col } { . }
1433     }
1434 }

```

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`.

```

1435     l
1436     <
1437     {
1438         \bool_if:NF \l_@@_NiceTabular_bool \c_math_toggle_token

```

```

1439 \hbox_set_end:
1440 \@@_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.

```

1441 \dim_gset:Nn \g_@@_width_first_col_dim
1442 { \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.

```

1443 \hbox_overlap_left:n
1444 {
1445   \dim_compare:nNnTF { \box_wd:N \l_@@_cell_box } > \c_zero_dim
1446   \@@_node_for_the_cell:
1447   { \box_use_drop:N \l_@@_cell_box }
1448   \skip_horizontal:N \l_@@_left_delim_dim
1449   \skip_horizontal:N \l_@@_left_margin_dim
1450   \skip_horizontal:N \l_@@_extra_left_margin_dim
1451 }
1452 \skip_horizontal:N -2\col@sep
1453 }
1454 }

```

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

```

1455 \tl_const:Nn \c_@@_preamble_last_col_tl
1456 {
1457   >
1458   {

```

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

```

1459 \bool_gset_true:N \g_@@_last_col_found_bool
1460 \int_gincr:N \c@jCol
1461 \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.

```

1462 \hbox_set:Nw \l_@@_cell_box
1463 \bool_if:NF \l_@@_NiceTabular_bool \c_math_toggle_token
1464 \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”.

```

1465 \int_compare:nNnT \c@iRow > 0
1466 {
1467   \bool_lazy_or:nnT
1468   { \int_compare_p:nNn \l_@@_last_row_int < 0 }
1469   { \int_compare_p:nNn \c@iRow < \l_@@_last_row_int }
1470   {
1471     \l_@@_code_for_last_col_tl
1472     \xglobal \colorlet { nicematrix-last-col } { . }
1473   }
1474 }
1475 }
1476 1
1477 <
1478 {
1479   \bool_if:NF \l_@@_NiceTabular_bool \c_math_toggle_token
1480   \hbox_set_end:
1481   \@@_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.

```

1482 \dim_gset:Nn \g_@@_width_last_col_dim
1483 { \dim_max:nn \g_@@_width_last_col_dim { \box_wd:N \l_@@_cell_box } }
1484 \skip_horizontal:N -2\col@sep

```

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

```

1485 \hbox_overlap_right:n
1486 {

```

```

1487         \dim_compare:nNnT { \box_wd:N \l_@@_cell_box } > \c_zero_dim
1488         {
1489             \skip_horizontal:N \l_@@_right_delim_dim
1490             \skip_horizontal:N \l_@@_right_margin_dim
1491             \skip_horizontal:N \l_@@_extra_right_margin_dim
1492             \@@_node_for_the_cell:
1493         }
1494     }
1495 }
1496 }

```

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.

```

1497 \NewDocumentEnvironment { NiceArray } { }
1498 {
1499     \bool_set_true:N \l_@@_NiceArray_bool
1500     \str_if_empty:NT \g_@@_name_env_str
1501     { \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).

```

1502     \NiceArrayWithDelims . .
1503 }
1504 { \endNiceArrayWithDelims }

```

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

```

1505 \NewDocumentEnvironment { pNiceArray } { }
1506 {
1507     \str_if_empty:NT \g_@@_name_env_str
1508     { \str_gset:Nn \g_@@_name_env_str { pNiceArray } }
1509     \@@_test_if_math_mode:
1510     \NiceArrayWithDelims ( )
1511 }
1512 { \endNiceArrayWithDelims }

1513 \NewDocumentEnvironment { bNiceArray } { }
1514 {
1515     \str_if_empty:NT \g_@@_name_env_str
1516     { \str_gset:Nn \g_@@_name_env_str { bNiceArray } }
1517     \@@_test_if_math_mode:
1518     \NiceArrayWithDelims [ ]
1519 }
1520 { \endNiceArrayWithDelims }

1521 \NewDocumentEnvironment { BNiceArray } { }
1522 {
1523     \str_if_empty:NT \g_@@_name_env_str
1524     { \str_gset:Nn \g_@@_name_env_str { BNiceArray } }
1525     \@@_test_if_math_mode:
1526     \NiceArrayWithDelims \{ \}
1527 }
1528 { \endNiceArrayWithDelims }

1529 \NewDocumentEnvironment { vNiceArray } { }
1530 {
1531     \str_if_empty:NT \g_@@_name_env_str
1532     { \str_gset:Nn \g_@@_name_env_str { vNiceArray } }
1533     \@@_test_if_math_mode:
1534     \NiceArrayWithDelims | |
1535 }
1536 { \endNiceArrayWithDelims }

1537 \NewDocumentEnvironment { VNiceArray } { }

```

```

1538 {
1539   \str_if_empty:NT \g_@@_name_env_str
1540   { \str_gset:Nn \g_@@_name_env_str { VNiceArray } }
1541   \@@_test_if_math_mode:
1542   \NiceArrayWithDelims \l \l
1543 }
1544 { \endNiceArrayWithDelims }

```

The environment {NiceMatrix} and its variants

```

1545 \cs_new_protected:Npn \@@_define_env:n #1
1546 {
1547   \NewDocumentEnvironment { #1 NiceMatrix } { ! 0 { } }
1548   {
1549     \str_gset:Nn \g_@@_name_env_str { #1 NiceMatrix }
1550     \tl_set:Nn \l_@@_type_of_col_tl C
1551     \keys_set:nn { NiceMatrix / NiceMatrix } { ##1 }
1552     \exp_args:Nnx \@@_begin_of_NiceMatrix:nn { #1 } \l_@@_type_of_col_tl
1553   }
1554   { \use:c { end #1 NiceArray } }
1555 }

1556 \cs_new_protected:Npn \@@_begin_of_NiceMatrix:nn #1 #2
1557 {
1558   \use:c { #1 NiceArray }
1559   {
1560     *
1561     {
1562       \int_compare:nNnTF \l_@@_last_col_int < 0
1563       \c@MaxMatrixCols
1564       { \@@_pred:n \l_@@_last_col_int }
1565     }
1566     #2
1567   }
1568 }

1569 \@@_define_env:n { }
1570 \@@_define_env:n p
1571 \@@_define_env:n b
1572 \@@_define_env:n B
1573 \@@_define_env:n v
1574 \@@_define_env:n V

```

The environment {NiceTabular}

```

1575 \NewDocumentEnvironment { NiceTabular } { 0 { } m ! 0 { } }
1576 {
1577   \keys_set:nn { NiceMatrix / NiceTabular } { #1 , #3 }
1578   \bool_set_true:N \l_@@_NiceTabular_bool
1579   \NiceArray { #2 }
1580 }
1581 { \endNiceArray }

```

After the construction of the array

```

1582 \cs_new_protected:Npn \@@_after_array:
1583 {
1584   \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.

```
1585 \bool_if:NT \g_@@_last_col_found_bool
1586 { \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.

```
1587 \bool_if:NT \l_@@_last_col_without_value_bool
1588 {
1589   \dim_set_eq:NN \l_@@_last_col_int \g_@@_col_total_int
1590   \iow_shipout:Nn \@mainaux \ExplSyntaxOn
1591   \iow_shipout:Nx \@mainaux
1592   {
1593     \cs_gset:cpn { @@_last_col_ \int_use:N \g_@@_env_int }
1594     { \int_use:N \g_@@_col_total_int }
1595   }
1596   \str_if_empty:NF \l_@@_name_str
1597   {
1598     \iow_shipout:Nx \@mainaux
1599     {
1600       \cs_gset:cpn { @@_last_col_ \l_@@_name_str }
1601       { \int_use:N \g_@@_col_total_int }
1602     }
1603   }
1604   \iow_shipout:Nn \@mainaux \ExplSyntaxOff
1605 }
```

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.

```
1606 \bool_if:NT \l_@@_last_row_without_value_bool
1607 {
1608   \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.

```
1609 \bool_if:NF \l_@@_light_syntax_bool
1610 {
1611   \iow_shipout:Nn \@mainaux \ExplSyntaxOn
1612   \iow_shipout:Nx \@mainaux
1613   {
1614     \cs_gset:cpn { @@_last_row_ \int_use:N \g_@@_env_int }
1615     { \int_use:N \g_@@_row_total_int }
1616   }
```

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.

```
1617 \str_if_empty:NF \l_@@_name_str
1618 {
1619   \iow_shipout:Nx \@mainaux
1620   {
1621     \cs_gset:cpn { @@_last_row_ \l_@@_name_str }
1622     { \int_use:N \g_@@_row_total_int }
1623   }
1624 }
1625 \iow_shipout:Nn \@mainaux \ExplSyntaxOff
1626 }
1627 }
```

If the key code-before is used, we have to write on the aux file the actual size of the array.

```
1628 \bool_if:NT \l_@@_code_before_bool
1629 {
1630   \iow_now:Nn \@mainaux \ExplSyntaxOn
1631   \iow_now:Nx \@mainaux
1632   { \seq_clear_new:c { @@_size _ \int_use:N \g_@@_env_int _ seq } }
1633   \iow_now:Nx \@mainaux
1634   {
```

```

1635 \seq_gset_from_clist:cn { @@_size _ \int_use:N \g_@@_env_int _ seq }
1636 {
1637     \int_use:N \l_@@_first_row_int ,
1638     \int_use:N \g_@@_row_total_int ,
1639     \int_use:N \l_@@_first_col_int ,

```

If the user has used a key `last-row` in an environment with preamble (like `{pNiceArray}`) and that that last row has not been found, we have to increment the value because it will be decreased when used in the code-before.

```

1640     \bool_lazy_and:nnTF
1641     { \int_compare_p:nNn \l_@@_last_col_int > { -2 } }
1642     { \bool_not_p:n \g_@@_last_col_found_bool }
1643     \@@_succ:n
1644     \int_use:N
1645     \g_@@_col_total_int
1646 }
1647 }
1648 \iow_now:Nn \@mainaux \ExplSyntaxOff
1649 }

```

By default, the diagonal lines will be parallelized³⁴. 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.

```

1650 \bool_if:NT \l_@@_parallelize_diags_bool
1651 {
1652     \int_gzero_new:N \g_@@_ddots_int
1653     \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.

```

1654     \dim_gzero_new:N \g_@@_delta_x_one_dim
1655     \dim_gzero_new:N \g_@@_delta_y_one_dim
1656     \dim_gzero_new:N \g_@@_delta_x_two_dim
1657     \dim_gzero_new:N \g_@@_delta_y_two_dim
1658 }
1659 \bool_if:nTF \l_@@_medium_nodes_bool
1660 {
1661     \bool_if:NTF \l_@@_large_nodes_bool
1662     \@@_create_medium_and_large_nodes:
1663     \@@_create_medium_nodes:
1664 }
1665 { \bool_if:NT \l_@@_large_nodes_bool \@@_create_large_nodes: }
1666 \int_zero_new:N \l_@@_initial_i_int
1667 \int_zero_new:N \l_@@_initial_j_int
1668 \int_zero_new:N \l_@@_final_i_int
1669 \int_zero_new:N \l_@@_final_j_int
1670 \bool_set_false:N \l_@@_initial_open_bool
1671 \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.

```

1672 \bool_if:NT \l_@@_small_bool
1673 {
1674     \dim_set:Nn \l_@@_radius_dim { 0.37 pt }
1675     \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.

```

1676     \dim_set:Nn \l_@@_xdots_shorten_dim { 0.6 \l_@@_xdots_shorten_dim }
1677 }

```

³⁴It's possible to use the option `parallelize-diags` to disable this parallelization.

Now, we really draw the dotted lines.

```

1678 \@@_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.
1679 \bool_if:NT \l_@@_vlines_bool \@@_draw_vlines:
1680 \g_@@_internal_code_after_tl
1681 \tl_gclear:N \g_@@_internal_code_after_tl
1682 \bool_if:NT \c_@@_tikz_loaded_bool
1683 {
1684   \tikzset
1685   {
1686     every~picture / .style =
1687     {
1688       overlay ,
1689       remember~picture ,
1690       name~prefix = \@@_env: -
1691     }
1692   }
1693 }
1694 \cs_set_eq:NN \line \@@_line

```

When `light-syntax` is used, we insert systematically a `\CodeAfter` in the flow. Thus, it's possible to have two instructions `\CodeAfter` and the second one is eventually present in `\g_@@_code_after_tl`. That's why we set `\Code-after` to be *no-op* now.

```

1695 \cs_set_eq:NN \CodeAfter \prg_do_nothing:

```

And here's the code-after:

```

1696 \g_@@_code_after_tl
1697 \tl_gclear:N \g_@@_code_after_tl
1698 \group_end:
1699 \str_gclear:N \g_@@_name_env_str
1700 \@@_restore_iRow_jCol:
1701 }

```

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).

```

1702 \AtBeginDocument
1703 {
1704   \cs_new_protected:Npx \@@_draw_dotted_lines:
1705   {
1706     \c_@@_pgfortikzpicture_tl
1707     \@@_draw_dotted_lines_i:
1708     \c_@@_endpgfortikzpicture_tl
1709   }
1710 }

```

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

```

1711 \cs_new_protected:Npn \@@_draw_dotted_lines_i:
1712 {
1713   \pgfrememberpicturepositiononpagetrue
1714   \pgf@relevantforpicturesizefalse
1715   \g_@@_HVdotsfor_lines_tl
1716   \g_@@_Vdots_lines_tl
1717   \g_@@_Ddots_lines_tl
1718   \g_@@_Iddots_lines_tl
1719   \g_@@_Cdots_lines_tl
1720   \g_@@_Ldots_lines_tl
1721 }
1722 \cs_new_protected:Npn \@@_restore_iRow_jCol:
1723 {
1724   \cs_if_exist:NT \theiRow { \int_gset_eq:NN \c@iRow \l_@@_old_iRow_int }

```

```

1725     \cs_if_exist:NT \thejCol { \int_gset_eq:NN \c@jCol \l_@@_old_jCol_int }
1726 }

```

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.

```

1727 \cs_new_protected:Npn \@@_find_extremities_of_line:nnnn #1 #2 #3 #4
1728 {

```

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

```

1729     \cs_set:cpn { @@ _ dotted _ #1 - #2 } { }

```

Initialization of variables.

```

1730     \int_set:Nn \l_@@_initial_i_int { #1 }
1731     \int_set:Nn \l_@@_initial_j_int { #2 }
1732     \int_set:Nn \l_@@_final_i_int { #1 }
1733     \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.

```

1734     \bool_set_false:N \l_@@_stop_loop_bool
1735     \bool_do_until:Nn \l_@@_stop_loop_bool
1736     {
1737         \int_add:Nn \l_@@_final_i_int { #3 }
1738         \int_add:Nn \l_@@_final_j_int { #4 }

```

We test if we are still in the matrix.

```

1739     \bool_set_false:N \l_@@_final_open_bool
1740     \int_compare:nNnTF \l_@@_final_i_int > \c@iRow
1741     {
1742         \int_compare:nNnTF { #3 } = 1
1743         { \bool_set_true:N \l_@@_final_open_bool }
1744         {
1745             \int_compare:nNnT \l_@@_final_j_int > \c@jCol
1746             { \bool_set_true:N \l_@@_final_open_bool }
1747         }
1748     }
1749     {
1750         \int_compare:nNnTF \l_@@_final_j_int < 1

```



```

1751     {
1752       \int_compare:nNtT { #4 } = { -1 }
1753       { \bool_set_true:N \l_@@_final_open_bool }
1754     }
1755     {
1756       \int_compare:nNtT \l_@@_final_j_int > \c@jCol
1757       {
1758         \int_compare:nNtT { #4 } = 1
1759         { \bool_set_true:N \l_@@_final_open_bool }
1760       }
1761     }
1762   }
1763   \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.

```

1764   {
We do a step backwards.
1765     \int_sub:Nn \l_@@_final_i_int { #3 }
1766     \int_sub:Nn \l_@@_final_j_int { #4 }
1767     \bool_set_true:N \l_@@_stop_loop_bool
1768   }

```

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`.

```

1769   {
1770     \cs_if_exist:cTF
1771     {
1772       @@ _ dotted _
1773       \int_use:N \l_@@_final_i_int -
1774       \int_use:N \l_@@_final_j_int
1775     }
1776     {
1777       \int_sub:Nn \l_@@_final_i_int { #3 }
1778       \int_sub:Nn \l_@@_final_j_int { #4 }
1779       \bool_set_true:N \l_@@_final_open_bool
1780       \bool_set_true:N \l_@@_stop_loop_bool
1781     }
1782     {
1783       \cs_if_exist:cTF
1784       {
1785         pgf @ sh @ ns @ \@@_env:
1786         - \int_use:N \l_@@_final_i_int
1787         - \int_use:N \l_@@_final_j_int
1788       }
1789       { \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.

```

1790     {
1791       \cs_set:cpn
1792       {
1793         @@ _ dotted _
1794         \int_use:N \l_@@_final_i_int -
1795         \int_use:N \l_@@_final_j_int
1796       }
1797       { }
1798     }
1799   }
1800 }
1801 }

```

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

```

1802 \bool_set_false:N \l_@@_stop_loop_bool
1803 \bool_do_until:Nn \l_@@_stop_loop_bool
1804 {
1805   \int_sub:Nn \l_@@_initial_i_int { #3 }
1806   \int_sub:Nn \l_@@_initial_j_int { #4 }
1807   \bool_set_false:N \l_@@_initial_open_bool
1808   \int_compare:nNnTF \l_@@_initial_i_int < 1
1809   {
1810     \int_compare:nNnTF { #3 } = 1
1811     { \bool_set_true:N \l_@@_initial_open_bool }
1812     {
1813       \int_compare:nNnT \l_@@_initial_j_int = 0
1814       { \bool_set_true:N \l_@@_initial_open_bool }
1815     }
1816   }
1817   {
1818     \int_compare:nNnTF \l_@@_initial_j_int < 1
1819     {
1820       \int_compare:nNnT { #4 } = 1
1821       { \bool_set_true:N \l_@@_initial_open_bool }
1822     }
1823     {
1824       \int_compare:nNnT \l_@@_initial_j_int > \c@jCol
1825       {
1826         \int_compare:nNnT { #4 } = { -1 }
1827         { \bool_set_true:N \l_@@_initial_open_bool }
1828       }
1829     }
1830   }
1831   \bool_if:NTF \l_@@_initial_open_bool
1832   {
1833     \int_add:Nn \l_@@_initial_i_int { #3 }
1834     \int_add:Nn \l_@@_initial_j_int { #4 }
1835     \bool_set_true:N \l_@@_stop_loop_bool
1836   }
1837   {
1838     \cs_if_exist:cTF
1839     {
1840       @@ _ dotted _
1841       \int_use:N \l_@@_initial_i_int -
1842       \int_use:N \l_@@_initial_j_int
1843     }
1844     {
1845       \int_add:Nn \l_@@_initial_i_int { #3 }
1846       \int_add:Nn \l_@@_initial_j_int { #4 }
1847       \bool_set_true:N \l_@@_initial_open_bool
1848       \bool_set_true:N \l_@@_stop_loop_bool
1849     }
1850     {
1851       \cs_if_exist:cTF
1852       {
1853         pgf @ sh @ ns @ \@@_env:
1854         - \int_use:N \l_@@_initial_i_int
1855         - \int_use:N \l_@@_initial_j_int
1856       }
1857       { \bool_set_true:N \l_@@_stop_loop_bool }
1858       {
1859         \cs_set:cpn
1860         {
1861           @@ _ dotted _
1862           \int_use:N \l_@@_initial_i_int -

```

```

1863             \int_use:N \l_@@_initial_j_int
1864         }
1865     { }
1866 }
1867 }
1868 }
1869 }
1870 }

1871 \cs_new_protected:Npn \@@_set_initial_coords:
1872 {
1873     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
1874     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
1875 }
1876 \cs_new_protected:Npn \@@_set_final_coords:
1877 {
1878     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
1879     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
1880 }
1881 \cs_new_protected:Npn \@@_set_initial_coords_from_anchor:n #1
1882 {
1883     \pgfpointanchor
1884     {
1885         \@@_env:
1886         - \int_use:N \l_@@_initial_i_int
1887         - \int_use:N \l_@@_initial_j_int
1888     }
1889     { #1 }
1890     \@@_set_initial_coords:
1891 }
1892 \cs_new_protected:Npn \@@_set_final_coords_from_anchor:n #1
1893 {
1894     \pgfpointanchor
1895     {
1896         \@@_env:
1897         - \int_use:N \l_@@_final_i_int
1898         - \int_use:N \l_@@_final_j_int
1899     }
1900     { #1 }
1901     \@@_set_final_coords:
1902 }

```

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.

```

1903 \cs_new_protected:Npn \@@_draw_Ldots:nnn #1 #2 #3
1904 {
1905     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1906     {
1907         \@@_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.

```

1908     \group_begin:
1909     \int_compare:nNnTF { #1 } = 0
1910     { \color { nicematrix-first-row } }
1911     {

```

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.

```

1912         \int_compare:nNnT { #1 } = \l_@@_last_row_int
1913         { \color { nicematrix-last-row } }
1914     }

```

```

1915         \keys_set:nn { NiceMatrix / xdots } { #3 }
1916         \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
1917         \@@_actually_draw_Ldots:
1918     \group_end:
1919 }
1920 }

```

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`.

```

1921 \cs_new_protected:Npn \@@_actually_draw_Ldots:
1922 {
1923     \bool_if:NTF \l_@@_initial_open_bool
1924     {
1925         \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
1926         \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
1927         \dim_add:Nn \l_@@_x_initial_dim
1928             { \bool_if:NTF \l_@@_NiceTabular_bool \tabcolsep \arraycolsep }
1929         \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int - base }
1930         \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
1931     }
1932     { \@@_set_initial_coords_from_anchor:n { base-east } }
1933     \bool_if:NTF \l_@@_final_open_bool
1934     {
1935         \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
1936         \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
1937         \dim_sub:Nn \l_@@_x_final_dim
1938             { \bool_if:NTF \l_@@_NiceTabular_bool \tabcolsep \arraycolsep }
1939         \@@_qpoint:n { row - \int_use:N \l_@@_final_i_int - base }
1940         \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
1941     }
1942     { \@@_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. Of course, maybe we should not do that when the option `line-style` is used (?).

```

1943     \dim_add:Nn \l_@@_y_initial_dim \l_@@_radius_dim
1944     \dim_add:Nn \l_@@_y_final_dim \l_@@_radius_dim
1945     \@@_draw_line:
1946 }

```

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.

```

1947 \cs_new_protected:Npn \@@_draw_Cdots:nnn #1 #2 #3
1948 {
1949     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1950     {
1951         \@@_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.

```

1952     \group_begin:
1953     \int_compare:nNnTF { #1 } = 0
1954     { \color { nicematrix-first-row } }
1955     {

```

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.

```

1956         \int_compare:nNnT { #1 } = \l_@@_last_row_int
1957         { \color { nicematrix-last-row } }
1958     }
1959     \keys_set:nn { NiceMatrix / xdots } { #3 }
1960     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
1961     \@@_actually_draw_Cdots:
1962 \group_end:
1963 }
1964 }

```

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`.

```

1965 \cs_new_protected:Npn \@@_actually_draw_Cdots:
1966 {
1967     \bool_if:NTF \l_@@_initial_open_bool
1968     {
1969         \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
1970         \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
1971         \dim_add:Nn \l_@@_x_initial_dim
1972         { \bool_if:NTF \l_@@_NiceTabular_bool \tabcolsep \arraycolsep }
1973     }
1974     { \@@_set_initial_coords_from_anchor:n { mid~east } }
1975     \bool_if:NTF \l_@@_final_open_bool
1976     {
1977         \@@_qpoint:n { col - \@@_succ:n \l_@@_final_j_int }
1978         \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
1979         \dim_sub:Nn \l_@@_x_final_dim
1980         { \bool_if:NTF \l_@@_NiceTabular_bool \tabcolsep \arraycolsep }
1981     }
1982     { \@@_set_final_coords_from_anchor:n { mid~west } }
1983     \bool_lazy_and:nnTF
1984     \l_@@_initial_open_bool
1985     \l_@@_final_open_bool
1986     {
1987         \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
1988         \dim_set_eq:NN \l_tmpa_dim \pgf@y
1989         \@@_qpoint:n { row - \@@_succ:n \l_@@_initial_i_int }
1990         \dim_set:Nn \l_@@_y_initial_dim { ( \l_tmpa_dim + \pgf@y ) / 2 }
1991         \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim
1992     }
1993     {
1994         \bool_if:NT \l_@@_initial_open_bool
1995         { \dim_set_eq:NN \l_@@_y_initial_dim \l_@@_y_final_dim }
1996         \bool_if:NT \l_@@_final_open_bool
1997         { \dim_set_eq:NN \l_@@_y_final_dim \l_@@_y_initial_dim }
1998     }
1999     \@@_draw_line:
2000 }

```

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.

```

2001 \cs_new_protected:Npn \l_@@_draw_Vdots:nnn #1 #2 #3
2002 {
2003   \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2004   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2005   {
2006     \@@_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.

```

2007     \group_begin:
2008     \int_compare:nNnTF { #2 } = 0
2009     { \color { nicematrix-first-col } }
2010     {
2011       \int_compare:nNnT { #2 } = \l_@@_last_col_int
2012       { \color { nicematrix-last-col } }
2013     }
2014     \keys_set:nn { NiceMatrix / xdots } { #3 }
2015     \@@_actually_draw_Vdots:
2016   \group_end:
2017 }
2018 }

```

The command `\@@_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`.

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

```

2019 \cs_new_protected:Npn \l_@@_actually_draw_Vdots:
2020 {

```

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

```

2021   \bool_set_false:N \l_tmpa_bool
2022   \bool_lazy_or:nnF \l_@@_initial_open_bool \l_@@_final_open_bool
2023   {
2024     \@@_set_initial_coords_from_anchor:n { south-west }
2025     \@@_set_final_coords_from_anchor:n { north-west }
2026     \bool_set:Nn \l_tmpa_bool
2027     { \dim_compare_p:nNn \l_@@_x_initial_dim = \l_@@_x_final_dim }
2028   }

```

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

```

2029   \bool_if:NTF \l_@@_initial_open_bool
2030   {
2031     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2032     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2033   }
2034   { \@@_set_initial_coords_from_anchor:n { south } }
2035   \bool_if:NTF \l_@@_final_open_bool
2036   {
2037     \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }
2038     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2039   }
2040   { \@@_set_final_coords_from_anchor:n { north } }

```

```

2041 \bool_if:NTF \l_@@_initial_open_bool
2042 {
2043   \bool_if:NTF \l_@@_final_open_bool
2044   {
2045     \@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2046     \dim_set_eq:NN \l_tmpa_dim \pgf@x
2047     \@@_qpoint:n { col - \@@_succ:n \l_@@_initial_j_int }
2048     \dim_set:Nn \l_@@_x_initial_dim { ( \pgf@x + \l_tmpa_dim ) / 2 }
2049     \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim

```

We may think that the final user won't use a "last column" which contains only a command `\Vdots`. However, if the `\Vdots` is in fact used to draw, not a dotted line, but an arrow (to indicate the number of rows of the matrix), it may be really encountered.

```

2050     \int_compare:nNnT \l_@@_last_col_int > { -2 }
2051     {
2052       \int_compare:nNnT \l_@@_initial_j_int = \g_@@_col_total_int
2053       {
2054         \dim_set_eq:NN \l_tmpa_dim \l_@@_right_margin_dim
2055         \dim_add:Nn \l_tmpa_dim \l_@@_extra_right_margin_dim
2056         \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
2057         \dim_add:Nn \l_@@_x_final_dim \l_tmpa_dim
2058       }
2059     }
2060   }
2061   { \dim_set_eq:NN \l_@@_x_initial_dim \l_@@_x_final_dim }
2062 }
2063 {
2064   \bool_if:NTF \l_@@_final_open_bool
2065   { \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim }
2066   {

```

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.

```

2067     \dim_compare:nNnF \l_@@_x_initial_dim = \l_@@_x_final_dim
2068     {
2069       \dim_set:Nn \l_@@_x_initial_dim
2070       {
2071         \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
2072         \l_@@_x_initial_dim \l_@@_x_final_dim
2073       }
2074       \dim_set_eq:NN \l_@@_x_final_dim \l_@@_x_initial_dim
2075     }
2076   }
2077 }
2078 \@@_draw_line:
2079 }

```

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.

```

2080 \cs_new_protected:Npn \@@_draw_Ddots:nnn #1 #2 #3
2081 {
2082   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2083   {
2084     \@@_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.

```

2085   \group_begin:
2086   \keys_set:nn { NiceMatrix / xdots } { #3 }
2087   \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }

```

```

2088         \l_@@_actually_draw_Ddots:
2089     \group_end:
2090 }
2091 }

```

The command `\l_@@_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.`

```

2092 \cs_new_protected:Npn \l_@@_actually_draw_Ddots:
2093 {
2094     \bool_if:NTF \l_@@_initial_open_bool
2095     {
2096         \l_@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2097         \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2098         \l_@@_qpoint:n { col - \int_use:N \l_@@_initial_j_int }
2099         \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2100     }
2101     { \l_@@_set_initial_coords_from_anchor:n { south-east } }
2102     \bool_if:NTF \l_@@_final_open_bool
2103     {
2104         \l_@@_qpoint:n { row - \l_@@_succ:n \l_@@_final_i_int }
2105         \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2106         \l_@@_qpoint:n { col - \l_@@_succ:n \l_@@_final_j_int }
2107         \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2108     }
2109     { \l_@@_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.

```

2110     \bool_if:NT \l_@@_parallelize_diags_bool
2111     {
2112         \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).

```

2113         \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.

```

2114     {
2115         \dim_gset:Nn \g_@@_delta_x_one_dim
2116         { \l_@@_x_final_dim - \l_@@_x_initial_dim }
2117         \dim_gset:Nn \g_@@_delta_y_one_dim
2118         { \l_@@_y_final_dim - \l_@@_y_initial_dim }
2119     }

```

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`.

```

2120     {
2121         \dim_set:Nn \l_@@_y_final_dim
2122         {
2123             \l_@@_y_initial_dim +
2124             ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2125             \dim_ratio:nn \g_@@_delta_y_one_dim \g_@@_delta_x_one_dim
2126         }

```



```

2127     }
2128   }
2129   \@@_draw_line:
2130 }

```

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.

```

2131 \cs_new_protected:Npn \@@_draw_Iddots:nnn #1 #2 #3
2132 {
2133   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
2134   {
2135     \@@_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.

```

2136     \group_begin:
2137     \keys_set:nn { NiceMatrix / xdots } { #3 }
2138     \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2139     \@@_actually_draw_Iddots:
2140   \group_end:
2141 }
2142 }

```

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`.

```

2143 \cs_new_protected:Npn \@@_actually_draw_Iddots:
2144 {
2145   \bool_if:NTF \l_@@_initial_open_bool
2146   {
2147     \@@_qpoint:n { row - \int_use:N \l_@@_initial_i_int }
2148     \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2149     \@@_qpoint:n { col - \@@_succ:n \l_@@_initial_j_int }
2150     \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2151   }
2152   { \@@_set_initial_coords_from_anchor:n { south-west } }
2153   \bool_if:NTF \l_@@_final_open_bool
2154   {
2155     \@@_qpoint:n { row - \@@_succ:n \l_@@_final_i_int }
2156     \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2157     \@@_qpoint:n { col - \int_use:N \l_@@_final_j_int }
2158     \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2159   }
2160   { \@@_set_final_coords_from_anchor:n { north-east } }
2161   \bool_if:NT \l_@@_parallelize_diags_bool
2162   {
2163     \int_gincr:N \g_@@_iddots_int
2164     \int_compare:nNnTF \g_@@_iddots_int = 1
2165     {
2166       \dim_gset:Nn \g_@@_delta_x_two_dim
2167       { \l_@@_x_final_dim - \l_@@_x_initial_dim }
2168       \dim_gset:Nn \g_@@_delta_y_two_dim

```

```

2169         { \l_@@_y_final_dim - \l_@@_y_initial_dim }
2170     }
2171     {
2172         \dim_set:Nn \l_@@_y_final_dim
2173         {
2174             \l_@@_y_initial_dim +
2175             ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2176             \dim_ratio:nn \g_@@_delta_y_two_dim \g_@@_delta_x_two_dim
2177         }
2178     }
2179 }
2180 \@@_draw_line:
2181 }

```

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).

```

2182 \NewExpandableDocumentCommand \NiceMatrixLastEnv { }
2183 { \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`

```

2184 \cs_new_protected:Npn \@@_draw_line:
2185 {
2186     \pgfrememberpicturepositiononpagetrue
2187     \pgf@relevantforpicturesizefalse
2188     \tl_if_eq:NNTF \l_@@_xdots_line_style_tl \c_@@_standard_tl
2189         \@@_draw_standard_dotted_line:
2190         \@@_draw_non_standard_dotted_line:
2191 }

```

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.

```

2192 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:
2193 {
2194     \begin { scope }
2195     \exp_args:No \@@_draw_non_standard_dotted_line:n
2196         { \l_@@_xdots_line_style_tl , \l_@@_xdots_color_tl }
2197 }

```

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.

```

2198 \cs_new_protected:Npn \@@_draw_non_standard_dotted_line:n #1
2199 {
2200     \draw

```

```

2201 [
2202   #1 ,
2203   shorten~> = \l_@@_xdots_shorten_dim ,
2204   shorten~< = \l_@@_xdots_shorten_dim ,
2205 ]
2206 ( \l_@@_x_initial_dim , \l_@@_y_initial_dim )
2207 -- node [ sloped , above ]
2208   { \c_math_toggle_token \scriptstyle \l_@@_xdots_up_tl \c_math_toggle_token }
2209   node [ sloped , below ]
2210     {
2211       \c_math_toggle_token
2212       \scriptstyle \l_@@_xdots_down_tl
2213       \c_math_toggle_token
2214     }
2215 ( \l_@@_x_final_dim , \l_@@_y_final_dim ) ;
2216 \end { scope }
2217 }

```

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

```

2218 \cs_new_protected:Npn \@@_draw_standard_dotted_line:
2219 {

```

First, we put the labels.

```

2220   \bool_lazy_and:nnF
2221   { \tl_if_empty_p:N \l_@@_xdots_up_tl }
2222   { \tl_if_empty_p:N \l_@@_xdots_down_tl }
2223   {
2224     \pgfscope
2225     \pgftransformshift
2226     {
2227       \pgfpointlineattime { 0.5 }
2228       { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
2229       { \pgfpoint \l_@@_x_final_dim \l_@@_y_final_dim }
2230     }
2231     \pgftransformrotate
2232     {
2233       \fp_eval:n
2234       {
2235         atand
2236         (
2237           \l_@@_y_final_dim - \l_@@_y_initial_dim ,
2238           \l_@@_x_final_dim - \l_@@_x_initial_dim
2239         )
2240       }
2241     }
2242     \pgfnode
2243     { rectangle }
2244     { south }
2245     {
2246       \c_math_toggle_token
2247       \scriptstyle \l_@@_xdots_up_tl
2248       \c_math_toggle_token
2249     }
2250     { }
2251     { \pgfusepath { } }
2252     \pgfnode
2253     { rectangle }
2254     { north }
2255     {
2256       \c_math_toggle_token
2257       \scriptstyle \l_@@_xdots_down_tl
2258       \c_math_toggle_token

```

```

2259     }
2260     { }
2261     { \pgfusepath { } }
2262   \endpgfscope
2263 }
2264 \pgfrememberpicturepositiononpagetrue
2265 \pgf@relevantforpicturesizefalse
2266 \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.

```

2267   \dim_zero_new:N \l_@@_l_dim
2268   \dim_set:Nn \l_@@_l_dim
2269   {
2270     \fp_to_dim:n
2271     {
2272       sqrt
2273       (
2274         ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) ^ 2
2275         +
2276         ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) ^ 2
2277       )
2278     }
2279   }

```

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.

```

2280   \bool_lazy_or:nnF
2281   { \dim_compare_p:nNn { \dim_abs:n \l_@@_l_dim } > \c_@@_max_l_dim }
2282   { \dim_compare_p:nNn \l_@@_l_dim = \c_zero_dim }
2283   \@@_draw_standard_dotted_line_i:
2284   \group_end:
2285 }
2286 \dim_const:Nn \c_@@_max_l_dim { 50 cm }
2287 \cs_new_protected:Npn \@@_draw_standard_dotted_line_i:
2288 {

```

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

```

2289   \bool_if:NTF \l_@@_initial_open_bool
2290   {
2291     \bool_if:NTF \l_@@_final_open_bool
2292     {
2293       \int_set:Nn \l_tmpa_int
2294       { \dim_ratio:nn \l_@@_l_dim \l_@@_inter_dots_dim }
2295     }
2296     {
2297       \int_set:Nn \l_tmpa_int
2298       {
2299         \dim_ratio:nn
2300         { \l_@@_l_dim - \l_@@_xdots_shorten_dim }
2301         \l_@@_inter_dots_dim
2302       }
2303     }
2304   }
2305   {
2306     \bool_if:NTF \l_@@_final_open_bool
2307     {
2308       \int_set:Nn \l_tmpa_int
2309       {
2310         \dim_ratio:nn
2311         { \l_@@_l_dim - \l_@@_xdots_shorten_dim }

```

```

2312         \l_@@_inter_dots_dim
2313     }
2314 }
2315 {
2316     \int_set:Nn \l_tmpa_int
2317     {
2318         \dim_ratio:nn
2319         { \l_@@_l_dim - 2 \l_@@_xdots_shorten_dim }
2320         \l_@@_inter_dots_dim
2321     }
2322 }
2323 }

```

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

```

2324 \dim_set:Nn \l_tmpa_dim
2325 {
2326     ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2327     \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
2328 }
2329 \dim_set:Nn \l_tmpb_dim
2330 {
2331     ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
2332     \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
2333 }

```

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`.

```

2334 \int_set:Nn \l_tmpb_int
2335 {
2336     \bool_if:NTF \l_@@_initial_open_bool
2337     { \bool_if:NTF \l_@@_final_open_bool 1 0 }
2338     { \bool_if:NTF \l_@@_final_open_bool 2 1 }
2339 }

```

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.

```

2340 \dim_gadd:Nn \l_@@_x_initial_dim
2341 {
2342     ( \l_@@_x_final_dim - \l_@@_x_initial_dim ) *
2343     \dim_ratio:nn
2344     { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
2345     { 2 \l_@@_l_dim }
2346     * \l_tmpb_int
2347 }
2348 \dim_gadd:Nn \l_@@_y_initial_dim
2349 {
2350     ( \l_@@_y_final_dim - \l_@@_y_initial_dim ) *
2351     \dim_ratio:nn
2352     { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
2353     { 2 \l_@@_l_dim }
2354     * \l_tmpb_int
2355 }
2356 \pgf@relevantforpicturesizefalse
2357 \int_step_inline:nnn 0 \l_tmpa_int
2358 {
2359     \pgfpathcircle
2360     { \pgfpoint \l_@@_x_initial_dim \l_@@_y_initial_dim }
2361     { \l_@@_radius_dim }
2362     \dim_add:Nn \l_@@_x_initial_dim \l_tmpa_dim
2363     \dim_add:Nn \l_@@_y_initial_dim \l_tmpb_dim
2364 }

```

```

2365 \pgfusepathqfill
2366 }

```

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.

```

2367 \AtBeginDocument
2368 {
2369   \tl_set:Nn \l_@@_argspec_tl { s 0 { } E { _ ^ } { { } { } } }
2370   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
2371   \exp_args:NNV \NewDocumentCommand \@@_Ldots \l_@@_argspec_tl
2372     {
2373       \bool_if:nTF { #1 }
2374         { \@@_error:n { starred-commands } }
2375         {
2376           \int_compare:nNnTF \c@jCol = 0
2377             { \@@_error:nn { in-first-col } \Ldots }
2378             {
2379               \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
2380                 { \@@_error:nn { in-last-col } \Ldots }
2381                 {
2382                   \@@_instruction_of_type:nn { Ldots }
2383                   { #2 , down = #3 , up = #4 }
2384                 }
2385             }
2386         }
2387       \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_ldots }
2388       \bool_gset_true:N \g_@@_empty_cell_bool
2389     }

2390   \exp_args:NNV \NewDocumentCommand \@@_Cdots \l_@@_argspec_tl
2391     {
2392       \bool_if:nTF { #1 }
2393         { \@@_error:n { starred-commands } }
2394         {
2395           \int_compare:nNnTF \c@jCol = 0
2396             { \@@_error:nn { in-first-col } \Cdots }
2397             {
2398               \int_compare:nNnTF \c@jCol = \l_@@_last_col_int
2399                 { \@@_error:nn { in-last-col } \Cdots }
2400                 {
2401                   \@@_instruction_of_type:nn { Cdots }
2402                   { #2 , down = #3 , up = #4 }
2403                 }
2404             }
2405         }
2406       \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_cdots }
2407       \bool_gset_true:N \g_@@_empty_cell_bool
2408     }

```

```

2409 \exp_args:NNV \NewDocumentCommand \@@_Vdots \l_@@_argspec_tl
2410 {
2411   \bool_if:nTF { #1 }
2412   { \@@_error:n { starred-commands } }
2413   \int_compare:nNnTF \c@iRow = 0
2414   { \@@_error:nn { in-first-row } \Vdots }
2415   {
2416     \int_compare:nNnTF \c@iRow = \l_@@_last_row_int
2417     { \@@_error:nn { in-last-row } \Vdots }
2418     {
2419       \@@_instruction_of_type:nn { Vdots }
2420       { #2 , down = #3 , up = #4 }
2421     }
2422   }
2423   \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_vdots }
2424   \bool_gset_true:N \g_@@_empty_cell_bool
2425 }

2426 \exp_args:NNV \NewDocumentCommand \@@_Ddots \l_@@_argspec_tl
2427 {
2428   \bool_if:nTF { #1 }
2429   { \@@_error:n { starred-commands } }
2430   {
2431     \int_case:nnF \c@iRow
2432     {
2433       0 { \@@_error:nn { in-first-row } \Ddots }
2434       \l_@@_last_row_int { \@@_error:nn { in-last-row } \Ddots }
2435     }
2436     {
2437       \int_case:nnF \c@jCol
2438       {
2439         0 { \@@_error:nn { in-first-col } \Ddots }
2440         \l_@@_last_col_int { \@@_error:nn { in-last-col } \Ddots }
2441       }
2442       {
2443         \@@_instruction_of_type:nn { Ddots }
2444         { #2 , down = #3 , up = #4 }
2445       }
2446     }
2447   }
2448 }
2449 \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_ddots }
2450 \bool_gset_true:N \g_@@_empty_cell_bool
2451 }

2452 \exp_args:NNV \NewDocumentCommand \@@_Iddots \l_@@_argspec_tl
2453 {
2454   \bool_if:nTF { #1 }
2455   { \@@_error:n { starred-commands } }
2456   {
2457     \int_case:nnF \c@iRow
2458     {
2459       0 { \@@_error:nn { in-first-row } \Iddots }
2460       \l_@@_last_row_int { \@@_error:nn { in-last-row } \Iddots }
2461     }
2462     {
2463       \int_case:nnF \c@jCol
2464       {
2465         0 { \@@_error:nn { in-first-col } \Iddots }
2466         \l_@@_last_col_int { \@@_error:nn { in-last-col } \Iddots }
2467       }
2468       {

```

```

2469         \@@_instruction_of_type:nn { Iddots }
2470         { #2 , down = #3 , up = #4 }
2471     }
2472
2473     }
2474 }
2475 \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_old_iddots }
2476 \bool_gset_true:N \g_@@_empty_cell_bool
2477 }
2478 }

```

End of the `\AtBeginDocument`.

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

```

2479 \cs_new_protected:Npn \@@_Hspace:
2480 {
2481     \bool_gset_true:N \g_@@_empty_cell_bool
2482     \hspace
2483 }

```

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

```

2484 \cs_set_eq:NN \@@_old_multicolumn \multicolumn
2485 \cs_new:Npn \@@_multicolumn:nnn #1 #2 #3
2486 {
2487     \@@_old_multicolumn { #1 } { #2 } { #3 }
2488     \int_compare:nNnT #1 > 1
2489     {
2490         \seq_gput_left:Nx \g_@@_multicolumn_cells_seq
2491         { \int_use:N \c@iRow - \int_use:N \c@jCol }
2492         \seq_gput_left:Nn \g_@@_multicolumn_sizes_seq { #1 }
2493     }
2494     \int_gadd:Nn \c@jCol { #1 - 1 }
2495 }

```

The command `\@@_Hdotsfor` will be linked to `\Hdotsfor` in `{NiceArrayWithDelims}`. Tikz nodes are created also in the implicit cells of the `\Hdotsfor` (maybe we should modify that point).

This command must *not* be protected since it begins with `\multicolumn`.

```

2496 \cs_new:Npn \@@_Hdotsfor:
2497 {
2498     \multicolumn { 1 } { C } { }
2499     \@@_Hdotsfor_i
2500 }

```

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:`).

```

2501 \AtBeginDocument
2502 {
2503     \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
2504     \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl

```

We don't put `!` before the last optionnal argument for homogeneity with `\Cdots`, etc. which have only one optional argument.

```

2505     \exp_args:NNV \NewDocumentCommand \@@_Hdotsfor_i \l_@@_argspec_tl
2506     {
2507         \tl_gput_right:Nx \g_@@_HVDotsfor_lines_tl
2508         {
2509             \@@_Hdotsfor:nnnn
2510             { \int_use:N \c@iRow }
2511             { \int_use:N \c@jCol }
2512             { #2 }

```



```

2513         {
2514             #1 , #3 ,
2515             down = \exp_not:n { #4 } , up = \exp_not:n { #5 }
2516         }
2517     }
2518     \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { C } { } }
2519 }
2520 }

```

Enf of \AtBeginDocument.

```

2521 \cs_new_protected:Npn \@@_Hdotsfor:nnnn #1 #2 #3 #4
2522 {
2523     \bool_set_false:N \l_@@_initial_open_bool
2524     \bool_set_false:N \l_@@_final_open_bool

```

For the row, it's easy.

```

2525     \int_set:Nn \l_@@_initial_i_int { #1 }
2526     \int_set_eq:NN \l_@@_final_i_int \l_@@_initial_i_int

```

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

```

2527     \int_compare:nNnTF #2 = 1
2528     {
2529         \int_set:Nn \l_@@_initial_j_int 1
2530         \bool_set_true:N \l_@@_initial_open_bool
2531     }
2532     {
2533         \cs_if_exist:cTF
2534         {
2535             pgf @ sh @ ns @ \@@_env:
2536             - \int_use:N \l_@@_initial_i_int
2537             - \int_eval:n { #2 - 1 }
2538         }
2539         { \int_set:Nn \l_@@_initial_j_int { #2 - 1 } }
2540         {
2541             \int_set:Nn \l_@@_initial_j_int { #2 }
2542             \bool_set_true:N \l_@@_initial_open_bool
2543         }
2544     }
2545     \int_compare:nNnTF { #2 + #3 - 1 } = \c@jCol
2546     {
2547         \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
2548         \bool_set_true:N \l_@@_final_open_bool
2549     }
2550     {
2551         \cs_if_exist:cTF
2552         {
2553             pgf @ sh @ ns @ \@@_env:
2554             - \int_use:N \l_@@_final_i_int
2555             - \int_eval:n { #2 + #3 }
2556         }
2557         { \int_set:Nn \l_@@_final_j_int { #2 + #3 } }
2558         {
2559             \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
2560             \bool_set_true:N \l_@@_final_open_bool
2561         }
2562     }
2563     \group_begin:
2564     \int_compare:nNnTF { #1 } = 0
2565     { \color { nicematrix-first-row } }
2566     {
2567         \int_compare:nNnT { #1 } = \g_@@_row_total_int
2568         { \color { nicematrix-last-row } }
2569     }

```

```

2570 \keys_set:nn { NiceMatrix / xdots } { #4 }
2571 \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2572 \@@_actually_draw_Ldots:
2573 \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).

```

2574 \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
2575 { \cs_set:cpn { @@ _ dotted _ #1 - ##1 } { } }
2576 }

2577 \AtBeginDocument
2578 {
2579 \tl_set:Nn \l_@@_argspec_tl { 0 { } m 0 { } E { _ ^ } { { } { } } }
2580 \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
2581 \exp_args:NNV \NewDocumentCommand \@@_Vdotsfor: \l_@@_argspec_tl
2582 {
2583 \tl_gput_right:Nx \g_@@_HVdotsfor_lines_tl
2584 {
2585 \@@_Vdotsfor:nnnn
2586 { \int_use:N \c@iRow }
2587 { \int_use:N \c@jCol }
2588 { #2 }
2589 {
2590 #1 , #3 ,
2591 down = \exp_not:n { #4 } , up = \exp_not:n { #5 }
2592 }
2593 }
2594 }
2595 }

```

Enf of \AtBeginDocument.

```

2596 \cs_new_protected:Npn \@@_Vdotsfor:nnnn #1 #2 #3 #4
2597 {
2598 \bool_set_false:N \l_@@_initial_open_bool
2599 \bool_set_false:N \l_@@_final_open_bool

```

For the column, it’s easy.

```

2600 \int_set:Nn \l_@@_initial_j_int { #2 }
2601 \int_set_eq:NN \l_@@_final_j_int \l_@@_initial_j_int

```

For the row, it’s a bit more complicated.

```

2602 \int_compare:nNnTF #1 = 1
2603 {
2604 \int_set:Nn \l_@@_initial_i_int 1
2605 \bool_set_true:N \l_@@_initial_open_bool
2606 }
2607 {
2608 \cs_if_exist:cTF
2609 {
2610 pgf @ sh @ ns @ \@@_env:
2611 - \int_eval:n { #1 - 1 }
2612 - \int_use:N \l_@@_initial_j_int
2613 }
2614 { \int_set:Nn \l_@@_initial_i_int { #1 - 1 } }
2615 {
2616 \int_set:Nn \l_@@_initial_i_int { #1 }
2617 \bool_set_true:N \l_@@_initial_open_bool
2618 }
2619 }
2620 \int_compare:nNnTF { #1 + #3 - 1 } = \c@iRow
2621 {

```

```

2622     \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }
2623     \bool_set_true:N \l_@@_final_open_bool
2624 }
2625 {
2626     \cs_if_exist:cTF
2627     {
2628         pgf @ sh @ ns @ \@@_env:
2629         - \int_eval:n { #1 + #3 }
2630         - \int_use:N \l_@@_final_j_int
2631     }
2632     { \int_set:Nn \l_@@_final_i_int { #1 + #3 } }
2633     {
2634         \int_set:Nn \l_@@_final_i_int { #1 + #3 - 1 }
2635         \bool_set_true:N \l_@@_final_open_bool
2636     }
2637 }
2638 \group_begin:
2639 \int_compare:nNnTF { #2 } = 0
2640 { \color { nicematrix-first-col } }
2641 {
2642     \int_compare:nNnT { #2 } = \g_@@_col_total_int
2643     { \color { nicematrix-last-col } }
2644 }
2645 \keys_set:nn { NiceMatrix / xdots } { #4 }
2646 \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2647 \@@_actually_draw_Vdots:
2648 \group_end:

```

We declare all the cells concerned by the `\Vdotsfor` 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).

```

2649     \int_step_inline:nnn { #1 } { #1 + #3 - 1 }
2650     { \cs_set:cpn { @@ _ dotted _ ##1 - #2 } { } }
2651 }

```

The command `\@@_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`.

```

2652 \cs_new_protected:Npn \@@_rotate: { \group_insert_after:N \@@_rotate_i: }
2653 \cs_new_protected:Npn \@@_rotate_i: { \group_insert_after:N \@@_rotate_ii: }
2654 \cs_new_protected:Npn \@@_rotate_ii: { \group_insert_after:N \@@_rotate_iii: }
2655 \cs_new_protected:Npn \@@_rotate_iii:
2656 {
2657     \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.

```

2658     \int_compare:nNnT \c@iRow = \l_@@_last_row_int
2659     {
2660         \vbox_set_top:Nn \l_@@_cell_box
2661         {
2662             \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`).

```

2663         \skip_vertical:n { - \box_ht:N \@arstrutbox + 0.8 ex }
2664         \box_use:N \l_@@_cell_box
2665     }
2666 }
2667 }

```

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).³⁵

```
2668 \cs_new:Npn \@@_double_int_eval:n #1-#2 \q_stop
2669   { \int_eval:n { #1 } - \int_eval:n { #2 } }
```

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

```
2670 \AtBeginDocument
2671 {
2672   \tl_set:Nn \l_@@_argspec_tl { 0 { } m m ! 0 { } E { _ ^ } { { } { } } }
2673   \tl_set_rescan:Nno \l_@@_argspec_tl { } \l_@@_argspec_tl
2674   \exp_args:NNV \NewDocumentCommand \@@_line \l_@@_argspec_tl
2675     {
2676       \group_begin:
2677       \keys_set:nn { NiceMatrix / xdots } { #1 , #4 , down = #5 , up = #6 }
2678       \tl_if_empty:VF \l_@@_xdots_color_tl { \color { \l_@@_xdots_color_tl } }
2679       \use:x
2680       {
2681         \@@_line_i:nn
2682         { \@@_double_int_eval:n #2 \q_stop }
2683         { \@@_double_int_eval:n #3 \q_stop }
2684       }
2685       \group_end:
2686     }
2687 }

2688 \cs_new_protected:Npn \@@_line_i:nn #1 #2
2689 {
2690   \bool_set_false:N \l_@@_initial_open_bool
2691   \bool_set_false:N \l_@@_final_open_bool
2692   \bool_if:nTF
2693     {
2694       \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #1 }
2695       ||
2696       \cs_if_free_p:c { pgf @ sh @ ns @ \@@_env: - #2 }
2697     }
2698     {
2699       \@@_error:nnn { unknown~cell~for~line~in~code~after } { #1 } { #2 }
2700     }
2701     { \@@_draw_line_ii:nn { #1 } { #2 } }
2702 }

2703 \AtBeginDocument
2704 {
2705   \cs_new_protected:Npx \@@_draw_line_ii:nn #1 #2
2706   {
```

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:.`

```
2707   \c_@@_pgfortikzpicture_tl
2708   \@@_draw_line_iii:nn { #1 } { #2 }
2709   \c_@@_endpgfortikzpicture_tl
2710 }
2711 }
```

³⁵Indeed, we want that the user may use the command `\line` in code-after with LaTeX counters in the arguments — with the command `\value`.

The following command *must* be protected (it's used in the construction of `\@@_draw_line_ii:nn`).

```

2712 \cs_new_protected:Npn \@@_draw_line_iii:nn #1 #2
2713 {
2714   \pgfrememberpicturepositiononpagetrue
2715   \pgfpointshapeborder { \@@_env: - #1 } { \@@_qpoint:n { #2 } }
2716   \dim_set_eq:NN \l_@@_x_initial_dim \pgf@x
2717   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2718   \pgfpointshapeborder { \@@_env: - #2 } { \@@_qpoint:n { #1 } }
2719   \dim_set_eq:NN \l_@@_x_final_dim \pgf@x
2720   \dim_set_eq:NN \l_@@_y_final_dim \pgf@y
2721   \@@_draw_line:
2722 }

```

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).

Commands available in the code-before

In the beginning of the code-before, the command `\@@_rowcolor:nn` will be linked to `\rowcolor` and the command `\@@_columncolor:nn` to `\columncolor`.

```

2723 \cs_set_protected:Npn \@@_cut_on_hyphen:w #1-#2\q_stop
2724 {
2725   \tl_set:Nn \l_tmpa_tl { #1 }
2726   \tl_set:Nn \l_tmpb_tl { #2 }
2727 }

```

Here an example : `\@@_rowcolor:nn {red!15} {1,3,5-7,10-}`

```

2728 \cs_new_protected:Npn \@@_rowcolor:nn #1 #2
2729 {
2730   \tl_if_blank:nF { #1 }
2731   {
2732     \pgfpicture
2733     \pgf@relevantforpicturesizefalse
2734     \pgfsetfillcolor { #1 }

```

`\l_tmpa_dim` is the x -value of the left side of the rows.

```

2735     \@@_qpoint:n { col - \@@_succ:n \c@jCol }
2736     \dim_set_eq:NN \l_tmpa_dim \pgf@x
2737     \clist_map_inline:nn { #2 }
2738     {
2739       \tl_set:Nn \l_tmpa_tl { ##1 }
2740       \tl_if_in:NnTF \l_tmpa_tl { - }
2741       { \@@_cut_on_hyphen:w ##1 \q_stop }
2742       { \@@_cut_on_hyphen:w ##1 - ##1 \q_stop }
2743       \tl_if_empty:NT \l_tmpa_tl { \tl_set:Nn \l_tmpa_tl { 1 } }
2744       \tl_if_empty:NT \l_tmpb_tl
2745       { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@iRow } }
2746       \int_compare:nNnT \l_tmpb_tl > \c@iRow
2747       { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@iRow } }

```

Now, the numbers of both rows are in `\l_tmpa_tl` and `\l_tmpb_tl`.

```

2748     \@@_qpoint:n { row - \@@_succ:n \l_tmpb_tl }
2749     \pgfpathrectanglecorners
2750     { \@@_qpoint:n { row - \l_tmpa_tl } }
2751     { \pgfpoint \l_tmpa_dim \pgf@y }
2752   }
2753   \pgfusepathqfill
2754   \endpgfpicture
2755 }
2756 }

```

Here an example : \@@_columncolor:nn {red!15} {1,3,5-7,10-}

```

2757 \cs_new_protected:Npn \@@_columncolor:nn #1 #2
2758 {
2759   \tl_if_blank:nF { #1 }
2760   {
2761     \pgfpicture
2762     \pgf@relevantforpicturesizefalse
2763     \pgfsetfillcolor { #1 }
2764     \@@_qpoint:n { row - 1 }

```

\l_tmpa_dim is the y -value of the top of the columns et \l_tmpb_dim is the y -value of the bottom.

```

2765     \dim_set_eq:NN \l_tmpa_dim \pgf@y
2766     \@@_qpoint:n { row - \@@_succ:n \c@iRow }
2767     \dim_set_eq:NN \l_tmpb_dim \pgf@y
2768     \clist_map_inline:nn { #2 }
2769     {
2770       \tl_set:Nn \l_tmpa_tl { ##1 }
2771       \tl_if_in:NnTF \l_tmpa_tl { - }
2772       { \@@_cut_on_hyphen:w ##1 \q_stop }
2773       { \@@_cut_on_hyphen:w ##1 - ##1 \q_stop }
2774       \tl_if_empty:NT \l_tmpa_tl { \tl_set:Nn \l_tmpa_tl { 1 } }
2775       \tl_if_empty:NT \l_tmpb_tl
2776       { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
2777       \int_compare:nNnT \l_tmpb_tl > \c@jCol
2778       { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }

```

Now, the numbers of both columns are in \l_tmpa_tl and \l_tmpb_tl.

```

2779       \@@_qpoint:n { col - \l_tmpa_tl }
2780       \dim_set_eq:NN \l_tmpc_dim \pgf@x
2781       \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
2782       \pgfpathrectanglecorners
2783       { \pgfpoint \l_tmpc_dim \l_tmpa_dim }
2784       { \pgfpoint \pgf@x \l_tmpb_dim }
2785     }
2786     \pgfusepathqfill
2787     \endpgfpicture
2788   }
2789 }

```

Here an example : \@@_cellcolor:nn {red!15}{2-3,3-4,4-5,5-6}

```

2790 \cs_new_protected:Npn \@@_cellcolor:nn #1 #2
2791 {
2792   \tl_if_blank:nF { #1 }
2793   {
2794     \pgfpicture
2795     \pgf@relevantforpicturesizefalse
2796     \pgfsetfillcolor { #1 }
2797     \clist_map_inline:nn { #2 }
2798     {
2799       \@@_cut_on_hyphen:w ##1 \q_stop
2800       \@@_qpoint:n { row - \l_tmpa_tl }
2801       \bool_lazy_and:nnT
2802       { \int_compare_p:n { \l_tmpa_tl <= \c@iRow } }
2803       { \int_compare_p:n { \l_tmpb_tl <= \c@jCol } }
2804       {
2805         \dim_set_eq:NN \l_tmpb_dim \pgf@y
2806         \@@_qpoint:n { row - \@@_succ:n \l_tmpa_tl }
2807         \dim_set_eq:NN \l_tmpa_dim \pgf@y
2808         \@@_qpoint:n { col - \l_tmpb_tl }
2809         \dim_set_eq:NN \l_tmpc_dim \pgf@x
2810         \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
2811         \dim_set_eq:NN \l_tmpd_dim \pgf@x
2812         \pgfpathrectanglecorners

```

```

2813         { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
2814         { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
2815     }
2816 }
2817 \pgfusepathqfill
2818 \endpgfpicture
2819 }
2820 }

```

Here an example : `\@@_rectanglecolor:nn {red!15}{2-3}{5-6}`

```

2821 \cs_new_protected:Npn \@@_rectanglecolor:nnn #1 #2 #3
2822 {
2823     \tl_if_blank:nF { #1 }
2824     {
2825         \pgfpicture
2826         \pgf@relevantforpicturesizefalse
2827         \pgfsetfillcolor { #1 }
2828         \@@_cut_on_hyphen:w #2 \q_stop
2829         \bool_lazy_and:nnT
2830             { \int_compare_p:n { \l_tmpa_tl <= \c@iRow } }
2831             { \int_compare_p:n { \l_tmpb_tl <= \c@jCol } }
2832         {
2833             \@@_qpoint:n { row - \l_tmpa_tl }
2834             \dim_set_eq:NN \l_tmpb_dim \pgf@y
2835             \@@_qpoint:n { col - \l_tmpb_tl }
2836             \@@_cut_on_hyphen:w #3 \q_stop
2837             \int_compare:nNnT \l_tmpa_tl > \c@iRow
2838                 { \tl_set:Nx \l_tmpa_tl { \int_use:N \c@iRow } }
2839             \int_compare:nNnT \l_tmpb_tl > \c@jCol
2840                 { \tl_set:Nx \l_tmpb_tl { \int_use:N \c@jCol } }
2841             \dim_set_eq:NN \l_tmpc_dim \pgf@x
2842             \@@_qpoint:n { row - \@@_succ:n \l_tmpa_tl }
2843             \dim_set_eq:NN \l_tmpa_dim \pgf@y
2844             \@@_qpoint:n { col - \@@_succ:n \l_tmpb_tl }
2845             \dim_set_eq:NN \l_tmpd_dim \pgf@x
2846             \pgfpathrectanglecorners
2847                 { \pgfpoint \l_tmpc_dim \l_tmpb_dim }
2848                 { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
2849             \pgfusepathqfill
2850         }
2851     }
2852 }
2853 }

```

The command `\rowcolors` (accessible in the code-before) is inspired by the command `\rowcolors` of the package `xcolor` (with the option `table`). However, the command `\rowcolors` of `nicematrix` has *not* the optional argument of the command `\rowcolors` of `xcolor`.

```

2854 \cs_new_protected:Npn \@@_rowcolors:nnn #1 #2 #3
2855 {
2856     \int_step_inline:nnn { #1 } { \int_use:N \c@iRow }
2857     {
2858         \int_if_odd:nTF { ##1 }
2859             { \@@_rowcolor:nn { #2 } { ##1 } }
2860             { \@@_rowcolor:nn { #3 } { ##1 } }
2861     }
2862 }

2863 \cs_new_protected:Npn \@@_chessboardcolors:nn #1 #2
2864 {
2865     \int_step_inline:nn { \int_use:N \c@iRow }
2866     {

```

```

2867 \int_step_inline:nn { \int_use:N \c@jCol }
2868 {
2869   \int_if_even:nTF { ####1 + ##1 }
2870   { \@@_cellcolor:nn { #1 } { ####1 - ##1 } }
2871   { \@@_cellcolor:nn { #2 } { ####1 - ##1 } }
2872 }
2873 }
2874 }

```

The vertical rules

We give to the user the possibility to define new types of columns (with `\newcolumntype` 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`.

```

2875 \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`.

```

2876 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix:n #1
2877 {
2878   \int_compare:nNnTF \l_@@_first_col_int = 0
2879   { \@@_OnlyMainNiceMatrix_i:n { #1 } }
2880   {
2881     \int_compare:nNnTF \c@jCol = 0
2882     {
2883       \int_compare:nNnF \c@iRow = { -1 }
2884       { \int_compare:nNnF \c@iRow = { \l_@@_last_row_int - 1 } { #1 } }
2885     }
2886     { \@@_OnlyMainNiceMatrix_i:n { #1 } }
2887   }
2888 }

```

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.

```

2889 \cs_new_protected:Npn \@@_OnlyMainNiceMatrix_i:n #1
2890 {
2891   \int_compare:nNnF \c@iRow = 0
2892   { \int_compare:nNnF \c@iRow = \l_@@_last_row_int { #1 } }
2893 }

```

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 `\newcolumntype`:

```

\newcolumntype { | } { ! { \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` in 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). The command `\CT@arc@` is a command of `colortbl` which sets the color of the rules in the array. The package `nicematrix` uses it even if `colortbl` is not loaded.

```
2894 \cs_new_protected:Npn \@@_vline:
2895   { \@@_OnlyMainNiceMatrix:n { { \CT@arc@ \vline } } }
```

The command `\@@_draw_vlines` will be executed when the user uses the option `vlines` (which draws all the vlines of the array).

```
2896 \cs_new_protected:Npn \@@_draw_vlines:
2897   {
2898     \group_begin:
```

The command `\CT@arc@` is a command of `colortbl` which sets the color of the rules in the array. The package `nicematrix` uses it even if `colortbl` is not loaded.

```
2899     \CT@arc@
2900     \pgfpicture
2901     \pgfrememberpicturepositiononpagetrue
2902     \pgf@relevantforpicturesizefalse
2903     \pgfsetlinewidth \arrayrulewidth
```

First, we compute in `\l_tmpa_dim` the height of the rules we have to draw.

```
2904     \@@_qpoint:n { row - 1 }
2905     \dim_set_eq:NN \l_tmpa_dim \pgf@y
2906     \pgfusepathqfill
2907     \@@_qpoint:n { row - \@@_succ:n \c@iRow }
2908     \dim_sub:Nn \l_tmpa_dim \pgf@y
2909     \pgfusepathqfill
```

We translate vertically to take into account the potential “last row”.

```
2910     \dim_zero:N \l_tmpb_dim
2911     \int_compare:nNnT \l_@@_last_row_int > { -1 }
2912     {
2913       \dim_set_eq:NN \l_tmpb_dim \g_@@_dp_last_row_dim
2914       \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”.

```
2915       \@@_qpoint:n { row - \@@_succ:n \c@iRow }
2916       \dim_add:Nn \l_tmpa_dim \pgf@y
2917       \@@_qpoint:n { row - \@@_succ:n \g_@@_row_total_int }
2918       \dim_sub:Nn \l_tmpa_dim \pgf@y
2919       \dim_sub:Nn \l_tmpa_dim \l_tmpb_dim
2920     }
2921     \dim_add:Nn \l_tmpa_dim \arrayrulewidth
```

Now, we can draw the vertical rules with a loop.

```
2922     \int_step_inline:nnn
2923       { \bool_if:NTF \l_@@_NiceArray_bool 1 2 }
2924       { \bool_if:NTF \l_@@_NiceArray_bool { \@@_succ:n \c@jCol } \c@jCol }
2925       {
2926         \pgfpathmoveto { \@@_qpoint:n { col - ##1 } }
2927         \pgfpathlineto
2928           {
2929             \pgfpointadd
2930               { \@@_qpoint:n { col - ##1 } }
2931               { \pgfpoint \c_zero_dim { \l_tmpb_dim + \l_tmpa_dim } }
2932           }
2933       }
```

```

2934 \pgfusepathqstroke
2935 \endpgfpicture
2936 \group_end:
2937 }

```

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`.

```

2938 \cs_new:Npn \@@_hdottedline:
2939 {
2940   \noalign { \skip_vertical:N 2\l_@@_radius_dim }
2941   \@@_hdottedline_i:
2942 }

```

On the other side, the following command should be protected.

```

2943 \cs_new_protected:Npn \@@_hdottedline_i:
2944 {

```

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).

```

2945   \tl_gput_right:Nx \g_@@_internal_code_after_tl
2946   { \@@_hdottedline:n { \int_use:N \c@iRow } }
2947 }

```

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.

```

2948 \AtBeginDocument
2949 {

```

We recall that, when externalization is used, `\tikzpicture` and `\endtikzpicture` (or `\pgfpicture` and `\endpgfpicture`) must be directly “visible”.

```

2950   \cs_new_protected:Npx \@@_hdottedline:n #1
2951   {
2952     \bool_set_true:N \exp_not:N \l_@@_initial_open_bool
2953     \bool_set_true:N \exp_not:N \l_@@_final_open_bool
2954     \c_@@_pgfortikzpicture_tl
2955     \@@_hdottedline_i:n { #1 }
2956     \c_@@_endpgfortikzpicture_tl
2957   }
2958 }

```

The following command *must* be protected since it is used in the construction of `\@@_hdottedline:n`.

```

2959 \cs_new_protected:Npn \@@_hdottedline_i:n #1
2960 {
2961   \pgfrememberpicturepositiononpagetrue
2962   \@@_qpoint:n { 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 rule drawn by “|” (considering the rule having a width equal to the diameter of the dots).

```

2963   \dim_set_eq:NN \l_@@_y_initial_dim \pgf@y
2964   \dim_sub:Nn \l_@@_y_initial_dim \l_@@_radius_dim
2965   \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}
```

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ \hdots & \hdots & \hdots & \hdots \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

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}
```

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ \hdots & \hdots & \hdots & \hdots \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

```
2966 \@@_qpoint:n { col - 1 }
2967 \dim_set:Nn \l_@@_x_initial_dim
2968 {
2969 \pgf@x +
2970 \bool_if:NTF \l_@@_NiceTabular_bool \tabcolsep \arraycolsep
2971 - \l_@@_left_margin_dim
2972 }
2973 \@@_qpoint:n { col - \@@_succ:n \c@jCol }
2974 \dim_set:Nn \l_@@_x_final_dim
2975 {
2976 \pgf@x -
2977 \bool_if:NTF \l_@@_NiceTabular_bool \tabcolsep \arraycolsep
2978 + \l_@@_right_margin_dim
2979 }
```

For reasons purely aesthetic, we do an adjustment in the case of a rounded bracket. The correction by `0.5 \l_@@_inter_dots_dim` is *ad hoc* for a better result.

```
2980 \tl_set:Nn \l_tmpa_tl { ( }
2981 \tl_if_eq:NNF \l_@@_left_delim_tl \l_tmpa_tl
2982 { \dim_gadd:Nn \l_@@_x_initial_dim { 0.5 \l_@@_inter_dots_dim } }
2983 \tl_set:Nn \l_tmpa_tl { ) }
2984 \tl_if_eq:NNF \l_@@_right_delim_tl \l_tmpa_tl
2985 { \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`.

```
2986 \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
2987 \@@_draw_line:
2988 }
```

Vertical dotted lines

```
2989 \cs_new_protected:Npn \@@_vdottedline:n #1
2990 {
2991 \bool_set_true:N \l_@@_initial_open_bool
2992 \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”.

```
2993 \bool_if:NTF \c_@@_tikz_loaded_bool
2994 {
2995 \tikzpicture
2996 \@@_vdottedline_i:n { #1 }
2997 \endtikzpicture
```

```

2998     }
2999     {
3000         \pgfpicture
3001         \@@_vdottedline_i:n { #1 }
3002         \endpgfpicture
3003     }
3004 }

```

```

3005 \cs_new_protected:Npn \@@_vdottedline_i:n #1
3006 {

```

The command `\CT@arc@` is a command of `colortbl` which sets the color of the rules in the array. The package `nicematrix` uses it even if `colortbl` is not loaded.

```

3007     \CT@arc@
3008     \pgfrememberpicturepositiononpagetrue
3009     \@@_qpoint:n { 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).

```

3010     \dim_set:Nn \l_@@_x_initial_dim { \pgf@x - \l_@@_radius_dim }
3011     \dim_set:Nn \l_@@_x_final_dim { \pgf@x - \l_@@_radius_dim }
3012     \@@_qpoint:n { 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.

```

3013     \dim_set:Nn \l_@@_y_initial_dim { \pgf@y - 0.5 \l_@@_inter_dots_dim }
3014     \@@_qpoint:n { row - \@@_succ:n \c@iRow }
3015     \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`.

```

3016     \tl_set_eq:NN \l_@@_xdots_line_style_tl \c_@@_standard_tl
3017     \@@_draw_line:
3018 }

```

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.

```

3019 \bool_new:N \l_@@_block_auto_columns_width_bool

```

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

```

3020 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
3021 {
3022     auto-columns-width .code:n =
3023     {
3024         \bool_set_true:N \l_@@_block_auto_columns_width_bool
3025         \dim_gzero_new:N \g_@@_max_cell_width_dim
3026         \bool_set_true:N \l_@@_auto_columns_width_bool
3027     }
3028 }

```

```

3029 \NewDocumentEnvironment { NiceMatrixBlock } { ! 0 { } }
3030 {
3031     \int_gincr:N \g_@@_NiceMatrixBlock_int
3032     \dim_zero:N \l_@@_columns_width_dim
3033     \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
3034     \bool_if:NT \l_@@_block_auto_columns_width_bool
3035     {

```

```

3036 \cs_if_exist:cT { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
3037 {
3038     \exp_args:NNc \dim_set:Nn \l_@@_columns_width_dim
3039     { @@_max_cell_width _ \int_use:N \g_@@_NiceMatrixBlock_int }
3040 }
3041 }
3042 }

```

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`).

```

3043 {
3044     \bool_if:NT \l_@@_block_auto_columns_width_bool
3045     {
3046         \iow_shipout:Nn \@mainaux \ExplSyntaxOn
3047         \iow_shipout:Nx \@mainaux
3048         {
3049             \cs_gset:cpn
3050             { @@ _ 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.

```

3051         { \dim_eval:n { \g_@@_max_cell_width_dim + \arrayrulewidth } }
3052     }
3053     \iow_shipout:Nn \@mainaux \ExplSyntaxOff
3054 }
3055 }

```

The extra nodes

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

```

3056 \cs_generate_variant:Nn \dim_min:nn { v n }
3057 \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`.

```

3058 \cs_new_protected:Npn \@@_computations_for_medium_nodes:
3059 {
3060     \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
3061     {
3062         \dim_zero_new:c { l_@@_row_\@@_i: _min_dim }
3063         \dim_set_eq:cn { l_@@_row_\@@_i: _min_dim } \c_max_dim
3064         \dim_zero_new:c { l_@@_row_\@@_i: _max_dim }
3065         \dim_set:cn { l_@@_row_\@@_i: _max_dim } { - \c_max_dim }

```

```

3066     }
3067     \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
3068     {
3069         \dim_zero_new:c { l_@@_column_\@@_j: _min_dim }
3070         \dim_set_eq:cn { l_@@_column_\@@_j: _min_dim } \c_max_dim
3071         \dim_zero_new:c { l_@@_column_\@@_j: _max_dim }
3072         \dim_set:cn { l_@@_column_\@@_j: _max_dim } { - \c_max_dim }
3073     }

```

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

```

3074     \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
3075     {
3076         \int_step_variable:nnNn
3077         \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.

```

3078         {
3079             \cs_if_exist:cT
3080             { 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`.

```

3081         {
3082             \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { south-west }
3083             \dim_set:cn { l_@@_row_\@@_i: _min_dim }
3084             { \dim_min:vn { l_@@_row _ \@@_i: _min_dim } \pgf@y }
3085             \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
3086             {
3087                 \dim_set:cn { l_@@_column _ \@@_j: _min_dim }
3088                 { \dim_min:vn { l_@@_column _ \@@_j: _min_dim } \pgf@x }
3089             }

```

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`.

```

3090             \pgfpointanchor { \@@_env: - \@@_i: - \@@_j: } { north-east }
3091             \dim_set:cn { l_@@_row _ \@@_i: _ max_dim }
3092             { \dim_max:vn { l_@@_row _ \@@_i: _ max_dim } \pgf@y }
3093             \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
3094             {
3095                 \dim_set:cn { l_@@_column _ \@@_j: _ max_dim }
3096                 { \dim_max:vn { l_@@_column _ \@@_j: _ max_dim } \pgf@x }
3097             }
3098         }
3099     }
3100 }

```

Now, we have to deal with empty rows or empty columns since we don't have created nodes in such rows and columns.

```

3101     \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
3102     {
3103         \dim_compare:nNnT
3104         { \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } } = \c_max_dim
3105         {
3106             \@@_qpoint:n { row - \@@_i: - base }
3107             \dim_set:cn { l_@@_row _ \@@_i: _ max _ dim } \pgf@y
3108             \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim } \pgf@y
3109         }
3110     }
3111     \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
3112     {
3113         \dim_compare:nNnT
3114         { \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } } = \c_max_dim
3115         {
3116             \@@_qpoint:n { col - \@@_j: }

```

```

3117         \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim } \pgf@y
3118         \dim_set:cn { l_@@_column _ \@@_j: _ min _ dim } \pgf@y
3119     }
3120 }
3121 }

```

Here is the command `\@@_create_medium_nodes:`. When this command is used, the “medium nodes” are created.

```

3122 \cs_new_protected:Npn \@@_create_medium_nodes:
3123 {
3124     \pgfpicture
3125     \pgfrememberpicturepositiononpagetrue
3126     \pgf@relevantforpicturesizefalse
3127     \@@_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”.

```

3128     \tl_set:Nn \l_@@_suffix_tl { -medium }
3129     \@@_create_nodes:
3130     \endpgfpicture
3131 }

```

The command `\@@_create_large_nodes:` must be used when we want to create only the “large nodes” and not the medium ones³⁶. 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:`.

```

3132 \cs_new_protected:Npn \@@_create_large_nodes:
3133 {
3134     \pgfpicture
3135     \pgfrememberpicturepositiononpagetrue
3136     \pgf@relevantforpicturesizefalse
3137     \@@_computations_for_medium_nodes:
3138     \@@_computations_for_large_nodes:
3139     \tl_set:Nn \l_@@_suffix_tl { - large }
3140     \@@_create_nodes:
3141     \endpgfpicture
3142 }
3143 \cs_new_protected:Npn \@@_create_medium_and_large_nodes:
3144 {
3145     \pgfpicture
3146     \pgfrememberpicturepositiononpagetrue
3147     \pgf@relevantforpicturesizefalse
3148     \@@_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”.

```

3149     \tl_set:Nn \l_@@_suffix_tl { - medium }
3150     \@@_create_nodes:
3151     \@@_computations_for_large_nodes:
3152     \tl_set:Nn \l_@@_suffix_tl { - large }
3153     \@@_create_nodes:
3154     \endpgfpicture
3155 }

```

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.

```

3156 \cs_new_protected:Npn \@@_computations_for_large_nodes:
3157 {
3158     \int_set:Nn \l_@@_first_row_int 1
3159     \int_set:Nn \l_@@_first_col_int 1

```

³⁶If we want to create both, we have to use `\@@_create_medium_and_large_nodes:`

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`.

```

3160 \int_step_variable:nNn { \c@iRow - 1 } \@@_i:
3161 {
3162   \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim }
3163   {
3164     (
3165       \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } +
3166       \dim_use:c { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
3167     )
3168     / 2
3169   }
3170   \dim_set_eq:cc { l_@@_row _ \@@_succ:n \@@_i: _ max _ dim }
3171   { l_@@_row _ \@@_i: _ min _ dim }
3172 }
3173 \int_step_variable:nNn { \c@jCol - 1 } \@@_j:
3174 {
3175   \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim }
3176   {
3177     (
3178       \dim_use:c { l_@@_column _ \@@_j: _ max _ dim } +
3179       \dim_use:c
3180       { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
3181     )
3182     / 2
3183   }
3184   \dim_set_eq:cc { l_@@_column _ \@@_succ:n \@@_j: _ min _ dim }
3185   { l_@@_column _ \@@_j: _ max _ dim }
3186 }

```

Here, we have to use `\dim_sub:cn` because of the number 1 in the name.

```

3187 \dim_sub:cn
3188 { l_@@_column _ 1 _ min _ dim }
3189 \l_@@_left_margin_dim
3190 \dim_add:cn
3191 { l_@@_column _ \int_use:N \c@jCol _ max _ dim }
3192 \l_@@_right_margin_dim
3193 }

```

The command `\@@_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).

```

3194 \cs_new_protected:Npn \@@_create_nodes:
3195 {
3196   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
3197   {
3198     \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
3199     {

```

We draw the rectangular node for the cell (`\@@_i-\@@_j`).

```

3200 \@@_pgf_rect_node:nnnnn
3201 { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
3202 { \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } }
3203 { \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } }
3204 { \dim_use:c { l_@@_column _ \@@_j: _ max _ dim } }
3205 { \dim_use:c { l_@@_row _ \@@_i: _ max _ dim } }
3206 \str_if_empty:NF \l_@@_name_str
3207 {
3208   \pgfnodealias
3209   { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }

```



```

3210         { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
3211     }
3212 }
3213 }

```

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 .

```

3214 \seq_mapthread_function:NNN
3215   \g_@@_multicolumn_cells_seq
3216   \g_@@_multicolumn_sizes_seq
3217   \@@_node_for_multicolumn:nn
3218 }

```

```

3219 \cs_new_protected:Npn \@@_extract_coords_values: #1 - #2 \q_stop
3220 {
3221   \cs_set:Npn \@@_i: { #1 }
3222   \cs_set:Npn \@@_j: { #2 }
3223 }

```

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”).

```

3224 \cs_new_protected:Npn \@@_node_for_multicolumn:nn #1 #2
3225 {
3226   \@@_extract_coords_values: #1 \q_stop
3227   \@@_pgf_rect_node:nnnnn
3228     { \@@_env: - \@@_i: - \@@_j: \l_@@_suffix_tl }
3229     { \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } }
3230     { \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } }
3231     { \dim_use:c { l_@@_column _ \int_eval:n { \@@_j: +#2-1 } _ max _ dim } }
3232     { \dim_use:c { l_@@_row _ \@@_i: _ max _ dim } }
3233   \str_if_empty:NF \l_@@_name_str
3234   {
3235     \pgfnodealias
3236       { \l_@@_name_str - \@@_i: - \@@_j: \l_@@_suffix_tl }
3237       { \int_use:N \g_@@_env_int - \@@_i: - \@@_j: \l_@@_suffix_tl }
3238   }
3239 }

```

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)

```

3240 \NewDocumentCommand \@@_Block: { 0 { } m D < > { } m }
3241 { \@@_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.

```

3242 \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.

```

3243 \cs_new_protected:Npn \@@_Block_ii:nnnnn #1 #2 #3 #4 #5
3244 {

```

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).

```

3245 \tl_gput_left:Nx \g_@@_internal_code_after_tl
3246 {
3247   \@@_Block_iii:nnnnnn
3248   { \int_use:N \c@iRow }
3249   { \int_use:N \c@jCol }
3250   { \int_eval:n { \c@iRow + #1 - 1 } }
3251   { \int_eval:n { \c@jCol + #2 - 1 } }
3252   { #3 }
3253   \exp_not:n { { #4 $ #5 $ } }
3254 }

```

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.

```

3255 \cs_set_eq:NN \Block \@@_Block_error:nn
3256 }

3257 \cs_new:Npn \@@_Block_error:nn #1 #2
3258 {
3259   \@@_error:n { Second~Block }
3260   \cs_set_eq:NN \Block \use:nn
3261 }

3262 \keys_define:nn { NiceMatrix / Block }
3263 {
3264   tikz .tl_set:N = \l_@@_tikz_tl ,
3265   tikz .value_required:n = true ,
3266   white .bool_set:N = \l_@@_white_bool ,
3267   white .default:n = true ,
3268   white .value_forbidden:n = true ,
3269 }

```

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

```

3270 \cs_new_protected:Npn \@@_Block_iii:nnnnnn #1 #2 #3 #4 #5 #6
3271 {

```

The group is for the keys.

```

3272   \group_begin:
3273   \keys_set:nn { NiceMatrix / Block } { #5 }
3274   \bool_if:nTF
3275   {
3276     \int_compare_p:nNn { #3 } > \c@iRow
3277     || \int_compare_p:nNn { #4 } > \c@jCol
3278   }
3279   { \msg_error:nnnn { nicematrix } { Block-too~large } { #1 } { #2 } }
3280   {

```

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.

```

3281   \hbox_set:Nn \l_@@_cell_box { #6 }

```

The construction of the node corresponding to the merged cells.

```

3282   \pgfpicture
3283   \pgfrememberpicturepositiononpagetrue
3284   \pgf@relevantforpicturesizefalse
3285   \@@_qpoint:n { row - #1 }
3286   \dim_set_eq:NN \l_tmpa_dim \pgf@y
3287   \@@_qpoint:n { col - #2 }
3288   \dim_set_eq:NN \l_tmpb_dim \pgf@x

```

```

3289 \@@_qpoint:n { row - \@@_succ:n { #3 } }
3290 \dim_set_eq:NN \l_tmpc_dim \pgf@y
3291 \@@_qpoint:n { col - \@@_succ:n { #4 } }
3292 \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.

```

3293 \bool_if:NT \l_@@_white_bool
3294 {
3295   \begin { pgfscope }
3296   \pgfsetfillcolor { white }
3297   \pgfpathrectanglecorners
3298   {
3299     \pgfpoint
3300     { \l_tmpb_dim + 0.5 \arrayrulewidth }
3301     { \l_tmpa_dim - 0.5 \arrayrulewidth }
3302   }
3303   {
3304     \pgfpoint
3305     { \l_tmpd_dim - 0.5 \arrayrulewidth }
3306     { \l_tmpc_dim + 0.5 \arrayrulewidth }
3307   }
3308   \pgfusepathqfill
3309   \end { pgfscope }
3310 }

```

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.

```

3311 \begin { pgfscope }
3312 \exp_args:Nx \pgfset { \l_@@_tikz_tl }
3313 \@@_pgf_rect_node:nnnnn
3314 { \@@_env: - #1 - #2 - block }
3315 \l_tmpb_dim \l_tmpa_dim \l_tmpd_dim \l_tmpc_dim
3316 \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.

```

3317 \bool_if:NT \l_@@_medium_nodes_bool
3318 {
3319   \@@_pgf_rect_node:nnn
3320   { \@@_env: - #1 - #2 - block - medium }
3321   { \pgfpointanchor { \@@_env: - #1 - #2 - medium } { north-west } }
3322   { \pgfpointanchor { \@@_env: - #3 - #4 - medium } { south-east } }
3323 }

```

Now, we will put the label of the block.

```

3324 \int_compare:nNnTF { #1 } = { #3 }
3325 {

```

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`.

```

3326 \pgfextracty \l_tmpa_dim { \@@_qpoint:n { row - #1 - base } }

```

We retrieve (in `\pgf@x`) the x -value of the center of the block.

```

3327 \@@_qpoint:n { #1 - #2 - block }

```

We put the label of the block which has been composed in `\l_@@_cell_box`.

```

3328 \pgftransformshift { \pgfpoint \pgf@x \l_tmpa_dim }
3329 \pgfnode { rectangle } { base }
3330 { \box_use_drop:N \l_@@_cell_box } { } { }
3331 }

```

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`).

```

3332     {
3333         \pgftransformshift { \l_@@_qpoint:n { #1 - #2 - block } }
3334         \pgfnode { rectangle } { center }
3335         { \box_use_drop:N \l_@@_cell_box } { } { }
3336     }
3337     \endpgfpicture
3338 }
3339 \group_end:
3340 }

```

How to draw the dotted lines transparently

```

3341 \cs_set_protected:Npn \l_@@_renew_matrix:
3342 {
3343     \RenewDocumentEnvironment { pmatrix } { } { }
3344     { \pNiceMatrix }
3345     { \endpNiceMatrix }
3346     \RenewDocumentEnvironment { vmatrix } { } { }
3347     { \vNiceMatrix }
3348     { \endvNiceMatrix }
3349     \RenewDocumentEnvironment { Vmatrix } { } { }
3350     { \VNiceMatrix }
3351     { \endVNiceMatrix }
3352     \RenewDocumentEnvironment { bmatrix } { } { }
3353     { \bNiceMatrix }
3354     { \endbNiceMatrix }
3355     \RenewDocumentEnvironment { Bmatrix } { } { }
3356     { \BNiceMatrix }
3357     { \endBNiceMatrix }
3358 }

```

Automatic arrays

```

3359 \cs_new_protected:Npn \l_@@_set_size:n #1-#2 \q_stop
3360 {
3361     \int_set:Nn \l_@@_nb_rows_int { #1 }
3362     \int_set:Nn \l_@@_nb_cols_int { #2 }
3363 }
3364 \NewDocumentCommand \AutoNiceMatrixWithDelims { m m O { } m O { } m ! O { } }
3365 {
3366     \int_zero_new:N \l_@@_nb_rows_int
3367     \int_zero_new:N \l_@@_nb_cols_int
3368     \l_@@_set_size:n #4 \q_stop
3369     \begin { NiceArrayWithDelims } { #1 } { #2 }
3370         { * { \l_@@_nb_cols_int } { C } } [ #3 , #5 , #7 ]
3371     \int_compare:nNnT \l_@@_first_row_int = 0
3372     {
3373         \int_compare:nNnT \l_@@_first_col_int = 0 { & }
3374         \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
3375         \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
3376     }
3377     \prg_replicate:nn \l_@@_nb_rows_int
3378     {
3379         \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`).

```

3380     \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { { } #6 & } #6
3381     \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\\
3382   }
3383   \int_compare:nNnT \l_@@_last_row_int > { -2 }
3384   {
3385     \int_compare:nNnT \l_@@_first_col_int = 0 { & }
3386     \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
3387     \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\\
3388   }
3389   \end { NiceArrayWithDelims }
3390 }
3391 \cs_set_protected:Npn \@@_define_com:nnn #1 #2 #3
3392 {
3393   \cs_set_protected:cpn { #1 AutoNiceMatrix }
3394   {
3395     \str_gset:Nx \g_@@_name_env_str { #1 AutoNiceMatrix }
3396     \AutoNiceMatrixWithDelims { #2 } { #3 }
3397   }
3398 }
3399 \@@_define_com:nnn p ( )
3400 \@@_define_com:nnn b [ ]
3401 \@@_define_com:nnn v | |
3402 \@@_define_com:nnn V \| \|
3403 \@@_define_com:nnn B \{ \}

```

We define also an command `\AutoNiceMatrix` similar to the environment `{NiceMatrix}`.

```

3404 \NewDocumentCommand \AutoNiceMatrix { O { } m O { } m ! O { } }
3405 {
3406   \group_begin:
3407     \bool_set_true:N \l_@@_NiceArray_bool
3408     \AutoNiceMatrixWithDelims . . { #2 } { #4 } [ #1 , #3 , #5 ]
3409   \group_end:
3410 }

```

The redefinition of the command `\dotfill`

```

3411 \cs_set_eq:NN \@@_dotfill \dotfill
3412 \cs_new_protected:Npn \@@_dotfill:
3413 {

```

First, we insert `\@@_dotfill` (which is the saved version of `\dotfill`) in case of use of `\dotfill` “internally” in the cell (e.g. `\hbox to 1cm {\dotfill}`).

```

3414   \@@_dotfill
3415   \group_insert_after:N \@@_dotfill_i:
3416 }
3417 \cs_new_protected:Npn \@@_dotfill_i: { \group_insert_after:N \@@_dotfill_ii: }
3418 \cs_new_protected:Npn \@@_dotfill_ii: { \group_insert_after:N \@@_dotfill_iii: }

```

Now, if the box is not empty (unfortunately, we can’t actually test whether the box is empty and that’s why we only consider it’s width), we insert `\@@_dotfill` (which is the saved version of `\dotfill`) in the cell of the array, and it will extends, since it’s no longer in `\l_@@_cell_box`.

```

3419 \cs_new_protected:Npn \@@_dotfill_iii:
3420 { \dim_compare:nNnT { \box_wd:N \l_@@_cell_box } = \c_zero_dim { \@@_dotfill }

```

The command `\slashbox`

```

3421 \cs_new_protected:Npn \@@_slashbox:nn #1 #2
3422 {
3423   \tl_gput_right:Nx \g_@@_internal_code_after_tl
3424   {
3425     \@@_actually_slashbox:nnnn
3426     { \int_use:N \c_iRow } { \int_use:N \c_jCol } { #1 } { #2 }
3427   }
3428 }

```

The two arguments of `\@@_actually_slashbox:nn` are the number of row and the number of column of the cell to slash. The two other are the elements to draw below and above the diagonal line.

```

3429 \cs_new_protected:Npn \@@_actually_slashbox:nnnn #1 #2 #3 #4
3430 {
3431   \pgfpicture
3432   \pgf@relevantforpicturesizefalse
3433   \pgfrememberpicturepositiononpagetrue
3434   \@@_qpoint:n { row - #1 }
3435   \dim_set_eq:NN \l_tmpa_dim \pgf@y
3436   \@@_qpoint:n { col - #2 }
3437   \dim_set_eq:NN \l_tmpb_dim \pgf@x
3438   \pgfpathmoveto { \pgfpoint \l_tmpb_dim \l_tmpa_dim }
3439   \@@_qpoint:n { row - \@@_succ:n { #1 } }
3440   \dim_set_eq:NN \l_tmpc_dim \pgf@y
3441   \@@_qpoint:n { col - \@@_succ:n { #2 } }
3442   \dim_set_eq:NN \l_tmpd_dim \pgf@x
3443   \pgfpathlineto { \pgfpoint \l_tmpd_dim \l_tmpc_dim }
3444   {

```

The command `\CT@arc@` is a command of `colortbl` which sets the color of the rules in the array. The package `nicematrix` uses it even if `colortbl` is not loaded.

```

3445   \CT@arc@
3446   \pgfsetroundcap
3447   \pgfusepathqstroke
3448 }
3449 \pgfset { inner~sep = 1 pt }
3450 \pgfscope
3451 \pgftransformshift { \pgfpoint \l_tmpb_dim \l_tmpc_dim }
3452 \pgfnode { rectangle } { south~west } { $ #3 $ } { } { }
3453 \endpgfscope
3454 \pgftransformshift { \pgfpoint \l_tmpd_dim \l_tmpa_dim }
3455 \pgfnode { rectangle } { north~east } { $ #4 $ } { } { }
3456 \endpgfpicture
3457 }

```

The command `\CodeAfter`

In fact, in this subsection, we define the user command `\CodeAfter` for the case of the “normal syntax”. For the case of “light-syntax”, see the definition of the environment `{@@-light-syntax}` on p. 61.

The command `\CodeAfter` catches everything until the end of the current environment (of `nicematrix`). First, we go until the next command `\end`.

```

3458 \cs_new_protected:Npn \@@_CodeAfter:n #1 \end
3459 {
3460   \tl_gput_right:Nn \g_@@_code_after_tl { #1 }
3461   \@@_CodeAfter_i:n
3462 }

```

We catch the argument of the command `\end` (in `#1`).

```

3463 \cs_new_protected:Npn \@@_CodeAfter_i:n #1
3464 {

```

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.

```

3465   \str_set:NV \l_tmpa_str \@currenvir
3466   \bool_if:NTF { \str_if_eq_p:Vn \l_tmpa_str { #1 } }
3467   { \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`.

```

3468   {
3469     \tl_gput_right:Nn \g_@@_code_after_tl { \end { #1 } }
3470     \@@_CodeAfter:n
3471   }
3472 }

```

We process the options

```
3473 \@@_msg_new:nn { obsolete~environments }
3474 {
3475   The~obsolete~environments~(eg.~{pNiceArrayC})~have~been~deleted~
3476   from~the~package~nicematrix.\\
3477   However,~if~you~still~want~to~use~them,~you~will~find~the~code~for~
3478   those~environments~at~the~end~of~the~file~'nicematrix.sty',~after~
3479   a~command~'\token_to_str:N\file_input_stop:'.
3480 }
```

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.

```
3481 \keys_define:nn { NiceMatrix / Package }
3482 {
3483   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
3484   renew-dots .value_forbidden:n = true ,
3485   renew-matrix .code:n = \@@_renew_matrix: ,
3486   renew-matrix .value_forbidden:n = true ,
3487   transparent .meta:n = { renew-dots , renew-matrix } ,
3488   transparent .value_forbidden:n = true,
3489   obsolete-environments .code:n = \@@_fatal:n { obsolete~environments },
3490   starred-commands .code:n =
3491     \@@_msg_redirect_name:nn { starred~commands } { none } ,
3492   starred-commands .value_forbidden:n = true ,
3493 }
3494 }
3495 \ProcessKeysOptions { NiceMatrix / Package }
```

Error messages of the package

The following command converts all the elements of a sequence (which are token lists) into strings.

```
3496 \cs_new_protected:Npn \@@_convert_to_str_seq:N #1
3497 {
3498   \seq_clear:N \l_tmpa_seq
3499   \seq_map_inline:Nn #1
3500   {
3501     \seq_put_left:Nx \l_tmpa_seq { \tl_to_str:n { ##1 } }
3502   }
3503   \seq_set_eq:NN #1 \l_tmpa_seq
3504 }
```

The following command creates a sequence of strings (`str`) from a `clist`.

```
3505 \cs_new_protected:Npn \@@_set_seq_of_str_from_clist:Nn #1 #2
3506 {
3507   \seq_set_from_clist:Nn #1 { #2 }
3508   \@@_convert_to_str_seq:N #1
3509 }
3510 \@@_set_seq_of_str_from_clist:Nn \c_@@_types_of_matrix_seq
3511 {
3512   NiceMatrix ,
3513   pNiceMatrix , bNiceMatrix , vNiceMatrix, BNiceMatrix, VNiceMatrix
3514 }
```

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`.

```
3515 \cs_new_protected:Npn \@@_error_too_much_cols:
```

```

3516 {
3517   \seq_if_in:NVTf \c_@@_types_of_matrix_seq \g_@@_name_env_str
3518   {
3519     \int_compare:nNtF \l_@@_last_col_int = { -2 }
3520     { \@@_fatal:n { too-much-cols-for-matrix } }
3521     {
3522       \bool_if:NF \l_@@_last_col_without_value_bool
3523       { \@@_fatal:n { too-much-cols-for-matrix-with-last-col } }
3524     }
3525   }
3526   { \@@_fatal:n { too-much-cols-for-array } }
3527 }

```

The following command must *not* be protected since it's used in an error message.

```

3528 \cs_new:Npn \@@_message_hdotsfor:
3529 {
3530   \tl_if_empty:VF \g_@@_HVdotsfor_lines_tl
3531   { ~Maybe-your-use-of~\token_to_str:N \Hdotsfor\ is-incorrect.}
3532 }
3533 \@@_msg_new:nn { too-much-cols-for-matrix-with-last-col }
3534 {
3535   You-try-to-use-more-columns-than-allowed-by-your~
3536   \@@_full_name_env:.\@@_message_hdotsfor:\ The-maximal-number-of~
3537   columns-is~\int_eval:n { \l_@@_last_col_int - 1 }~(plus-the-potential~
3538   exterior-ones).~This-error-is-fatal.
3539 }
3540 \@@_msg_new:nn { too-much-cols-for-matrix }
3541 {
3542   You-try-to-use-more-columns-than-allowed-by-your~
3543   \@@_full_name_env:.\@@_message_hdotsfor:\ Recall-that-the-maximal~
3544   number-of-columns-for-a-matrix-is-fixed-by-the-LaTeX-counter~
3545   'MaxMatrixCols'.~Its-actual-value-is~\int_use:N \c@MaxMatrixCols.~
3546   This-error-is-fatal.
3547 }

```

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`.

```

3548 \@@_msg_new:nn { too-much-cols-for-array }
3549 {
3550   You-try-to-use-more-columns-than-allowed-by-your~
3551   \@@_full_name_env:.\@@_message_hdotsfor:\ The-maximal-number-of-columns-is~
3552   \int_eval:n { \c@jCol - 1 }~(plus-the-potential-exterior-ones).~
3553   This-error-is-fatal.
3554 }
3555 \@@_msg_new:nn { in-first-col }
3556 {
3557   You-can't-use-the-command~#1 in-the-first-column-(number~0)~of-the-array.\\
3558   If-you-go-on,~this-command-will-be-ignored.
3559 }
3560 \@@_msg_new:nn { in-last-col }
3561 {
3562   You-can't-use-the-command~#1 in-the-last-column-(exterior)~of-the-array.\\
3563   If-you-go-on,~this-command-will-be-ignored.
3564 }
3565 \@@_msg_new:nn { in-first-row }
3566 {
3567   You-can't-use-the-command~#1 in-the-first-row-(number~0)~of-the-array.\\
3568   If-you-go-on,~this-command-will-be-ignored.
3569 }

```



```

3570 \@@_msg_new:nn { in-last-row }
3571 {
3572   You~can't~use~the~command~\#1~in~the~last~row~(exterior)~of~the~array.\\
3573   If~you~go~on,~this~command~will~be~ignored.
3574 }
3575 \@@_msg_new:nn { option-S~without~siunitx }
3576 {
3577   You~can't~use~the~option~'S'~in~your~environment~\@@_full_name_env:
3578   because~you~have~not~loaded~siunitx.\\
3579   If~you~go~on,~this~option~will~be~ignored.
3580 }
3581 \@@_msg_new:nn { bad-option-for~line-style }
3582 {
3583   Since~you~haven't~loaded~Tikz,~the~only~value~you~can~give~to~'line-style'~
3584   is~'standard'.~If~you~go~on,~this~option~will~be~ignored.
3585 }
3586 \@@_msg_new:nn { Unknown~option~for~xdots }
3587 {
3588   As~for~now~there~is~only~three~options~available~here:~'color',~'line-style'~
3589   and~'shorten'~(and~you~try~to~use~'\l_keys_key_str').~If~you~go~on,~
3590   this~option~will~be~ignored.
3591 }
3592 \@@_msg_new:nn { ampersand~in~light-syntax }
3593 {
3594   You~can't~use~an~ampersand~(\token_to_str &)~to~separate~columns~because
3595   ~you~have~used~the~option~'light-syntax'.~This~error~is~fatal.
3596 }
3597 \@@_msg_new:nn { double-backslash~in~light-syntax }
3598 {
3599   You~can't~use~\token_to_str:N~\\~to~separate~rows~because~you~have~used~
3600   the~option~'light-syntax'.~You~must~use~the~character~'\l_@@_end_of_row_tl'~
3601   (set~by~the~option~'end-of-row').~This~error~is~fatal.
3602 }
3603 \@@_msg_new:nn { starred~commands }
3604 {
3605   The~starred~versions~of~\token_to_str:N~\Cdots,~\token_to_str:N~\Ldots,~
3606   \token_to_str:N~\Vdots,~\token_to_str:N~\Ddots~and~\token_to_str:N~\Iddots~
3607   are~deprecated.~However,~you~can~go~on~for~this~time.~If~you~don't~want~to~
3608   see~this~error~we~should~load~'nicematrix'~with~the~option~
3609   'starred-commands'.
3610 }
3611 \@@_msg_new:nn { bad~value~for~baseline }
3612 {
3613   The~value~given~to~'baseline'~(\int_use:N~\l_tmpa_int)~is~not~
3614   valid.~The~value~must~be~between~\int_use:N~\l_@@_first_row_int~and~
3615   \int_use:N~\g_@@_row_total_int~or~equal~to~'t',~'c'~or~'b'.\\
3616   If~you~go~on,~a~value~of~1~will~be~used.
3617 }
3618 \@@_msg_new:nn { Second~Block }
3619 {
3620   You~can't~use~\token_to_str:N~\Block~twice~in~the~same~cell~of~the~array.\\
3621   If~you~go~on,~this~command~(and~the~other)~will~be~ignored.
3622 }
3623 \@@_msg_new:nn { empty~environment }
3624 { Your~\@@_full_name_env:~is~empty.~This~error~is~fatal. }
3625 \@@_msg_new:nn { unknown~cell~for~line~in~code~after }
3626 {
3627   Your~command~\token_to_str:N~\line~\{#1\}\{#2\}~in~the~'code~after'~
3628   can't~be~executed~because~a~cell~doesn't~exist.\\

```

```

3629     If~you~go~on~this~command~will~be~ignored.
3630 }

3631 \@@_msg_new:nn { last-col~non~empty~for~NiceArray }
3632 {
3633     In~the~\@@_full_name_env:,~you~must~use~the~option~
3634     'last-col'~without~value.\\
3635     However,~you~can~go~on~for~this~time~
3636     (the~value~'\l_keys_value_tl'~will~be~ignored).
3637 }

3638 \@@_msg_new:nn { last-col~non~empty~for~NiceMatrixOptions }
3639 {
3640     In~\NiceMatrixoptions,~you~must~use~the~option~
3641     'last-col'~without~value.\\
3642     However,~you~can~go~on~for~this~time~
3643     (the~value~'\l_keys_value_tl'~will~be~ignored).
3644 }

3645 \@@_msg_new:nn { Block~too~large }
3646 {
3647     You~try~to~draw~a~block~in~the~cell~#1-#2~of~your~matrix~but~the~matrix~is~
3648     too~small~for~that~block. \\
3649     If~you~go~on,~this~command~will~be~ignored.
3650 }

3651 \@@_msg_new:nn { Wrong~last~row }
3652 {
3653     You~have~used~'last-row=\int_use:N \l_@@_last_row_int'~but~your~
3654     \@@_full_name_env:\ seems~to~have~\int_use:N \c@iRow \ rows.~
3655     If~you~go~on,~the~value~of~\int_use:N \c@iRow \ will~be~used~for~
3656     last~row.~You~can~avoid~this~problem~by~using~'last-row'~
3657     without~value~(more~compilations~might~be~necessary).
3658 }

3659 \@@_msg_new:nn { Yet~in~env }
3660 {
3661     Environments~\{NiceArray\}~(or~\{NiceMatrix\},~etc.)~can't~be~nested.\\
3662     This~error~is~fatal.
3663 }

3664 \@@_msg_new:nn { Outside~math~mode }
3665 {
3666     The~\@@_full_name_env:\ can~be~used~only~in~math~mode~
3667     (and~not~in~\token_to_str:N \vcenter).\\
3668     This~error~is~fatal.
3669 }

3670 \@@_msg_new:nn { Bad~value~for~letter~for~dotted~lines }
3671 {
3672     The~value~of~key~'\tl_use:N \l_keys_key_str'~must~be~of~length~1.\\
3673     If~you~go~on,~it~will~be~ignored.
3674 }

3675 \@@_msg_new:nnn { Unknown~key~for~NiceMatrixOptions }
3676 {
3677     The~key~'\tl_use:N \l_keys_key_str'~is~unknown~for~the~command~
3678     \token_to_str:N \NiceMatrixOptions. \\
3679     If~you~go~on,~it~will~be~ignored. \\
3680     For~a~list~of~the~available~keys,~type~H~<return>.
3681 }
3682 {
3683     The~available~options~are~(in~alphabetic~order):~
3684     allow~duplicate~names,~
3685     code~for~first~col,~
3686     cell~space~bottom~limit,~
3687     cell~space~top~limit,~
3688     code~for~first~row,~

```

```

3689 code-for-last-col,~
3690 code-for-last-row,~
3691 create-extra-nodes,~
3692 create-medium-nodes,~
3693 create-large-nodes,~
3694 end-of-row,~
3695 exterior-arraycolsep,~
3696 first-col,~
3697 first-row,~
3698 hlines,~
3699 hvlines,~
3700 last-col,~
3701 last-row,~
3702 left-margin,~
3703 letter-for-dotted-lines,~
3704 light-syntax,~
3705 nullify-dots,~
3706 parallelize-diags,~
3707 renew-dots,~
3708 renew-matrix,~
3709 right-margin,~
3710 small,~
3711 transparent,~
3712 vlines,~
3713 xdots/color,~
3714 xdots/shorten-and~
3715 xdots/line-style.
3716 }

3717 \@@_msg_new:nnn { Unknown~option~for~NiceArray }
3718 {
3719   The~option~'\tl_use:N\l_keys_key_str'~is~unknown~for~the~environment~
3720   \{NiceArray\}. \\
3721   If~you~go~on,~it~will~be~ignored. \\
3722   For~a~list~of~the~available~options,~type~H~<return>.
3723 }
3724 {
3725   The~available~options~are~(in~alphabetic~order):~
3726   b,~
3727   baseline,~
3728   c,~
3729   cell-space-bottom-limit,~
3730   cell-space-top-limit,~
3731   code-after,~
3732   code-for-first-col,~
3733   code-for-first-row,~
3734   code-for-last-col,~
3735   code-for-last-row,~
3736   columns-width,~
3737   create-extra-nodes,~
3738   create-medium-nodes,~
3739   create-large-nodes,~
3740   end-of-row,~
3741   extra-left-margin,~
3742   extra-right-margin,~
3743   first-col,~
3744   first-row,~
3745   hlines,~
3746   hvlines,~
3747   last-col,~
3748   last-row,~
3749   left-margin,~
3750   light-syntax,~
3751   name,~

```

```

3752 nullify-dots,~
3753 parallelize-diags,~
3754 renew-dots,~
3755 right-margin,~
3756 small,~
3757 t,~
3758 vlines,~
3759 xdots/color,~
3760 xdots/shorten~and~
3761 xdots/line-style.
3762 }

```

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).

```

3763 \@@_msg_new:nnn { Unknown~option~for~NiceMatrix }
3764 {
3765   The~option~'\tl_use:N\l_keys_key_str'~is~unknown~for~the~
3766   \@@_full_name_env:. \\\
3767   If~you~go~on,~it~will~be~ignored. \\\
3768   For~a~list~of~the~available~options,~type~H~<return>.
3769 }
3770 {
3771   The~available~options~are~(in~alphabetic~order):~
3772   b,~
3773   baseline,~
3774   c,~
3775   cell-space-bottom-limit,~
3776   cell-space-top-limit,~
3777   code-after,~
3778   code-for-first-col,~
3779   code-for-first-row,~
3780   code-for-last-col,~
3781   code-for-last-row,~
3782   columns-width,~
3783   create-extra-nodes,~
3784   create-medium-nodes,~
3785   create-large-nodes,~
3786   end-of-row,~
3787   extra-left-margin,~
3788   extra-right-margin,~
3789   first-col,~
3790   first-row,~
3791   hlines,~
3792   hvlines,~
3793   l~(=L),~
3794   last-col,~
3795   last-row,~
3796   left-margin,~
3797   light-syntax,~
3798   name,~
3799   nullify-dots,~
3800   parallelize-diags,~
3801   r~(=R),~
3802   renew-dots,~
3803   right-margin,~
3804   S,~
3805   small,~
3806   t,~
3807   vlines,~
3808   xdots/color,~
3809   xdots/shorten~and~
3810   xdots/line-style.
3811 }

```

```

3812 \@@_msg_new:nnn { Unknown~option~for~NiceTabular }
3813 {
3814   The~option~'\tl_use:N\l_keys_key_str'~is~unknown~for~the~environment~
3815   \{NiceTabular\}. \\\
3816   If~you~go~on,~it~will~be~ignored. \\\
3817   For~a~list~of~the~available~options,~type~H~<return>.
3818 }
3819 {
3820   The~available~options~are~(in~alphabetic~order):~
3821   b,~
3822   baseline,~
3823   c,~
3824   cell-space-bottom-limit,~
3825   cell-space-top-limit,~
3826   code-after,~
3827   code-for-first-col,~
3828   code-for-first-row,~
3829   code-for-last-col,~
3830   code-for-last-row,~
3831   columns-width,~
3832   create-extra-nodes,~
3833   create-medium-nodes,~
3834   create-large-nodes,~
3835   end-of-row,~
3836   extra-left-margin,~
3837   extra-right-margin,~
3838   first-col,~
3839   first-row,~
3840   hlines,~
3841   hvlines,~
3842   last-col,~
3843   last-row,~
3844   left-margin,~
3845   light-syntax,~
3846   name,~
3847   nullify-dots,~
3848   parallelize-diags,~
3849   renew-dots,~
3850   right-margin,~
3851   t,~
3852   vlines,~
3853   xdots/color,~
3854   xdots/shorten~and~
3855   xdots/line-style.
3856 }
3857 \@@_msg_new:nnn { Duplicate~name }
3858 {
3859   The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
3860   the~same~environment~name~twice.~You~can~go~on,~but,~
3861   maybe,~you~will~have~incorrect~results~especially~
3862   if~you~use~'columns-width=auto'.~If~you~don't~want~to~see~this~
3863   message~again,~use~the~option~'allow-duplicate-names'.\\
3864   For~a~list~of~the~names~already~used,~type~H~<return>. \\\
3865 }
3866 {
3867   The~names~already~defined~in~this~document~are:~
3868   \seq_use:Nnnn \g_@@_names_seq { ,~ } { ,~ } { ~and~ }.
3869 }
3870 \@@_msg_new:nn { Option~auto~for~columns~width }
3871 {
3872   You~can't~give~the~value~'auto'~to~the~option~'columns-width'~here.~
3873   If~you~go~on,~the~option~will~be~ignored.
3874 }

```

```

3875 \@@_msg_new:nn { Zero~row }
3876 {
3877   There~is~a~problem.~Maybe~you~have~used~l,~c~and~r~instead~of~L,~C~
3878   and~R~in~the~preamble~of~your~environment. \\\
3879   This~error~is~fatal.
3880 }

```

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.

```

3881 \file_input_stop:
3882 \NewDocumentEnvironment { pNiceArrayC } { }
3883 {
3884   \int_zero:N \l_@@_last_col_int
3885   \pNiceArray
3886 }
3887 { \endpNiceArray }
3888 \NewDocumentEnvironment { bNiceArrayC } { }
3889 {
3890   \int_zero:N \l_@@_last_col_int
3891   \bNiceArray
3892 }
3893 { \endbNiceArray }
3894 \NewDocumentEnvironment { BNiceArrayC } { }
3895 {
3896   \int_zero:N \l_@@_last_col_int
3897   \BNiceArray
3898 }
3899 { \endBNiceArray }
3900 \NewDocumentEnvironment { vNiceArrayC } { }
3901 {
3902   \int_zero:N \l_@@_last_col_int
3903   \vNiceArray
3904 }
3905 { \endvNiceArray }
3906 \NewDocumentEnvironment { VNiceArrayC } { }
3907 {
3908   \int_zero:N \l_@@_last_col_int
3909   \VNiceArray
3910 }
3911 { \endVNiceArray }
3912 \NewDocumentEnvironment { pNiceArrayRC } { }
3913 {
3914   \int_zero:N \l_@@_last_col_int
3915   \int_zero:N \l_@@_first_row_int
3916   \pNiceArray
3917 }
3918 { \endpNiceArray }
3919 \NewDocumentEnvironment { bNiceArrayRC } { }
3920 {
3921   \int_zero:N \l_@@_last_col_int
3922   \int_zero:N \l_@@_first_row_int
3923   \bNiceArray
3924 }
3925 { \endbNiceArray }
3926 \NewDocumentEnvironment { BNiceArrayRC } { }
3927 {
3928   \int_zero:N \l_@@_last_col_int
3929   \int_zero:N \l_@@_first_row_int
3930   \BNiceArray

```

```

3931 }
3932 { \endBNiceArray }
3933 \NewDocumentEnvironment { vNiceArrayRC } { }
3934 {
3935   \int_zero:N \l_@@_last_col_int
3936   \int_zero:N \l_@@_first_row_int
3937   \vNiceArray
3938 }
3939 { \endvNiceArray }
3940 \NewDocumentEnvironment { VNiceArrayRC } { }
3941 {
3942   \int_zero:N \l_@@_last_col_int
3943   \int_zero:N \l_@@_first_row_int
3944   \VNiceArray
3945 }
3946 { \endVNiceArray }
3947 \NewDocumentEnvironment { NiceArrayCwithDelims } { }
3948 {
3949   \int_zero:N \l_@@_last_col_int
3950   \NiceArrayWithDelims
3951 }
3952 { \endNiceArrayWithDelims }
3953 \NewDocumentEnvironment { NiceArrayRCwithDelims } { }
3954 {
3955   \int_zero:N \l_@@_last_col_int
3956   \int_zero:N \l_@@_first_row_int
3957   \NiceArrayWithDelims
3958 }
3959 { \endNiceArrayWithDelims }

```

17 History

Changes between versions 1.0 and 1.1

The dotted lines are no longer drawn with Tikz nodes but with Tikz circles (for efficiency).
Modification of the code which is now twice faster.

Changes between versions 1.1 and 1.2

New environment `{NiceArray}` with column types L, C and R.

Changes between version 1.2 and 1.3

New environment `{pNiceArrayC}` and its variants.

Correction of a bug in the definition of `{BNiceMatrix}`, `{vNiceMatrix}` and `{VNiceMatrix}` (in fact, it was a typo).

Options are now available locally in `{pNiceMatrix}` and its variants.

The names of the options are changed. The old names were names in “camel style”.

Changes between version 1.3 and 1.4

The column types `w` and `W` can now be used in the environments `{NiceArray}`, `{pNiceArrayC}` and its variants with the same meaning as in the package `array`.

New option `columns-width` to fix the same width for all the columns of the array.

Changes between version 1.4 and 2.0

The versions 1.0 to 1.4 of `nicematrix` were focused on the continuous dotted lines whereas the version 2.0 of `nicematrix` provides different features to improve the typesetting of mathematical matrices.

Changes between version 2.0 and 2.1

New implementation of the environment `{pNiceArrayRC}`. With this new implementation, there is no restriction on the width of the columns.

The package `nicematrix` no longer loads `mathtools` but only `amsmath`.

Creation of “medium nodes” and “large nodes”.

Changes between version 2.1 and 2.1.1

Small corrections: for example, the option `code-for-first-row` is now available in the command `\NiceMatrixOptions`.

Following a discussion on TeX StackExchange³⁷, Tikz externalization is now deactivated in the environments of the package `nicematrix`.³⁸

Changes between version 2.1.2 and 2.1.3

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

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

Changes between version 2.1.3 and 2.1.4

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

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

Changes between version 2.1.4 and 2.1.5

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

Option `allow-duplicate-names`.

Changes between version 2.1.5 and 2.2

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

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

³⁷cf. tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package

³⁸Before this version, there was an error when using `nicematrix` with Tikz externalization. In any case, it’s not possible to externalize the Tikz elements constructed by `nicematrix` because they use the options `overlay` and `remember picture`.

Changes between version 2.2 and 2.2.1

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

Changes between version 2.2.1 and 2.3

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

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 StackExchange³⁹, 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.

³⁹cf. tex.stackexchange.com/questions/510841/nicematrix-and-tikz-external-optimize

Changes between version 3.5 and 3.6

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

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

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

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

Changes between version 3.6 and 3.7

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

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

Changes between version 3.7 and 3.8

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

Changes between version 3.8 and 3.9

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

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

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.

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`).

Changes between versions 3.15 and 4.0

New environment `{NiceTabular}`

Commands to color cells, row and columns with a perfect result in the PDF.

Index

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

Symbols	
@@ commands:	
<code>\@@_Block:</code>	747, 3240
<code>\@@_Block_error:nn</code>	3255, 3257
<code>\@@_Block_i</code>	3241, 3242
<code>\@@_Block_ii:nnnnn</code>	3242, 3243
<code>\@@_Block_iii:nnnnnn</code>	3247, 3270
<code>\@@_Cdots</code>	738, 756, 2390
<code>\g_@@_Cdots_lines_tl</code>	827, 1719
<code>\@@_Cell:</code>	139, 487, 677, 769, 770, 771, 779, 793
<code>\@@_CodeAfter:n</code>	751, 3458, 3470
<code>\@@_CodeAfter_i:n</code>	3461, 3463
<code>\@@_Ddots</code>	740, 758, 2426
<code>\g_@@_Ddots_lines_tl</code>	830, 1717
<code>\g_@@_HVdotsfor_lines_tl</code>	832, 1715, 2507, 2583, 3530
<code>\@@_Hdotsfor:</code>	744, 761, 2496
<code>\@@_Hdotsfor:nnnn</code>	2509, 2521
<code>\@@_Hdotsfor_i</code>	2499, 2505
<code>\@@_Hspace:</code>	743, 2479
<code>\@@_Iddots</code>	741, 759, 2452
<code>\g_@@_Iddots_lines_tl</code>	831, 1718
<code>\@@_Ldots</code>	737, 755, 760, 2371
<code>\g_@@_Ldots_lines_tl</code>	828, 1720
<code>\l_@@_NiceArray_bool</code>	154, 857, 966, 984, 996, 1051, 1499, 2923, 2924, 3407
<code>\g_@@_NiceMatrixBlock_int</code>	150, 3031, 3036, 3039, 3050
<code>\l_@@_NiceTabular_bool</code>	155, 494, 553, 704, 842, 910, 912, 985, 997, 1005, 1122, 1126, 1421, 1438, 1463, 1479, 1578, 1928, 1938, 1972, 1980, 2970, 2977
<code>\@@_OnlyMainNiceMatrix:n</code>	749, 2876, 2895
<code>\@@_OnlyMainNiceMatrix_i:n</code>	2879, 2886, 2889
<code>\@@_Vdots</code>	739, 757, 2409
<code>\g_@@_Vdots_lines_tl</code>	829, 1716
<code>\@@_Vdotsfor:</code>	745, 2581
<code>\@@_Vdotsfor:nnnn</code>	2585, 2596
<code>\@@_actually_draw_Cdots:</code>	1961, 1965
<code>\@@_actually_draw_Ddots:</code>	2088, 2092
<code>\@@_actually_draw_Iddots:</code>	2139, 2143
<code>\@@_actually_draw_Ldots:</code>	1917, 1921, 2572
<code>\@@_actually_draw_Vdots:</code>	2015, 2019, 2647
<code>\@@_actually_slashbox:nnnn</code>	3425, 3429

\@@_adapt_S_column:	114, 129, 841	\@@_define_com:nnn	3391, 3399, 3400, 3401, 3402, 3403
\@@_after_array:	1144, 1582	\@@_define_env:n	1545, 1569, 1570, 1571, 1572, 1573, 1574
\@@_analyze_end:Nn	1224, 1269	\g_@@_delta_x_one_dim	1654, 2115, 2125
\l_@@_argspec_tl	2369,	\g_@@_delta_x_two_dim	1656, 2166, 2176
	2370, 2371, 2390, 2409, 2426, 2452, 2503,	\g_@@_delta_y_one_dim	1655, 2117, 2125
	2504, 2505, 2579, 2580, 2581, 2672, 2673, 2674	\g_@@_delta_y_two_dim	1657, 2168, 2176
\@@_array:	606, 1225, 1252	\@@_dotfill	3411, 3414, 3420
\l_@@_auto_columns_width_bool	269, 364, 1333, 1337, 3026	\@@_dotfill:	750, 3412
\l_@@_baseline_str	261, 262, 357, 358, 359, 360, 617,	\@@_dotfill_i:	3415, 3417
	1053, 1069, 1072, 1073, 1074, 1150, 1156, 1161	\@@_dotfill_ii:	3417, 3418
\@@_begin_of_NiceMatrix:nn	1552, 1556	\@@_dotfill_iii:	3418, 3419
\@@_begin_of_row:	491, 515, 1419	\@@_double_int_eval:n	2668, 2682, 2683
\l_@@_block_auto_columns_width_bool	854, 1338, 3019, 3024, 3034, 3044	\g_@@_dp_ante_last_row_dim	518, 722
\c_@@_booktabs_loaded_bool	29, 34, 687	\g_@@_dp_last_row_dim	518, 519, 725, 726, 922, 923, 1110, 2913
\l_@@_cell_box	493,	\g_@@_dp_row_zero_dim	538, 539, 716, 717, 1064, 1093, 1103
	539, 541, 547, 555, 556, 557, 558, 560, 563,	\@@_draw_Cdots:nnn	1947
	565, 567, 584, 689, 778, 786, 792, 801, 921,	\@@_draw_Ddots:nnn	2080
	923, 1420, 1442, 1445, 1447, 1462, 1483,	\@@_draw_Iddots:nnn	2131
	1487, 2657, 2660, 2664, 3281, 3330, 3335, 3420	\@@_draw_Ldots:nnn	1903
\l_@@_cell_space_bottom_limit_dim	250, 304, 558	\@@_draw_Vdots:nnn	2001
\l_@@_cell_space_top_limit_dim	249, 302, 556	\@@_draw_dotted_lines:	1678, 1704
\@@_cellcolor:nn	904, 2790, 2870, 2871	\@@_draw_dotted_lines_i:	1707, 1711
\g_@@_cells_seq	1263, 1264, 1265, 1267	\@@_draw_line:	1945,
\@@_chessboardcolors:nn	909, 2863		1999, 2078, 2129, 2180, 2184, 2721, 2987, 3017
\g_@@_code_after_tl	173, 377, 1246, 1696, 1697, 3460, 3469	\@@_draw_line_ii:nn	2701, 2705
\l_@@_code_before_bool	178, 354, 624,	\@@_draw_line_iii:nn	2708, 2712
	861, 1283, 1297, 1315, 1346, 1372, 1395, 1628	\@@_draw_non_standard_dotted_line:	2190, 2192
\l_@@_code_before_tl	353, 911	\@@_draw_non_standard_dotted_line:n	2195, 2198
\l_@@_code_for_first_col_tl	316, 1431	\@@_draw_standard_dotted_line:	2189, 2218
\l_@@_code_for_first_row_tl	320, 503	\@@_draw_standard_dotted_line_i:	2283, 2287
\l_@@_code_for_last_col_tl	318, 1471	\@@_draw_vlines:	1679, 2896
\l_@@_code_for_last_row_tl	322, 510	\g_@@_empty_cell_bool	185,
\g_@@_col_total_int	492, 767, 1042, 1365, 1366, 1397,		562, 569, 2388, 2407, 2424, 2450, 2476, 2481
	1402, 1407, 1408, 1461, 1586, 1589, 1594,	\@@_end_Cell:	141, 551, 682, 769, 770, 771, 783, 797
	1601, 1645, 2052, 2642, 3067, 3077, 3111, 3198	\l_@@_end_of_row_tl	277, 278, 310, 1248, 1249, 3600
\c_@@_colortbl_loaded_bool	84, 88, 706	\c_@@_endpgfortikzpicture_tl	40, 44, 1708, 2709, 2956
\@@_columncolor:nn	908, 2757	\@@_env:	147, 149, 524, 530, 585,
\l_@@_columns_width_dim	151, 365, 427, 1334, 1340, 3032, 3038		591, 629, 635, 641, 875, 876, 882, 883, 890,
\g_@@_com_or_env_str	165, 166, 169		891, 901, 1284, 1287, 1289, 1302, 1308,
\@@_computations_for_large_nodes:	3138, 3151, 3156		1311, 1320, 1326, 1329, 1351, 1357, 1360,
\@@_computations_for_medium_nodes:	3058, 3127, 3137, 3148		1377, 1383, 1389, 1397, 1402, 1408, 1690,
\@@_convert_to_str_seq:N	3496, 3508		1785, 1853, 1885, 1896, 2535, 2553, 2610,
\@@_create_col_nodes:	1228, 1256, 1275		2628, 2694, 2696, 2715, 2718, 3080, 3082,
\@@_create_large_nodes:	1665, 3132		3090, 3201, 3210, 3228, 3314, 3320, 3321, 3322
\@@_create_medium_and_large_nodes:	1662, 3143	\g_@@_env_int	146, 147, 853,
\@@_create_medium_nodes:	1663, 3122		863, 867, 870, 879, 880, 887, 888, 931, 934,
\@@_create_nodes:	3129, 3140, 3150, 3153, 3194		949, 952, 1593, 1614, 1632, 1635, 2183, 3237
\@@_create_row_node:	620, 652, 688	\@@_error:n	20, 289, 298, 415, 426, 435, 438,
\@@_cut_on_hyphen:w	2723, 2741, 2742, 2772, 2773, 2799, 2828, 2836		458, 461, 468, 470, 476, 481, 485, 1037,
\g_@@_ddots_int	1652, 2112, 2113		1081, 1168, 2374, 2393, 2412, 2429, 2455, 3259
\@@_define_columntype:nn	672, 772, 773, 774	\@@_error:nn	21,
			372, 2377, 2380, 2396, 2399, 2414, 2417,
			2433, 2434, 2439, 2440, 2459, 2460, 2465, 2466
		\@@_error:nnn	22, 2699

```

\\_error_too_much_cols: ..... 1003, 3515
\\_everycr: ..... 646, 711, 714
\\_everycr_i: ..... 646, 647
\\_@@_exterior_arraycolsep_bool .....
..... 263, 423, 987, 999
\\_@@_extra_left_margin_dim .....
..... 275, 341, 1016, 1450
\\_@@_extra_right_margin_dim .....
..... 276, 342, 1028, 1491, 2055
\\_@@_extract_coords_values: ..... 3219, 3226
\\_@@_fatal:n ..... 23, 159, 845, 1233,
1237, 1239, 1272, 1278, 3489, 3520, 3523, 3526
\\_@@_fatal:nn ..... 24
\\_@@_final_i_int 1668, 1732, 1737, 1740,
1765, 1773, 1777, 1786, 1794, 1897, 1939,
2037, 2104, 2155, 2526, 2554, 2622, 2632, 2634
\\_@@_final_j_int .....
..... 1669, 1733, 1738, 1745, 1750, 1756,
1766, 1774, 1778, 1787, 1795, 1898, 1935,
1977, 2106, 2157, 2547, 2557, 2559, 2601, 2630
\\_@@_final_open_bool ..... 1671, 1739,
1743, 1746, 1753, 1759, 1763, 1779, 1933,
1975, 1985, 1996, 2022, 2035, 2043, 2064,
2102, 2153, 2291, 2306, 2337, 2338, 2524,
2548, 2560, 2599, 2623, 2635, 2691, 2953, 2992
\\_@@_find_extremities_of_line:nnnn ...
..... 1727, 1907, 1951, 2006, 2084, 2135
\\_@@_first_col_int .. 191, 192, 312, 474,
491, 979, 1046, 1279, 1295, 1639, 2878,
3067, 3077, 3111, 3159, 3198, 3373, 3379, 3385
\\_@@_first_row_int ..... 189,
190, 313, 478, 765, 1061, 1077, 1090, 1101,
1164, 1637, 3060, 3074, 3101, 3158, 3196,
3371, 3614, 3915, 3922, 3929, 3936, 3943, 3956
\\_@@_full_name_env: ..... 167, 3536,
3543, 3551, 3577, 3624, 3633, 3654, 3666, 3766
\\_@@_hdottedline: ..... 742, 2938
\\_@@_hdottedline:n ..... 2946, 2950
\\_@@_hdottedline_i: ..... 2941, 2943
\\_@@_hdottedline_i:n ..... 2955, 2959
\\_@@_hlines_bool ..... 266, 324, 653
\\_g_@@_ht_last_row_dim .....
..... 520, 723, 724, 920, 921, 1109, 2914
\\_g_@@_ht_row_one_dim .... 546, 547, 720, 721
\\_g_@@_ht_row_zero_dim .....
..... 540, 541, 718, 719, 1064, 1093, 1104
\\_@@_i: ..... 3060, 3062,
3063, 3064, 3065, 3074, 3080, 3082, 3083,
3084, 3085, 3090, 3091, 3092, 3093, 3101,
3104, 3106, 3107, 3108, 3160, 3162, 3165,
3166, 3170, 3171, 3196, 3201, 3203, 3205,
3209, 3210, 3221, 3228, 3230, 3232, 3236, 3237
\\_g_@@_iddots_int ..... 1653, 2163, 2164
\\_@@_in_env_bool ..... 153, 845, 846
\\_@@_initial_i_int .....
1666, 1730, 1805, 1808, 1833, 1841, 1845,
1854, 1862, 1886, 1929, 1987, 1989, 2031,
2096, 2147, 2525, 2526, 2536, 2604, 2614, 2616
\\_@@_initial_j_int ... 1667, 1731, 1806,
1813, 1818, 1824, 1834, 1842, 1846, 1855,
1863, 1887, 1925, 1969, 2045, 2047, 2052,
2098, 2149, 2529, 2539, 2541, 2600, 2601, 2612
\\_@@_initial_open_bool .....
..... 1670, 1807, 1811, 1814, 1821, 1827,
1831, 1847, 1923, 1967, 1984, 1994, 2022,
2029, 2041, 2094, 2145, 2289, 2336, 2523,
2530, 2542, 2598, 2605, 2617, 2690, 2952, 2991
\\_@@_instruction_of_type:nn .....
..... 595, 2382, 2401, 2419, 2443, 2469
\\_@@_inter_dots_dim .....
..... 251, 252, 1675, 2294, 2301, 2312, 2320,
2327, 2332, 2344, 2352, 2982, 2985, 3013, 3015
\\_g_@@_internal_code_after_tl .....
..... 174, 818, 1680, 1681, 2945, 3245, 3423
\\_@@_j: ..... 3067, 3069,
3070, 3071, 3072, 3077, 3080, 3082, 3085,
3087, 3088, 3090, 3093, 3095, 3096, 3111,
3114, 3116, 3117, 3118, 3173, 3175, 3178,
3180, 3184, 3185, 3198, 3201, 3202, 3204,
3209, 3210, 3222, 3228, 3229, 3231, 3236, 3237
\\_@@_l_dim .....
..... 2267, 2268, 2281, 2282, 2294, 2300,
2311, 2319, 2327, 2332, 2344, 2345, 2352, 2353
\\_@@_large_nodes_bool 272, 332, 1661, 1665
\\_g_@@_last_col_found_bool ..... 199,
826, 1043, 1139, 1364, 1393, 1459, 1585, 1642
\\_@@_last_col_int ..... 197, 198,
416, 449, 451, 469, 477, 873, 945, 951, 958,
991, 1562, 1564, 1586, 1589, 1641, 2011,
2050, 2379, 2398, 2440, 2466, 3375, 3381,
3387, 3519, 3537, 3884, 3890, 3896, 3902,
3908, 3914, 3921, 3928, 3935, 3942, 3949, 3955
\\_@@_last_col_without_value_bool ...
..... 196, 448, 1587, 3522
\\_@@_last_row_int .. 193, 194, 314, 508,
657, 812, 871, 916, 926, 933, 940, 1031,
1035, 1038, 1045, 1107, 1250, 1251, 1427,
1428, 1468, 1469, 1608, 1912, 1956, 2416,
2434, 2460, 2658, 2884, 2892, 2911, 3383, 3653
\\_@@_last_row_without_value_bool ...
..... 195, 928, 1033, 1606
\\_g_@@_last_vdotted_col_int 815, 817, 823, 825
\\_@@_left_delim_dim 964, 968, 973, 1215, 1448
\\_@@_left_delim_tl ..... 836, 2981
\\_@@_left_margin_dim .....
..... 273, 335, 1015, 1449, 2971, 3189
\\_@@_letter_for_dotted_lines_str ...
..... 434, 440, 441, 805, 806
\\_@@_light_syntax_bool .....
..... 260, 308, 1018, 1023, 1609
\\_@@_light_syntax_i ..... 1241, 1244
\\_@@_line ..... 1694, 2674
\\_@@_line_i:nn ..... 2681, 2688
\\_@@_line_with_light_syntax:n ... 1255, 1259
\\_@@_line_with_light_syntax_i:n .....
..... 1254, 1260, 1261
\\_g_@@_max_cell_width_dim .....
..... 559, 560, 855, 1339, 3025, 3051
\\_@@_max_delimiter_width_bool .....
..... 280, 307, 1135
\\_c_@@_max_l_dim ..... 2281, 2286
\\_@@_medium_nodes_bool 271, 331, 1659, 3317
\\_@@_message_hdotsfor: 3528, 3536, 3543, 3551
\\_@@_msg_new:nn .....
..... 25, 3473, 3533, 3540, 3548, 3555,

```

3560, 3565, 3570, 3575, 3581, 3586, 3592, 3597, 3603, 3611, 3618, 3623, 3625, 3631, 3638, 3645, 3651, 3659, 3664, 3670, 3870, 3875	2917, 2926, 2930, 2962, 2966, 2973, 3009, 3012, 3014, 3106, 3116, 3285, 3287, 3289, 3291, 3326, 3327, 3333, 3434, 3436, 3439, 3441
\@@_msg_new:nnn 26, 3675, 3717, 3763, 3812, 3857	\l_@@_radius_dim 255, 256, 813, 1674, 1943, 1944, 2361, 2940, 2964, 3010, 3011
\@@_msg_redirect_name:nn 27, 429, 3491	\l_@@_real_left_delim_dim 1186, 1201, 1216
\@@_multicolumn:nnn 746, 2485	\l_@@_real_right_delim_dim 1187, 1213, 1219
\g_@@_multicolumn_cells_seq 763, 2490, 3085, 3093, 3215	\@@_rectanglecolor:nnn 905, 2821
\g_@@_multicolumn_sizes_seq 764, 2492, 3216	\@@_renew_NC@rewrite@S: 132, 824
\g_@@_name_env_str 164, 170, 171, 839, 840, 1271, 1500, 1501, 1507, 1508, 1515, 1516, 1523, 1524, 1531, 1532, 1539, 1540, 1549, 1699, 3395, 3517	\l_@@_renew_dots_bool 328, 753, 3483
\l_@@_name_str 270, 374, 526, 529, 587, 590, 637, 640, 929, 938, 941, 947, 956, 959, 1288, 1289, 1310, 1311, 1328, 1329, 1359, 1360, 1385, 1388, 1404, 1407, 1596, 1600, 1617, 1621, 3206, 3209, 3233, 3236	\@@_renew_matrix: 419, 3341, 3485
\g_@@_names_seq 152, 371, 373, 3868	\@@_restore_iRow_jCol: 1700, 1722
\l_@@_nb_cols_int 3362, 3367, 3370, 3374, 3380, 3386	\c_@@_revtex_bool 47, 49, 52, 608
\l_@@_nb_rows_int 3361, 3366, 3377	\l_@@_right_delim_dim 965, 969, 975, 1218, 1489
\@@_newcolumnntype 663, 674, 769, 770, 771, 775, 789	\l_@@_right_delim_tl 837, 2984
\@@_node_for_multicolumn:nn 3217, 3224	\l_@@_right_margin_dim 274, 337, 1027, 1490, 2054, 2978, 3192
\@@_node_for_the_cell: 566, 571, 1446, 1492	\@@_rotate: 748, 2652
\@@_node_position: 882, 884, 890, 892	\@@_rotate_i: 2652, 2653
\l_@@_nullify_dots_bool 268, 330, 2387, 2406, 2423, 2449, 2475	\@@_rotate_ii: 2653, 2654
\@@_old_arraycolsep_dim 186, 705, 1007	\@@_rotate_iii: 2654, 2655
\@@_old_cdots 731, 2406	\g_@@_row_of_col_done_bool 177, 650, 838, 1294
\@@_old_ddots 733, 2449	\g_@@_row_total_int 766, 1044, 1078, 1165, 1608, 1615, 1622, 1638, 2567, 2917, 3060, 3074, 3101, 3196, 3615
\l_@@_old_iRow_int 175, 691, 1724	\@@_rowcolor:nn 906, 2728, 2859, 2860
\@@_old_ialign: 619, 727	\@@_rowcolors:nnn 907, 2854
\@@_old_iddots 734, 2475	\g_@@_rows_seq . 1247, 1249, 1251, 1253, 1255
\l_@@_old_jCol_int 176, 694, 1725	\@@_set_final_coords: 1876, 1901
\@@_old_ldots 730, 2387	\@@_set_final_coords_from_anchor:n 1892, 1942, 1982, 2025, 2040, 2109, 2160
\@@_old_multicolumn 2484, 2487	\@@_set_initial_coords: 1871, 1890
\@@_old_pgftutil@check@rerun 76, 81	\@@_set_initial_coords_from_anchor:n 1881, 1932, 1974, 2024, 2034, 2101, 2152
\@@_old_vdots 732, 2423	\@@_set_seq_of_str_from_clist:Nn 3505, 3510
\l_@@_parallelize_diags_bool 264, 265, 327, 1650, 2110, 2161	\@@_set_size:n 3359, 3368
\@@_pgf_rect_node:nnn 225, 3319	\c_@@_siunitx_loaded_bool 107, 111, 116, 456, 824
\@@_pgf_rect_node:nnnn 200, 3200, 3227, 3313	\@@_slashbox:nn 752, 3421
\c_@@_pgfortikzpicture_tl 39, 43, 1706, 2707, 2954	\l_@@_small_bool 417, 459, 465, 479, 497, 697, 1422, 1464, 1672
\@@_picture_position: 876, 884, 892	\c_@@_standard_tl 258, 259, 2188, 2986, 3016
\@@_pre_array: 685, 963	\l_@@_stop_loop_bool 1734, 1735, 1767, 1780, 1789, 1802, 1803, 1835, 1848, 1857
\c_@@_preamble_first_col_tl 980, 1415	\@@_succ:n 105, 629, 635, 1174, 1377, 1383, 1388, 1389, 1397, 1402, 1407, 1408, 1643, 1935, 1977, 1989, 2037, 2047, 2104, 2106, 2149, 2155, 2735, 2748, 2766, 2781, 2806, 2810, 2842, 2844, 2907, 2915, 2917, 2924, 2973, 3014, 3166, 3170, 3180, 3184, 3289, 3291, 3439, 3441
\c_@@_preamble_last_col_tl 992, 1455	\l_@@_suffix_tl 3128, 3139, 3149, 3152, 3201, 3209, 3210, 3228, 3236, 3237
\@@_pred:n 106, 1564	\c_@@_table_collect_begin_tl . 124, 126, 139
\@@_put_box_in_flow: 1137, 1146, 1217	\c_@@_table_print_tl 127, 128, 141
\@@_put_box_in_flow_bis:nn 1136, 1184	\@@_test_if_math_mode: 156, 844, 1509, 1517, 1525, 1533, 1541
\@@_put_box_in_flow_i: 1152, 1154	\l_@@_the_array_box 977, 1004, 1066, 1070, 1096, 1125
\@@_qpoint:n 148, 1056, 1058, 1085, 1087, 1172, 1174, 1177, 1925, 1929, 1935, 1939, 1969, 1977, 1987, 1989, 2031, 2037, 2045, 2047, 2096, 2098, 2104, 2106, 2147, 2149, 2155, 2157, 2715, 2718, 2735, 2748, 2750, 2764, 2766, 2779, 2781, 2800, 2806, 2808, 2810, 2833, 2835, 2842, 2844, 2904, 2907, 2915,	\c_@@_tikz_loaded_bool 30, 38, 896, 1682, 2993
	\l_@@_tikz_tl 3264, 3312

`\l_@@_type_of_col_tl` 452, 453, 454, 455, 457, 1550, 1552
`\c_@@_types_of_matrix_seq` 3510, 3517
`\@@_update_for_first_and_last_row:` ..
..... 534, 561, 918, 1440, 1481
`\@@_vdottedline:n` 819, 2989
`\@@_vdottedline_i:n` 2996, 3001, 3005
`\@@_vline:` 856, 2894
`\l_@@_vlines_bool`
..... 267, 325, 986, 998, 1010, 1029, 1679
`\l_@@_white_bool` 3266, 3293
`\g_@@_width_first_col_dim`
..... 188, 1049, 1441, 1442
`\g_@@_width_last_col_dim`
..... 187, 1141, 1482, 1483
`\l_@@_x_final_dim` 181,
1878, 1936, 1937, 1978, 1979, 2027, 2049,
2057, 2061, 2065, 2067, 2072, 2074, 2107,
2116, 2124, 2158, 2167, 2175, 2215, 2229,
2238, 2274, 2326, 2342, 2719, 2974, 2985, 3011
`\l_@@_x_initial_dim`
..... 179, 1873, 1926, 1927, 1970, 1971,
2027, 2048, 2049, 2056, 2061, 2065, 2067,
2069, 2072, 2074, 2099, 2116, 2124, 2150,
2167, 2175, 2206, 2228, 2238, 2274, 2326,
2340, 2342, 2360, 2362, 2716, 2967, 2982, 3010
`\l_@@_xdots_color_tl` 279, 292, 1916,
1960, 2003, 2087, 2138, 2196, 2571, 2646, 2678
`\l_@@_xdots_down_tl` ... 296, 2212, 2222, 2257
`\l_@@_xdots_line_style_tl`
..... 257, 259, 288, 2188, 2196, 2986, 3016
`\l_@@_xdots_shorten_dim` 253,
254, 294, 1676, 2203, 2204, 2300, 2311, 2319
`\l_@@_xdots_up_tl` 297, 2208, 2221, 2247
`\l_@@_y_final_dim` 182,
1879, 1940, 1944, 1991, 1995, 1997, 2038,
2105, 2118, 2121, 2156, 2169, 2172, 2215,
2229, 2237, 2276, 2331, 2350, 2720, 2965, 3015
`\l_@@_y_initial_dim`
180, 1874, 1930, 1943, 1990, 1991, 1995,
1997, 2032, 2097, 2118, 2123, 2148, 2169,
2174, 2206, 2228, 2237, 2276, 2331, 2348,
2350, 2360, 2363, 2717, 2963, 2964, 2965, 3013
`\l` 1238, 1260, 3375,
3381, 3387, 3476, 3557, 3562, 3567, 3572,
3578, 3599, 3615, 3620, 3628, 3634, 3641,
3648, 3661, 3667, 3672, 3678, 3679, 3720,
3721, 3766, 3767, 3815, 3816, 3863, 3864, 3878
`\{` 171, 1526, 3403, 3627, 3661, 3720, 3815
`\}` 171, 1526, 3403, 3627, 3661, 3720, 3815
`\|` 1542, 3402

`\sqcup` 3531, 3536, 3543, 3551,
3606, 3614, 3615, 3620, 3624, 3654, 3655, 3666

A

`\array` 616
`\arraycolsep` 336, 338, 340, 700, 705,
968, 969, 1007, 1008, 1012, 1048, 1124,
1128, 1142, 1928, 1938, 1972, 1980, 2970, 2977
`\arrayrulecolor` 91
`\arrayrulewidth` 525, 628, 630, 636, 658,
1012, 1013, 1029, 1301, 1303, 1309, 1319,

1321, 1327, 1350, 1352, 1358, 1376, 1378,
1384, 2903, 2921, 3051, 3300, 3301, 3305, 3306
`\arraystretch` 699
`\AtBeginDocument` 31, 68, 85,
108, 1702, 2367, 2501, 2577, 2670, 2703, 2948
`\AutoNiceMatrix` 3404
`\AutoNiceMatrixWithDelims` ... 3364, 3396, 3408

B

`\Block` 747, 3255, 3260, 3620
`\BNiceArray` 3897, 3930
`\bNiceArray` 3891, 3923
`\BNiceMatrix` 3356
`\bNiceMatrix` 3353
bool commands:
`\bool_do_until:Nn` 1735, 1803
`\bool_gset_false:N` 569, 826, 838
`\bool_gset_true:N`
1294, 1459, 2388, 2407, 2424, 2450, 2476, 2481
`\bool_if:NTF` 116,
494, 497, 553, 624, 650, 653, 687, 697,
704, 753, 824, 845, 854, 861, 896, 910, 912,
1005, 1010, 1029, 1033, 1139, 1283, 1297,
1315, 1333, 1346, 1372, 1393, 1395, 1421,
1422, 1438, 1463, 1464, 1479, 1585, 1587,
1606, 1609, 1628, 1650, 1665, 1672, 1679,
1682, 1994, 1996, 2110, 2161, 2387, 2406,
2423, 2449, 2475, 3034, 3044, 3293, 3317, 3522
`\bool_if:nTF` 1043, 1075, 1162,
1659, 2373, 2392, 2411, 2428, 2454, 2692, 3274
`\bool_lazy_all:nTF` 982, 994
`\bool_lazy_and:nnTF`
... 1336, 1423, 1640, 1983, 2220, 2801, 2829
`\bool_lazy_or:nnTF` ... 285, 1467, 2022, 2280
`\bool_lazy_or_p:nn` 1426
`\bool_not_p:n`
... 985, 986, 987, 997, 998, 999, 1338, 1642
`\bool_set:Nn` 2026
box commands:
`\box_clear_new:N` 689, 977
`\box_dp:N` 519,
539, 558, 717, 726, 923, 1149, 1195, 1208
`\box_ht:N` 520, 541, 547,
556, 719, 721, 724, 921, 1148, 1195, 1208, 2663
`\box_move_up:nn` . 59, 61, 63, 1066, 1095, 1181
`\box_rotate:Nn` 2657
`\box_set_dp:Nn` 557, 1149
`\box_set_ht:Nn` 555, 1148
`\box_use:N` 679, 682, 2664
`\box_use_drop:N`
... 563, 567, 584, 786, 801, 1066, 1070,
1096, 1125, 1151, 1181, 1182, 1447, 3330, 3335
`\box_wd:N` 560, 565, 973,
975, 1202, 1214, 1442, 1445, 1483, 1487, 3420
`\l_tmpa_box` 972, 973, 974, 975,
1113, 1148, 1149, 1151, 1181, 1182, 1195, 1208
`\l_tmpb_box` 1188, 1202, 1203, 1214

C

`\Cdots` 738, 2396, 2399, 3605
`\cdots` 731, 756
`\cellcolor` 904
`\chessboardcolors` 909

I	
\ialign	619, 702, 727
\Iddots	741, 2459, 2460, 2465, 2466, 3606
\iddots	54, 734, 759
if commands:	
\if_mode_math:	158
\ifnum	95
\ifstandalone	850
int commands:	
\int_case:nnTF	2431, 2437, 2457, 2463
\int_compare:nNnTF	
.. 490, 491, 501, 508, 544, 655, 657, 810,	
812, 815, 871, 873, 916, 926, 945, 1031,	
1035, 1045, 1046, 1061, 1090, 1250, 1278,	
1279, 1465, 1745, 1752, 1756, 1758, 1813,	
1820, 1824, 1826, 1912, 1956, 2011, 2050,	
2052, 2488, 2567, 2642, 2658, 2746, 2777,	
2837, 2839, 2883, 2884, 2891, 2892, 2911,	
3371, 3373, 3375, 3379, 3381, 3383, 3385, 3387	
\int_compare_p:n	2802, 2803, 2830, 2831
\int_gadd:Nn	2494
\int_gincr:N	
. 489, 517, 853, 1370, 1460, 2112, 2163, 3031	
\int_if_even:nTF	2869
\int_step_inline:nn	2865, 2867
iow commands:	
\iow_now:Nn	71, 1630, 1631, 1633, 1648
\iow_shipout:Nn	1590, 1591, 1598,
1604, 1611, 1612, 1619, 1625, 3046, 3047, 3053	
K	
\kern	64
keys commands:	
\keys_define:nn	281, 300, 347,
380, 412, 444, 463, 472, 483, 3020, 3262, 3481	
\l_keys_key_str	
.... 15, 3589, 3672, 3677, 3719, 3765, 3814	
\keys_set:nn	306,
443, 858, 859, 1551, 1577, 1915, 1959,	
2014, 2086, 2137, 2570, 2645, 2677, 3033, 3273	
\l_keys_value_tl	3636, 3643, 3859
L	
\lasthline	736
\Ldots	737, 2377, 2380, 3605
\ldots	730, 755
\left	1116, 1191, 1206
legacy commands:	
\legacy_if:nTF	368
\let	96, 97, 98
\line	1694, 3627
M	
\makebox	785, 800
\mathinner	56
mode commands:	
\mode_leave_vertical:	679, 843
msg commands:	
\msg_error:nn	16, 20
\msg_error:nnn	21
\msg_error:nnnn	22, 3279
\msg_fatal:nn	23
\msg_fatal:nnn	24
\msg_new:nnn	9, 25

\msg_new:nnnn	26
\msg_redirect_name:nnn	28
\multicolumn	746, 2484, 2498, 2518
\myfiledate	6
\myfileversion	7
N	
\newcolumnntype	806
\NewDocumentCommand 91, 442, 2371, 2390, 2409,	
2426, 2452, 2505, 2581, 2674, 3240, 3364, 3404	
\NewDocumentEnvironment	834, 1221,
1231, 1497, 1505, 1513, 1521, 1529, 1537,	
1547, 1575, 3029, 3882, 3888, 3894, 3900,	
3906, 3912, 3919, 3926, 3933, 3940, 3947, 3953	
\NewExpandableDocumentCommand	2182
\NiceArray	1579
\NiceArrayWithDelims	
1502, 1510, 1518, 1526, 1534, 1542, 3950, 3957	
\NiceMatrixLastEnv	2182
\NiceMatrixOptions	442, 3678
\NiceMatrixoptions	3640
\noalign	95, 646, 710, 2940
\normalbaselines	696
\nulldelimiterspace	1202, 1214
\numexpr	105, 106
O	
\omit	1281, 1293, 1369
\OnlyMainNiceMatrix	749, 2875
P	
peek commands:	
\peek_meaning:NnTF	668
\peek_meaning_ignore_spaces:NnTF	1223
\pgfextracty	3326
\pgfgetlastxy	236
\pgfpathcircle	2359
\pgfpathlineto	2927, 3443
\pgfpathmoveto	2926, 3438
\pgfpathrectanglecorners	
..... 2749, 2782, 2812, 2846, 3297	
\pgfpointhead	234, 2929
\pgfpointanchor	
.... 149, 1883, 1894, 3082, 3090, 3321, 3322	
\pgfpointdiff	235, 884, 892
\pgfpointlineatime	2227
\pgfpointorigin	1287, 1403
\pgfpointscale	234
\pgfpointshapeborder	2715, 2718
\pgfrememberpicturepositiononpagetrue ...	
..... 522, 575, 634, 1286, 1307, 1325,	
1356, 1382, 1401, 1713, 2186, 2264, 2714,	
2901, 2961, 3008, 3125, 3135, 3146, 3283, 3433	
\pgfscope	2224, 3450
\pgfset	203, 228, 576, 898, 3312, 3449
\pgfsetbaseline	574
\pgfsetfillcolor ..	2734, 2763, 2796, 2827, 3296
\pgfsetlinewidth	2903
\pgfsetroundcap	3446
\pgftransformrotate	2231
\pgftransformshift	
.... 209, 234, 2225, 3328, 3333, 3451, 3454	
\pgfusepath	2251, 2261

`\pgfusepathqfill`
 2365, 2753, 2786, 2817, 2849, 2906, 2909, 3308
`\pgfusepathqstroke` 2934, 3447
`\phantom` 2387, 2406, 2423, 2449, 2475
`\pNiceArray` 3885, 3916
`\pNiceMatrix` 3344
 prg commands:
 `\prg_do_nothing:` 120, 129, 710, 1695
 `\prg_replicate:nn`
 ... 1365, 1366, 2518, 3374, 3377, 3380, 3386
`\ProcessKeysOptions` 3495
`\ProvideDocumentCommand` 54
`\ProvidesExplPackage` 4

Q

quark commands:
 `\q_stop` 1241, 1244, 2668, 2682,
 2683, 2723, 2741, 2742, 2772, 2773, 2799,
 2828, 2836, 3219, 3226, 3241, 3242, 3359, 3368

R

`\rectanglecolor` 905
`\relax` 105, 106
`\renewcommand` 134
`\RenewDocumentEnvironment`
 3343, 3346, 3349, 3352, 3355
`\RequirePackage` 1, 3, 17, 18, 19
`\right` 1132, 1198, 1210
`\rotate` 748
`\rowcolor` 906
`\rowcolors` 907

S

`\scriptstyle`
 ... 497, 1422, 1464, 2208, 2212, 2247, 2257
 seq commands:
 `\seq_clear:N` 3498
 `\seq_clear_new:N` 1632
 `\seq_count:N` 1251
 `\seq_gclear_new:N` 763, 764, 1247, 1263
 `\seq_gpop_left:NN` 1253, 1265
 `\seq_gput_left:Nn` 373, 2490, 2492
 `\seq_gset_from_clist:Nn` 1635
 `\seq_gset_split:Nnn` 1249, 1264
 `\seq_if_exist:NTF` 863
 `\seq_if_in:NnTF` 371, 3085, 3093, 3517
 `\seq_item:Nn` 867, 870, 879, 880, 887, 888
 `\seq_map_function:NN` 1255
 `\seq_map_inline:Nn` 1267, 3499
 `\seq_mapthread_function:NNN` 3214
 `\seq_new:N` 152
 `\seq_put_left:Nn` 3501
 `\seq_set_eq:NN` 3503
 `\seq_set_from_clist:Nn` 3507
 `\seq_use:Nnnn` 3868
 `\l_tmpa_seq` 3498, 3501, 3503
 skip commands:
 `\skip_gadd:Nn` 1341
 `\skip_gset:Nn` 1332
 `\skip_gset_eq:NN` 1339, 1340
 `\skip_horizontal:N` 813, 1013, 1015, 1016,
 1027, 1028, 1029, 1048, 1049, 1123, 1124,
 1127, 1128, 1141, 1142, 1215, 1216, 1218,
 1219, 1282, 1301, 1303, 1319, 1321, 1343,

 1350, 1352, 1371, 1376, 1378, 1399, 1411,
 1448, 1449, 1450, 1452, 1484, 1489, 1490, 1491
`\skip_vertical:N` . 628, 630, 1119, 1130, 2940
`\skip_vertical:n` 2663
`\g_tmpa_skip` 1332, 1339, 1340, 1341, 1343, 1371
`\c_zero_skip` 715
`\slashbox` 752
`\space` 170, 171
 str commands:
 `\c_backslash_str` 170
 `\c_colon_str` 441
 `\str_case:nnTF` 1156
 `\str_gclear:N` 1699
 `\str_gset:Nn` 840,
 1501, 1508, 1516, 1524, 1532, 1540, 1549, 3395
 `\str_if_empty:NTF`
 526, 587, 637, 839, 929, 947, 1288,
 1310, 1328, 1359, 1385, 1404, 1500, 1507,
 1515, 1523, 1531, 1539, 1596, 1617, 3206, 3233
 `\str_if_eq:nnTF` 80, 169,
 363, 425, 617, 1053, 1069, 1072, 1150, 1271
 `\str_if_eq_p:nn` 287, 3466
 `\str_lowercase:n` 785, 800
 `\str_new:N` 164, 165, 261, 270, 440
 `\str_set:Nn` 79,
 166, 262, 357, 358, 359, 370, 434, 1073, 3465
 `\str_set_eq:NN` 374, 441
 `\l_tmpa_str`
 79, 80, 370, 371, 373, 374, 3465, 3466

T

`\tabcolsep` 1008,
 1123, 1127, 1928, 1938, 1972, 1980, 2970, 2977
`\tabskip` 715
 T_EX and L^AT_EX 2_ε commands:
 `\@BTnormal` 688
 `\@acol` 612
 `\@acoll` 610
 `\@acolr` 611
 `\@addtopreamble` 856
 `\@array@array` 614
 `\@arrayacol` 610, 611, 612
 `\@arrayrule` 856
 `\@arstrutbox` 519,
 520, 679, 682, 717, 719, 721, 724, 726, 2663
 `\@currenvir` 3465
 `\@halignto` 613
 `\@height` 98
 `\@ifclassloaded` 48, 51
 `\@ifnextchar` 768
 `\@ifpackageloaded` 33, 36, 70, 87, 110
 `\@mainaux` 71,
 1590, 1591, 1598, 1604, 1611, 1612, 1619,
 1625, 1630, 1631, 1633, 1648, 3046, 3047, 3053
 `\@temptokena` 119, 122, 136, 138
 `\@width` 98
 `\@xhline` 101
 `\bBigg@` 972, 974
 `\c@MaxMatrixCols` 1563, 3545
 `\col@sep` .. 1282, 1341, 1399, 1411, 1452, 1484
 `\CT@arc@` 90, 92, 99, 658, 2895, 2899, 3007, 3445
 `\CT@everycr` 708
 `\CT@row@color` 710
 `\NC@` 667

3.5	Customization of the dotted lines	5
4	The PGF/Tikz nodes created by nicematrix	6
5	The code-after	7
6	The environments <code>{NiceArray}</code> and <code>{NiceTabular}</code>	8
7	The vertical position of the arrays	9
8	The exterior rows and columns	9
9	The dotted lines to separate rows or columns	11
10	The width of the columns	12
11	Block matrices	13
12	The color of the rows and columns	14
13	Advanced features	15
13.1	Alignment option in NiceMatrix	15
13.2	The command <code>\rotate</code>	16
13.3	The option <code>small</code>	16
13.4	The counters <code>iRow</code> and <code>jCol</code>	17
13.5	The options <code>hlines</code> , <code>vlines</code> and <code>hvlines</code>	17
13.6	The option <code>light-syntax</code>	18
13.7	Use of the column type <code>S</code> of <code>siunitx</code>	18
14	Technical remarks	19
14.1	Definition of new column types	19
14.2	The names of the PGF nodes created by nicematrix	19
14.3	Diagonal lines	19
14.4	The “empty” cells	20
14.5	The option <code>exterior-arraycolsep</code>	20
14.6	A technical problem with the argument of <code>\\</code>	21
14.7	Obsolete environments which have been deleted	21
15	Examples	21
15.1	Dotted lines	21
15.2	Dotted lines which are no longer dotted	23
15.3	Width of the columns	24
15.4	How to highlight cells of the matrix	24
15.5	Direct use of the Tikz nodes	27
16	Implementation	28
17	History	119
	Index	123