

The package `nicematrix`*

F. Pantigny
 fpantigny@wanadoo.fr

February 7, 2020

Abstract

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

1 Presentation

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

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

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

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

A command `\NiceMatrixOptions` is provided to fix the options (the scope of the options fixed by this command is the current TeX group).

An example for the continuous dotted lines

For example, consider the following code which uses an environment `{pmatrix}` of `amsmath`.

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

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

This code composes the matrix A on the right.

Now, if we use the package `nicematrix` with the option **transparent**, the same code will give the result on the right.

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

*This document corresponds to the version 3.11 of `nicematrix`, at the date of 2020/02/07.

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

2 The environments of this extension

The extension `nicematrix` defines the following new environments.

<code>{NiceMatrix}</code>	<code>{NiceArray}</code>	<code>{pNiceArray}</code>
<code>{pNiceMatrix}</code>		<code>{bNiceArray}</code>
<code>{bNiceMatrix}</code>		<code>{BNiceArray}</code>
<code>{BNiceMatrix}</code>		<code>{vNiceArray}</code>
<code>{vNiceMatrix}</code>		<code>{VNiceArray}</code>
<code>{VNiceMatrix}</code>		<code>{NiceArrayWithDelims}</code>

By default, the environments `{NiceMatrix}`, `{pNiceMatrix}`, `{bNiceMatrix}`, `{BNiceMatrix}`, `{vNiceMatrix}` and `{VNiceMatrix}` behave almost exactly as the corresponding environments of `amsmath`: `{matrix}`, `{pmatrix}`, `{bmatrix}`, `{Bmatrix}`, `{vmatrix}` and `{Vmatrix}`.

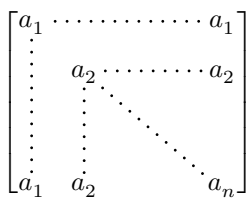
The environment `{NiceArray}` is similar to the environment `{array}` of the package `{array}`. However, for technical reasons, in the preamble of the environment `{NiceArray}`, the user must use the letters `L`, `C` and `R` instead of `l`, `c` and `r`. It's possible to use the constructions `w{...}{...}`, `W{...}{...}`², `l{...}`, `>{...}`, `<{...}`, `@{...}`, `!{...}` and `*{n}{...}` but the letters `p`, `m` and `b` should not be used. See p. 7 the section relating to `{NiceArray}`.

3 The continuous dotted lines

Inside the environments of the extension `nicematrix`, new commands are defined: `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, and `\Iddots`. These commands are intended to be used in place of `\dots`, `\cdots`, `\vdots`, `\ddots` and `\iddots`.³

Each of them must be used alone in the cell of the array and it draws a dotted line between the first non-empty cells⁴ on both sides of the current cell. Of course, for `\Ldots` and `\Cdots`, it's an horizontal line; for `\Vdots`, it's a vertical line and for `\Ddots` and `\Iddots` diagonal ones.

<code>\begin{bNiceMatrix}</code>		
<code>a_1</code>	<code>& \Cdots &</code>	<code>& & a_1 \\</code>
<code>\Vdots</code>	<code>& a_2 & \Cdots & & a_2 \\</code>	
	<code>& \Vdots & \Ddots \\</code>	
<code>\\</code>		
<code>a_1</code>	<code>& a_2 & & & a_n</code>	
<code>\end{bNiceMatrix}</code>		



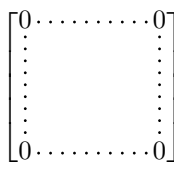
In order to represent the null matrix, one can use the following codage:

<code>\begin{bNiceMatrix}</code>		
<code>0</code>	<code>& \Cdots & 0</code>	<code>\\</code>
<code>\Vdots</code>	<code>& & \Vdots \\</code>	
<code>0</code>	<code>& \Cdots & 0</code>	
<code>\end{bNiceMatrix}</code>		



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

<code>\begin{bNiceMatrix}</code>		
<code>0</code>	<code>& \Cdots & \Cdots & 0</code>	<code>\\</code>
<code>\Vdots</code>	<code>& & & \Vdots \\</code>	
<code>\Vdots</code>	<code>& & & \Vdots \\</code>	
<code>0</code>	<code>& \Cdots & \Cdots & 0</code>	
<code>\end{bNiceMatrix}</code>		



²However, for the columns of type `w` and `W`, the cells are composed in math mode (in the environments of `nicematrix`) whereas in `{array}` of `array`, they are composed in text mode.

³The command `\iddots`, defined in `nicematrix`, is a variant of `\ddots` with dots going forward: $\cdot\cdot\cdot$. If `mathdots` is loaded, the version of `mathdots` is used. It corresponds to the command `\adots` of `unicode-math`.

⁴The precise definition of a “non-empty cell” is given below (cf. p. 16).

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

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

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

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

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

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

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

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

3.1 The option `nullify-dots`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.2 The command `\Hdotsfor`

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

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

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

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

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

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

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

3.3 How to generate the continuous dotted lines transparently

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

- The option `renew-dots`

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

- The option `renew-matrix`

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

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

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

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

3.4 Fine tuning of the dotted lines

The distance between a node and the end of a dotted line is set by `dotted-lines-margin`. The initial value of this key is 0.3 em (it’s recommended to use a unit dependant of the current font). For an example, cf. p. 18.

4 The Tikz nodes created by `nicematrix`

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

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

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

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

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

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

⁷The 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.)

In fact, the package `nicematrix` can create “extra nodes”: the “medium nodes” and the “large nodes”. The first ones are created with the option `create-medium-nodes` and the second ones with the option `create-large-nodes`.⁸

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

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

The names of the “large nodes” are constructed by adding the suffix “-large” to the names of the “normal nodes”. In the following example, we have underlined the “large nodes”. We consider that this example is self-explanatory.⁹

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

The “large nodes” of the first column and last column may appear too small for some usage. That’s why it’s possible to use the options `left-margin` and `right-margin` to add space on both sides of the array and also space in the “large nodes” of the first column and last column. In the following example, we have used the options `left-margin` and `right-margin`.¹⁰

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

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

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

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

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

We explain below how to fill the nodes created by `nicematrix` (cf. p. 20).

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

⁹There is no “large nodes” created in the exterior rows and columns (for these rows and columns, cf. p. 8).

¹⁰The options `left-margin` and `right-margin` take dimensions as values but, if no value is given, the default value is used, which is `\arraycolsep` (by default: 5 pt). There is also an option `margin` to fix both `left-margin` and `right-margin` to the same value.

5 The code-after

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

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

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

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

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

6 The environment {NiceArray}

The environment `{NiceArray}` is similar to the environment `{array}`. As for `{array}`, the mandatory argument is the preamble of the array. However, for technical reasons, in this preamble, the user must use the letters `L`, `C` and `R`¹¹ instead of `l`, `c` and `r`. It's possible to use the constructions `w{...}{...}`, `W{...}{...}`, `|`, `>{...}`, `<{...}`, `@{...}`, `!{...}` and `*{n}{...}` but the letters `p`, `m` and `b` should not be used.¹²

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

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

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

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

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

In the following example, we use an environment `{pNiceArray}` (we don't use `{pNiceMatrix}` because we want to use the types `L` and `R` — in `{pNiceMatrix}`, all the columns are of type `C`).

```
\begin{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 delimiters.

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

¹²In a command `\multicolumn`, one should also use the letters `L`, `C`, `R`.

```

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

```

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

7 The exterior rows and columns

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

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

```

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

```

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

We have several remarks to do.

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

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

```

\NiceMatrixOptions{code-for-first-row = \color{red},
                  code-for-first-col = \color{blue},
                  code-for-last-row = \color{green},
                  code-for-last-col = \color{magenta}}
 $\begin{pNiceArray}{CC|CC}[first-row,last-row,first-col,last-col]$ 

```



```

& C_1 & C_2 & C_3 & C_4 & \\
L_1 & a_{11} & a_{12} & a_{13} & a_{14} & L_1 \\
L_2 & a_{21} & a_{22} & a_{23} & a_{24} & L_2 \\
\hline
L_3 & a_{31} & a_{32} & a_{33} & a_{34} & L_3 \\
L_4 & a_{41} & a_{42} & a_{43} & a_{44} & L_4 \\
& C_1 & C_2 & C_3 & C_4 & \\
\end{pNiceArray}$

```

$$\begin{array}{c}
L_1 \\
L_2 \\
L_3 \\
L_4
\end{array}
\begin{array}{c}
C_1 \\
C_2 \\
C_3 \\
C_4
\end{array}
\begin{array}{c}
C_1 \\
C_2 \\
C_3 \\
C_4
\end{array}
\begin{array}{c}
C_1 \\
C_2 \\
C_3 \\
C_4
\end{array}
\begin{array}{c}
L_1 \\
L_2 \\
L_3 \\
L_4
\end{array}$$

Remarks

- As shown in the previous example, an horizontal rule (drawn by `\hline`) doesn't extend in the exterior columns and a vertical rule (specified by a “|” in the preamble of the array) doesn't extend in the exterior rows.¹³

If one wishes to define new specifiers for columns in order to draw vertical rules (for example thicker than the standard rules), he should consider the command `\OnlyMainNiceMatrix` described on page 14.

- Logically, the potential option `columns-width` (described p. 10) doesn't apply to the “first column” and “last column”.
- For technical reasons, it's not possible to use the option of the command `\` after the “first row” or before the “last row” (the placement of the delimiters would be wrong).

8 The dotted lines to separate rows or columns

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

```

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

```

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

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

```

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

```

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

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

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

```

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

```

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`. For example, in this document, we have loaded the extension `arydshln` which uses the letter “:” to specify a vertical dashed line. Thus, by using `letter-for-dotted-lines`, we can use the vertical lines of both `arydshln` and `nicematrix`.

```

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

```

9 The width of the columns

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

```

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

```

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

```

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

```

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

It's possible to give the special value `auto` to the option `columns-width`: all the columns of the array will have a width equal to the widest cell of the array.¹⁴

```

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

```

¹⁴The result is achieved with only one compilation (but Tikz will have written informations in the `.aux` file and a message requiring a second compilation will appear).

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

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

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

But it's also possible to fix a zone where all the matrices will have their columns of the same width, equal to the widest cell of all the matrices. This construction uses the environment `\NiceMatrixBlock` with the option `auto-columns-width`.¹⁵

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

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

Several compilations may be necessary to achieve the job.

10 Block matrices

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

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

The command `\Block` must be used in the upper leftmost cell of the array with two arguments. The first argument is the size of the block with the syntax `i-j` where `i` is the number of rows of the block and `j` its number of columns. The second argument is the content of the block (composed in math mode). A Tikz node corresponding to the merged cells is created with the name “`name-i-j`” where `name` is the name given to the array.

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

$$\left[\begin{array}{ccc|c} A & & & 0 \\ & \hspace{1cm} & & \vdots \\ & & & 0 \\ \hline 0 & \cdots & 0 & 0 \end{array} \right]$$

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

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

```

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

```

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

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

11 Advanced features

11.1 The option `small`

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

```

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

```

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

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

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

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

11.2 The counters `iRow` and `jCol`

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

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

`\begin{pNiceMatrix}`% don't forget the %

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

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

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

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

These commands take two mandatory arguments. The first is the format of the matrix, with the syntax $n \times p$ 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}}}`

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

11.3 The option `hlines`

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

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

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

11.4 The option `light-syntax`

The option `light-syntax`¹⁷ allow 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 a space 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]
{} \alpha & & \beta & ;
\alpha & 2\cos \alpha & \{\cos \alpha + \cos \beta\} ;
\beta & \cos \alpha + \cos \beta & \{2 \cos \beta\}
\end{bNiceMatrix}
```

$$\begin{matrix} & \alpha & \beta \\ \alpha & \begin{bmatrix} 2 \cos \alpha & \cos \alpha + \cos \beta \end{bmatrix} \\ \beta & \begin{bmatrix} \cos \alpha + \cos \beta & 2 \cos \beta \end{bmatrix} \end{matrix}$$

¹⁷This option is inspired by the extension `spalign` of Joseph Rabinoff.

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.

11.5 Utilisation of the column type S of siunitx

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

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

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

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

12 Technical remarks

12.1 Definition of new column types

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

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

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

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

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

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

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

12.2 Intersections of dotted lines

Since the version 3.1 of `nicematrix`, the dotted lines created by `\Cdots`, `\Ldots`, `\Vdots`, etc. can't intersect.¹⁸

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

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

¹⁸On the contrary, dotted lines created by `\hdottedline`, the letter “:” in the preamble of the array and the command `\line` in the `code-after` can have intersections with other dotted lines.

```

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

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

12.3 The names of the Tikz nodes created by nicematrix

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

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

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

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

12.4 Diagonal lines

By default, all the diagonal lines¹⁹ of a same array are “parallelized”. That means that the first diagonal line is drawn and, then, the other lines are drawn parallel to the first one (by rotation around the left-most extremity of the line). That's why the position of the instructions `\Ddots` in the array can have a marked effect on the final result.

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

Example with parallelization (default):

```

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

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

```

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

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

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

The same example without parallelization:

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

¹⁹We speak of the lines created by `\Ddots` and not the lines created by a command `\line` in `code-after`.

12.5 The “empty” cells

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

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

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

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

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

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

12.6 The option `exterior-arraycolsep`

The environment `{array}` inserts an horizontal space equal to `\arraycolsep` before and after each column. In particular, there is a space equal to `\arraycolsep` before and after the array. This feature of the environment `{array}` was probably not a good idea²⁰. The environment `{matrix}` of `amsmath` and its variants (`{pmatrix}`, `{vmatrix}`, etc.) of `amsmath` prefer to delete these spaces with explicit instructions `\hskip -\arraycolsep`. The extension `nicematrix` does the same in all its environments, `{NiceArray}` included. However, if the user wants the environment `{NiceArray}` behaving by default like the environment `{array}` of `array` (for example, when adapting an existing document) it's possible to control this behaviour with the option `exterior-arraycolsep`, set by the command `\NiceMatrixOptions`. With this option, exterior spaces of length `\arraycolsep` will be inserted in the environments `{NiceArray}` (the other environments of `nicematrix` are not affected).

12.7 The class option `draft`

The package `nicematrix` is rather slow when drawing the dotted lines (generated by `\Cdots`, `\Ldots`, `\Ddots`, etc. but also by `\hdottedline` or the specifier `:`).²¹ That's why, when the class option `draft` is used, the dotted lines are not drawn, for a faster compilation.

²⁰In the documentation of `{amsmath}`, we can read: *The extra space of `\arraycolsep` that `array` adds on each side is a waste so we remove it [in `{matrix}`] (perhaps we should instead remove it from `array` in general, but that's a harder task)*. It's possible to suppress these spaces for a given environment `{array}` with a construction like `\begin{array}{@{}cccc@{}}...\end{array}`.

²¹The main reason is that we want dotted lines with round dots (and not square dots) with the same space on both extremities of the lines. To achieve this goal, we have to construct our own system of dotted lines.

12.8 A technical problem with the argument of `\`

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

```
\begin{pNiceMatrix}
a & \frac{AB}{2mm} \\
b & c
\end{pNiceMatrix}
```

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

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

```
\begin{pNiceMatrix}
a & \frac{AB}{2mm} \\
\noalign{\kern2mm}
b & c
\end{pNiceMatrix}
```

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

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

```
\begin{pNiceMatrix}
a & \multicolumn{1}{C}{\frac{AB}{2mm}} \\
b & c
\end{pNiceMatrix}
```

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

12.9 Obsolete environments

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

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

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

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

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

13 Examples

13.1 Dotted lines

A tridiagonal matrix:

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

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

A permutation matrix (as an example, we have raised the value of `dotted-lines-margin`).

```

 $\begin{pNiceMatrix}[\textcolor{violet}{dotted-lines-margin=0.6em}]
0 & 1 & 0 & & \cdots & 0 & \\
\vdots & & & \ddots & & \vdots & \\
& & & \ddots & & & \\
& & & \ddots & & 0 & \\
0 & 0 & & & & 1 & \\
1 & 0 & & \cdots & & 0 & 
\end{pNiceMatrix}$ 

```

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

An example with `\Iddots`:

```

 $\begin{pNiceMatrix}
1 & \cdots & & 1 & \\
\vdots & & & 0 & \\
& \textcolor{violet}{\Iddots} & \textcolor{violet}{\Iddots} & \vdots & \\
1 & 0 & \cdots & 0 & 
\end{pNiceMatrix}$ 

```

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

An example with `\multicolumn`:

```

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

```

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

An example with `\Hdotsfor`:

```

 $\begin{pNiceMatrix}[\text{nullify-dots}]
0 & 1 & 1 & 1 & 1 & 0 & \\
0 & 1 & 1 & 1 & 1 & 0 & \\
\vdots & \textcolor{violet}{\Hdotsfor{4}} & \textcolor{violet}{\Hdotsfor{4}} & \textcolor{violet}{\Hdotsfor{4}} & \textcolor{violet}{\Hdotsfor{4}} & & \\
& \textcolor{violet}{\Hdotsfor{4}} & \textcolor{violet}{\Hdotsfor{4}} & \textcolor{violet}{\Hdotsfor{4}} & \textcolor{violet}{\Hdotsfor{4}} & & \\
& \textcolor{violet}{\Hdotsfor{4}} & \textcolor{violet}{\Hdotsfor{4}} & \textcolor{violet}{\Hdotsfor{4}} & \textcolor{violet}{\Hdotsfor{4}} & & \\
0 & 1 & 1 & 1 & 1 & 0 & 
\end{pNiceMatrix}$ 

```

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

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

$$\begin{array}{ccccccc}
1 & & & & & & \\
0 & & & & & & \\
0 & & & & & & \\
& & & & & & \\
& & & & & & \\
0 & & & & & & \\
& & & & & &
\end{array}$$

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

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

```

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

```

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

13.3 How to highlight cells of the matrix

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

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

```

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

```

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

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

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

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

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

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

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

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 $i-j$ 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).

```
$\begin{pNiceMatrix}%
[matrix,
  code-after = { \tikz \node [highlight = (1-1)] {} ; } ]
\Block{3-3}<\Large>{A} & & 0 \\
& \hspace*{1cm} & & \Vdots \\
& & 0 \\
0 & \Cdots & 0 & 0
\end{pNiceMatrix}$
```

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

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

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

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

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 = example-,
                             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,
code-after = {\tikz \path [name suffix = -large,
                           fill = red!15,
                           blend mode = multiply]
               (1-1.north west)
               |- (2-2.north west)
               |- (3-3.north west)
               |- (4-4.north west)
               |- (4-4.south east)
               |- (1-1.north west) ; } ]
A_{11} & A_{12} & A_{13} & A_{14} \\\
A_{21} & A_{22} & A_{23} & A_{24} \\\
```

```

A_{31} & A_{32} & A_{33} & A_{34} \\
A_{41} & A_{42} & A_{43} & A_{44} \\
\end{pNiceArray}

```

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

13.4 Direct utilisation of the Tikz nodes

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

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

```

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

\NiceMatrixOptions{nullify-dots}

```

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

```

$\begin{array}{cc}
&

```

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

```

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

```

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

```

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

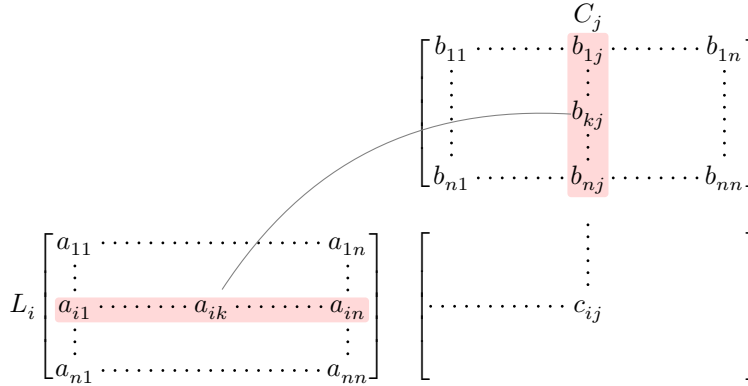
\end{NiceMatrixBlock}

```

```

\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}

```



14 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`.

14.1 Declaration of the package and extensions loaded

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

First, `tikz` and the Tikz library `fit` are loaded before the `\ProvidesExplPackage`. They are loaded this way because `\usetikzlibrary` in `expl3` code fails.²²

```

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

```

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

```

4 \RequirePackage{l3keys2e}
5 \ProvidesExplPackage
6   {nicematrix}
7   {\myfiledate}
8   {\myfileversion}
9   {Mathematical matrices with TikZ}

```

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

```

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

```

²²cf. tex.stackexchange.com/questions/57424/using-of-usetikzlibrary-in-an-expl3-package-fails

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

We load `array` and `amsmath`.

```

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

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

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

```

14.2 Technical definitions

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

```

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

```

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

```

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

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

```

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

```

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

```

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

The following counter will count the environments `{NiceArray}`. The value of this counter will be used to prefix the names of the Tikz nodes created in the array.

```

49 \int_new:N \g_@@_env_int

```

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

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

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

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

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

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

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

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

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

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

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

Consider the following code:

```
$\begin{pNiceMatrix}
a & b & c \\
d & e & \Vdots \\
f & \Cdots & \\
g & h & i \\
\end{pNiceMatrix}$
```

First, the dotted line created by the `\Vdots` will be drawn. The implicit cell in position 2-3 will be considered as “dotted”. Then, we will have to draw the dotted line specified by the `\Cdots`; the final extremity of that line will be exactly in position 2-3 and, for that new second line, it should be considered as a *closed* extremity (since it is dotted). However, we don't have the (normal) Tikz node of that node (since it's an implicit cell): we can't draw such a line. That's why that dotted line will be said *impossible* and an error will be raised.²³

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

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

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

²³Of course, the user should solve the problem by adding the lacking ampersands.

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

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

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

```
74 \dim_new:N \l_@@_dotted_lines_margin_dim
75 \dim_set:Nn \l_@@_dotted_lines_margin_dim { 0.3 em }
```

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

```
76 \dim_new:N \l_@@_radius_dim
77 \dim_set:Nn \l_@@_radius_dim { 0.53 pt }
```

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

```
78 \str_new:N \g_@@_type_env_str
79 \tl_new:N \g_@@_code_after_tl
```

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

```
80 \int_new:N \l_@@_save_iRow_int
81 \int_new:N \l_@@_save_jCol_int
```

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

14.2.1 Variables for the exterior rows and columns

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

- **First row**

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

```
82 \int_new:N \l_@@_first_row_int
83 \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.

```
84 \int_new:N \l_@@_first_col_int
85 \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).

```
86 \int_new:N \l_@@_last_row_int
87 \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”.²⁴

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

- **Last column**

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

```
89 \int_new:N \l_@@_last_col_int
90 \int_set:Nn \l_@@_last_col_int { -1 }
```

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

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

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

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

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

14.2.2 The column **S** of `siunitx`

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

```
92 \bool_new:N \c_@@_siunitx_loaded_bool
93 \AtBeginDocument
94 {
95   \ifpackageloaded { siunitx }
96     { \bool_set_true:N \c_@@_siunitx_loaded_bool }
97     { }
98 }
```

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:

²⁴We can’t use `\l_@@_last_row_int` for this usage because, if `nicematrix` has read its value from the `aux` file, the value of the counter won’t be `-1` any longer.

```

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

```

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

```

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

```

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

```

99 \cs_set_protected:Npn \@@adapt_S_column:
100 {

```

In the preamble of the LaTeX document, the boolean `\c_@@_siunitx_loaded_bool` won't be known. That's why we test the existence of `\c_@@_siunitx_loaded_bool` and not its value.²⁵

```

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

```

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

```

105     \cs_set_eq:NN \NC@find \prg_do_nothing:
106     \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`).

```

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

```

²⁵Indeed, `nicematrix` may be used in the preamble of the LaTeX document. For example, in this document, we compose a matrix in the box `\ExampleOne` before loading `arydshln` (because `arydshln` is not totally compatible with `nicematrix`).

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

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

```

114     \cs_gset_eq:NN \@@_adapt_S_column: \prg_do_nothing:
115   }
116 }

```

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).

```

117 \cs_new_protected:Npn \@@_renew_NC@rewrite@S:
118 {
119   \renewcommand*{\NC@rewrite@S}[1][]
120   {
121     \@temptokena \exp_after:wN
122     {
123       \tex_the:D \@temptokena
124       > { \@@_Cell: \c_@@_table_collect_begin_tl S {##1} }
125       c
126       < { \c_@@_table_print_tl \@@_end_Cell: }
127     }
128     \NC@find
129   }
130 }

```

14.3 The options

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

```

131 \bool_new:N \l_@@_light_syntax_bool

```

The token list `\l_@@_pos_env_str` will contain one of the three values `t`, `c` or `b` and will indicate the position of the environment as in the option of the environment `{array}`. For the environment `{pNiceMatrix}`, `{pNiceArray}` and their variants, the value will programmatically be fixed to `c`. For the environment `{NiceArray}`, however, the three values `t`, `c` and `b` are possible.

```

132 \str_new:N \l_@@_pos_env_str
133 \str_set:Nn \l_@@_pos_env_str c

```

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

```

134 \bool_new:N \l_@@_exterior_arraycolsep_bool

```

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

```

135 \bool_new:N \l_@@_parallelize_diags_bool
136 \bool_set_true:N \l_@@_parallelize_diags_bool

```

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

```

137 \bool_new:N \l_@@_hlines_bool

```

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

```

138 \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).

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

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

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

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

```
141 \bool_new:N \l_@@_medium_nodes_bool
142 \bool_new:N \g_@@_medium_nodes_bool
143 \bool_new:N \l_@@_large_nodes_bool
144 \bool_new:N \g_@@_large_nodes_bool
```

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

```
145 \dim_new:N \l_@@_left_margin_dim
146 \dim_new:N \l_@@_right_margin_dim
147 \dim_new:N \g_@@_width_last_col_dim
148 \dim_new:N \g_@@_width_first_col_dim
```

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

```
149 \dim_new:N \l_@@_extra_left_margin_dim
150 \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.

```
151 \tl_new:N \l_@@_end_of_row_tl
152 \tl_set:Nn \l_@@_end_of_row_tl { ; }
```

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

```
153 \keys_define:nn { NiceMatrix / Global }
154 {
155   dotted-lines-margin .dim_set:N = \l_@@_dotted_lines_margin_dim ,
156   dotted-lines-margin .value_required:n = true ,
157   light-syntax .bool_set:N = \l_@@_light_syntax_bool ,
158   light-syntax .default:n = true ,
159   end-of-row .tl_set:N = \l_@@_end_of_row_tl ,
160   end-of-row .value_required:n = true ,
161   code-for-first-col .tl_set:N = \l_@@_code_for_first_col_tl ,
162   code-for-first-col .value_required:n = true ,
163   code-for-last-col .tl_set:N = \l_@@_code_for_last_col_tl ,
164   code-for-last-col .value_required:n = true ,
165   code-for-first-row .tl_set:N = \l_@@_code_for_first_row_tl ,
166   code-for-first-row .value_required:n = true ,
167   code-for-last-row .tl_set:N = \l_@@_code_for_last_row_tl ,
168   code-for-last-row .value_required:n = true ,
169   small .bool_set:N = \l_@@_small_bool ,
170   hlines .bool_set:N = \l_@@_hlines_bool ,
171   parallelize-diags .bool_set:N = \l_@@_parallelize_diags_bool ,
```

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

```
172   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
173   renew-dots .value_forbidden:n = true ,
174   nullify-dots .bool_set:N = \l_@@_nullify_dots_bool ,
```

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

```

175   create-medium-nodes .bool_set:N = \l_@@_medium_nodes_bool ,
176   create-large-nodes .bool_set:N = \l_@@_large_nodes_bool ,
177   create-extra-nodes .meta:n =
178     { create-medium-nodes , create-large-nodes } ,
179   left-margin .dim_set:N = \l_@@_left_margin_dim ,
180   left-margin .default:n = \arraycolsep ,
181   right-margin .dim_set:N = \l_@@_right_margin_dim ,
182   right-margin .default:n = \arraycolsep ,
183   margin .meta:n = { left-margin = #1 , right-margin = #1 } ,
184   margin .default:n = \arraycolsep ,
185   extra-left-margin .dim_set:N = \l_@@_extra_left_margin_dim ,
186   extra-right-margin .dim_set:N = \l_@@_extra_right_margin_dim ,
187   extra-margin .meta:n =
188     { extra-left-margin = #1 , extra-right-margin = #1 }
189 }

```

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

```

190 \keys_define:nn { NiceMatrix / Env }
191 {
192   columns-width .code:n =
193     \str_if_eq:nnTF { #1 } { auto }
194     { \bool_set_true:N \l_@@_auto_columns_width_bool }
195     { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,
196   columns-width .value_required:n = true ,
197   name .code:n =
198     \unless \ifmeasuring@
199       \str_set:Nn \l_tmpa_str { #1 }
200       \seq_if_in:NVTF \g_@@_names_seq \l_tmpa_str
201         { \@@_error:nn { Duplicate-name } { #1 } }
202         { \seq_gput_left:NV \g_@@_names_seq \l_tmpa_str }
203       \str_set_eq:NN \l_@@_name_str \l_tmpa_str
204     \fi ,
205   name .value_required:n = true ,
206   code-after .tl_gset:N = \g_@@_code_after_tl ,
207   code-after .value_required:n = true ,
208   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
209   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
210   last-row .int_set:N = \l_@@_last_row_int ,
211   last-row .default:n = -1 ,
212 }

```

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

```

213 \keys_define:nn { NiceMatrix }
214 {
215   NiceMatrixOptions .inherit:n =
216     {
217       NiceMatrix / Global ,
218     } ,
219   NiceMatrix .inherit:n =
220     {
221       NiceMatrix / Global ,
222       NiceMatrix / Env
223     } ,
224   NiceArray .inherit:n =
225     {
226       NiceMatrix / Global ,
227       NiceMatrix / Env ,
228     } ,

```



```

229 pNiceArray .inherit:n =
230 {
231     NiceMatrix / Global ,
232     NiceMatrix / Env ,
233 }
234 }

```

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

```

235 \keys_define:nn { NiceMatrix / NiceMatrixOptions }
236 {

```

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

```

237 renew-matrix .code:n = \@@_renew_matrix: ,
238 renew-matrix .value_forbidden:n = true ,
239 RenewMatrix .code:n = \@@_error:n { Option~RenewMatrix~suppressed } ,
240 transparent .meta:n = { renew-dots , renew-matrix } ,
241 transparent .value_forbidden:n = true,
242 Transparent .code:n = \@@_error:n { Option~Transparent~suppressed } ,

```

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

```

243 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.

```

244 columns-width .code:n =
245 \str_if_eq:nnTF { #1 } { auto }
246 { \@@_error:n { Option~auto~for~columns~width } }
247 { \dim_set:Nn \l_@@_columns_width_dim { #1 } } ,

```

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

```

248 allow-duplicate-names .code:n =
249 \@@_msg_redirect_name:nn { Duplicate~name } { none } ,
250 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`).

```

251 letter-for-dotted-lines .code:n =
252 {
253     \int_compare:nTF { \tl_count:n { #1 } = \c_one_int }
254     { \str_set:Nx \l_@@_letter_for_dotted_lines_str { #1 } }
255     { \@@_error:n { Bad~value~for~letter~for~dotted~lines } }
256 } ,
257 letter-for-dotted-lines .value_required:n = true ,

258 unknown .code:n = \@@_error:n { Unknown~key~for~NiceMatrixOptions }
259 }

260 \str_new:N \l_@@_letter_for_dotted_lines_str
261 \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.

```
262 \NewDocumentCommand \NiceMatrixOptions { m }
263 { \keys_set:nn { NiceMatrix / NiceMatrixOptions } { #1 } }
```

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

```
264 \keys_define:nn { NiceMatrix / NiceMatrix }
265 {
266   last-col .code:n = \tl_if_empty:nTF {#1}
267     { \@@_error:n { last-col~empty~for~NiceMatrix } }
268     { \int_set:Nn \l_@@_last_col_int { #1 } } ,
269   unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix }
270 }
```

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

```
271 \keys_define:nn { NiceMatrix / NiceArray }
272 {
```

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

```
273 c .code:n = \str_set:Nn \l_@@_pos_env_str c ,
274 t .code:n = \str_set:Nn \l_@@_pos_env_str t ,
275 b .code:n = \str_set:Nn \l_@@_pos_env_str b ,
```

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

```
276 last-col .code:n = \tl_if_empty:nF { #1 }
277   { \@@_error:n { last-col~non~empty~for~NiceArray } }
278   { \int_zero:N \l_@@_last_col_int ,
279   unknown .code:n = \@@_error:n { Unknown~option~for~NiceArray }
280 }

281 \keys_define:nn { NiceMatrix / pNiceArray }
282 {
283   first-col .code:n = \int_zero:N \l_@@_first_col_int ,
284   last-col .code:n = \tl_if_empty:nF {#1}
285     { \@@_error:n { last-col~non~empty~for~NiceArray } }
286     { \int_zero:N \l_@@_last_col_int ,
287   first-row .code:n = \int_zero:N \l_@@_first_row_int ,
288   last-row .int_set:N = \l_@@_last_row_int ,
289   last-row .default:n = -1 ,
290   unknown .code:n = \@@_error:n { Unknown~option~for~NiceMatrix }
291 }
```

14.4 Important code used by `{NiceArrayWithDelims}`

The pseudo-environment `\@@_Cell:–\@@_end_Cell:` will be used to format the cells of the array. In the code, the affectations are global because this pseudo-environment will be used in the cells of a `\halign` (via an environment `{array}`).

```
292 \cs_new_protected:Nn \@@_Cell:
293 {
```

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

```
294   \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 `arydshn`, create special rows in the `\halign` that we don't want to take into account.

```

295 \int_compare:nNnT \c@jCol = \c_one_int
296 {
297   \int_compare:nNnT \l_@@_first_col_int = \c_one_int
298   \@@_begin_of_row:
299 }
300 \int_gset:Nn \g_@@_col_total_int
301 { \int_max:nn \g_@@_col_total_int \c@jCol }

```

The content of the cell is composed in the box `\l_tmpa_box` because we want to compute some dimensions of the box. The `\hbox_set_end:` corresponding to this `\hbox_set:Nw` will be in the `\@@_end_Cell:` (and the `\c_math_toggle_token` also).

```

302 \hbox_set:Nw \l_tmpa_box
303 \c_math_toggle_token
304 \bool_if:NT \l_@@_small_bool \scriptstyle

```

We will call *corners* of the matrix the cases which are at the intersection of the exterior rows and exterior columns (of course, the four corners doesn't always exist simultaneously). The codes `\l_@@_code_for_first_row_tl` and `al` don't apply in the corners of the matrix.

```

305 \int_compare:nNnTF \c@iRow = 0
306 { \int_compare:nNnT \c@jCol > 0 \l_@@_code_for_first_row_tl }
307 {
308   \int_compare:nNnT \c@iRow = \l_@@_last_row_int
309   \l_@@_code_for_last_row_tl
310 }
311 }

```

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

```

312 \cs_new_protected:Nn \@@_begin_of_row:
313 {
314   \int_gincr:N \c@iRow
315   \dim_gset_eq:NN \g_@@_dp_ante_last_row_dim \g_@@_dp_last_row_dim
316   \dim_gset:Nn \g_@@_dp_last_row_dim { \box_dp:N \@arstrutbox }
317   \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
318 }

```

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.

```

319 \cs_new_protected:Npn \@@_actualization_for_first_and_last_row:
320 {
321   \int_compare:nNnT \c@iRow = 0
322   {
323     \dim_gset:Nn \g_@@_dp_row_zero_dim
324     { \dim_max:nn \g_@@_dp_row_zero_dim { \box_dp:N \l_tmpa_box } }
325     \dim_gset:Nn \g_@@_ht_row_zero_dim
326     { \dim_max:nn \g_@@_ht_row_zero_dim { \box_ht:N \l_tmpa_box } }
327   }
328   \int_compare:nNnT \c@iRow = \c_one_int
329   {
330     \dim_gset:Nn \g_@@_ht_row_one_dim
331     { \dim_max:nn \g_@@_ht_row_one_dim { \box_ht:N \l_tmpa_box } }
332   }
333   \dim_gset:Nn \g_@@_ht_last_row_dim
334   { \dim_max:nn \g_@@_ht_last_row_dim { \box_ht:N \l_tmpa_box } }
335   \dim_gset:Nn \g_@@_dp_last_row_dim
336   { \dim_max:nn \g_@@_dp_last_row_dim { \box_dp:N \l_tmpa_box } }
337 }

```

```

338 \cs_new_protected:Nn \@@_end_Cell:
339 {
340   \c_math_toggle_token
341   \hbox_set_end:

```

We want to compute in `\g_@@_max_cell_width_dim` the width of the widest cell of the array (except the cells of the “first column” and the “last column”).

```

342   \dim_gset:Nn \g_@@_max_cell_width_dim
343   { \dim_max:nn \g_@@_max_cell_width_dim { \box_wd:N \l_tmpa_box } }

```

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

```

344   \@@_actualization_for_first_and_last_row:

```

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

```

345   \tikz
346   [
347     remember~picture ,
348     inner~sep = \c_zero_dim ,
349     minimum~width = \c_zero_dim ,
350     baseline
351   ]
352   \node
353   [
354     anchor = base ,
355     name = nm - \int_use:N \g_@@_env_int -
356             \int_use:N \c@iRow -
357             \int_use:N \c@jCol ,
358     alias =
359       \str_if_empty:NF \l_@@_name_str
360       {
361         \l_@@_name_str -
362         \int_use:N \c@iRow -
363         \int_use:N \c@jCol
364       }
365   ]
366   \bgroup
367   \box_use:N \l_tmpa_box
368   \egroup ;
369 }
370 \cs_generate_variant:Nn \dim_set:Nn { N x }

```

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

```

371 \cs_new_protected:Nn \@@_renewcolumnntype:nn
372 {
373   \newcolumnntype #1 [ 2 ]
374   {
375     > {
376       \hbox_set:Nw \l_tmpa_box
377       \@@_Cell:
378     }
379     c
380     < {
381       \@@_end_Cell:
382       \hbox_set_end:
383       #2
384       \tikz [ remember~picture ]
385       \node
386       [

```

```

387         name = nm - \int_use:N \g_@@_env_int -
388                 \int_use:N \c@iRow -
389                 \int_use:N \c@jCol - w,
390     alias =
391         \str_if_empty:NF \l_@@_name_str
392         {
393             \l_@@_name_str -
394             \int_use:N \c@iRow -
395             \int_use:N \c@jCol - w
396         } ,
397     inner~sep = \c_zero_dim ,
398 ]
399 { \makebox [ ##2 ] [ ##1 ] { \box_use:N \l_tmpa_box } }
400 ;
401 }
402 }
403 }

```

The argument of the following command `\@@_instruction_of_type:n` defined below is the type of the instruction (`Cdots`, `Vdots`, `Ddots`, etc.). This command writes in the corresponding `\g_@@_type_lines_tl` the instruction which will really draw the line after the construction of the matrix.

For example, for the following matrix,

```

\begin{pNiceMatrix}
1 & 2 & 3 & 4 \\
5 & \Cdots & & 6 \\
7 & \Cdots & & 
\end{pNiceMatrix}

```

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

the content of `\g_@@_Cdots_lines_tl` will be:

```

\@@_draw_Cdots:nn {2}{2}
\@@_draw_Cdots:nn {3}{2}

```

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

```

404 \bool_if:NTF \c_@@_draft_bool
405 { \cs_set_protected:Npn \@@_instruction_of_type:n #1 { } }
406 {
407     \cs_new_protected:Npn \@@_instruction_of_type:n #1
408     {

```

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).

```

409         \tl_gput_right:cx
410         { \g_@@_ #1 _ lines _ tl }
411         {
412             \use:c { @@ _ draw _ #1 : nn }
413             { \int_use:N \c@iRow }
414             { \int_use:N \c@jCol }
415         }
416     }
417 }

```

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`.

```

418 \cs_new_protected:Npn \@@_array:
419 {
420     \bool_if:NTF \c_@@_revtex_bool
421     {
422         \cs_set_eq:NN \@acoll \@arrayacol

```

```

423     \cs_set_eq:NN \@acolr \@arrayacol
424     \cs_set_eq:NN \@acol \@arrayacol
425     \cs_set:Npn \@halignto { }
426     \@array@array
427 }
428 \array

```

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

```

429 [ \l_@@_pos_env_str ]
430 }

```

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

```

431 \cs_new:Npn \@@_everycr:
432 { \noalign { \@@_everycr_i: } }
433 \cs_new_protected:Npn \@@_everycr_i:
434 {
435     \int_gzero:N \c@jCol
436     \bool_if:NT \l_@@_hlines_bool
437     {

```

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.

```

438     \int_compare:nNnT \c@iRow > { -1 }
439     {
440         \int_compare:nNnF \c@iRow = \l_@@_last_row_int
441         {
442             \hrule \@height \arrayrulewidth
443             \skip_vertical:n { - \arrayrulewidth }
444         }
445     }
446 }
447 }

```

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

```

448 \cs_new_protected:Npn \@@_pre_array:
449 {
450     \cs_if_exist:NT \theiRow
451     { \int_set_eq:NN \l_@@_save_iRow_int \c@iRow }
452     \int_gzero_new:N \c@iRow
453     \cs_if_exist:NT \thejCol
454     { \int_set_eq:NN \l_@@_save_jCol_int \c@jCol }
455     \int_gzero_new:N \c@jCol
456     \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}`).

```

457     \bool_if:NT \l_@@_small_bool
458     {
459         \cs_set:Npn \arraystretch { 0.47 }
460         \dim_set:Nn \arraycolsep { 1.45 pt }
461     }

```

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

```

462     \bool_gset_eq:NN \g_@@_medium_nodes_bool \l_@@_medium_nodes_bool
463     \bool_gset_eq:NN \g_@@_large_nodes_bool \l_@@_large_nodes_bool

```

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

```

464 \cs_set:Npn \ialign
465 {
466   \bool_if:NTF \c_@@_colortbl_loaded_bool
467   {
468     \CT@everycr
469     {
470       \noalign { \cs_gset_eq:NN \CT@row@color \prg_do_nothing: }
471       \@@_everycr:
472     }
473   }
474   { \everycr { \@@_everycr: } }
475   \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`²⁶ and `\extrarowheight` (of `array`). That box is inserted (via `\@arstrut`) in the beginning of each row of the array. That's why we use the dimensions of that box to initialize the variables which will be the dimensions of the potential first and last row of the environment. This initialization must be done after the creation of `\@arstrutbox` and that's why we do it in the `\ialign`.

```

476 \dim_gzero_new:N \g_@@_dp_row_zero_dim
477 \dim_gset:Nn \g_@@_dp_row_zero_dim { \box_dp:N \@arstrutbox }
478 \dim_gzero_new:N \g_@@_ht_row_zero_dim
479 \dim_gset:Nn \g_@@_ht_row_zero_dim { \box_ht:N \@arstrutbox }
480 \dim_gzero_new:N \g_@@_ht_row_one_dim
481 \dim_gset:Nn \g_@@_ht_row_one_dim { \box_ht:N \@arstrutbox }
482 \dim_gzero_new:N \g_@@_dp_ante_last_row_dim
483 \dim_gzero_new:N \g_@@_ht_last_row_dim
484 \dim_gset:Nn \g_@@_ht_last_row_dim { \box_ht:N \@arstrutbox }
485 \dim_gzero_new:N \g_@@_dp_last_row_dim
486 \dim_gset:Nn \g_@@_dp_last_row_dim { \box_dp:N \@arstrutbox }

```

After its first utilisation, the definition of `\ialign` will revert automatically to its default definition. With this programming, we will have, in the cells of the array, a clean version of `\ialign`.²⁷

```

487 \cs_set:Npn \ialign
488 {
489   \everycr { }
490   \tabskip = \c_zero_skip
491   \halign
492 }
493 \halign
494 }

```

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}`.

```

495 \newcolumnntype L { > \@@_Cell: l < \@@_end_Cell: }
496 \newcolumnntype C { > \@@_Cell: c < \@@_end_Cell: }
497 \newcolumnntype R { > \@@_Cell: r < \@@_end_Cell: }
498 \cs_set_eq:NN \Ldots \@@_Ldots
499 \cs_set_eq:NN \Cdots \@@_Cdots
500 \cs_set_eq:NN \Vdots \@@_Vdots
501 \cs_set_eq:NN \Ddots \@@_Ddots
502 \cs_set_eq:NN \Iddots \@@_Iddots
503 \cs_set_eq:NN \hdottedline \@@_hdottedline:
504 \cs_set_eq:NN \Hspace \@@_Hspace:
505 \cs_set_eq:NN \Hdotsfor \@@_Hdotsfor:
506 \cs_set_eq:NN \multicolumn \@@_multicolumn:nnn

```

²⁶The option `small` of `nicematrix` changes (among other) the value of `\arraystretch`. This is done, of course, before the call of `{array}`.

²⁷The user will probably not employ directly `\ialign` in the array... but more likely environments that utilize `\ialign` internally (e.g.: `{substack}`).

```

507 \cs_set_eq:NN \Block \@@_Block:
508 \cs_set_eq:NN \OnlyMainNiceMatrix \@@_OnlyMainNiceMatrix:n
509 \bool_if:NT \l_@@_renew_dots_bool
510 {
511   \cs_set_eq:NN \ldots \@@_Ldots
512   \cs_set_eq:NN \cdots \@@_Cdots
513   \cs_set_eq:NN \vdots \@@_Vdots
514   \cs_set_eq:NN \ddots \@@_Ddots
515   \cs_set_eq:NN \iddots \@@_Iddots
516   \cs_set_eq:NN \dots \@@_Ldots
517   \cs_set_eq:NN \hdotsfor \@@_Hdotsfor:
518 }

```

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

```

519 \seq_gclear_new:N \g_@@_multicolumn_cells_seq
520 \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).

```

521 \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`).

```

522 \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.

```

523 \int_gzero_new:N \g_@@_col_total_int
524 \cs_set_eq:NN \@ifnextchar \new@ifnextchar

```

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

```

525 \cs_set_eq:NN \NC@find@w \relax
526 \cs_set_eq:NN \NC@find@W \relax
527 \@@_renewcolumnntype:nn w { }
528 \@@_renewcolumnntype:nn W { \cs_set_eq:NN \hss \hfil }

```

By default, the letter used to specify a dotted line in the preamble of an environment of `nicematrix` (for example in `{pNiceArray}`) is the letter `.`. However, this letter is used by some extensions, for example `arydshln`. That’s why it’s possible to change the letter used by `nicematrix` with the option `letter-for-dotted-lines` which changes the value of `\l_@@_letter_for_dotted_lines_str`. We rescan this string (which is always of length 1) in particular for the case where `pdflatex` is used with `french-babel` (the colon is activated by `french-babel` at the beginning of the document).

```

529 \tl_set_rescan:Nno
530 \l_@@_letter_for_dotted_lines_str { } \l_@@_letter_for_dotted_lines_str
531 \exp_args:NV \newcolumnntype \l_@@_letter_for_dotted_lines_str
532 {
533   !
534   {
535     \skip_horizontal:n { 0.53 pt }

```

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

```

536 \bool_if:nF
537 {
538   \l_@@_auto_columns_width_bool
539   || \dim_compare_p:nNn \l_@@_columns_width_dim > \c_zero_dim

```



```

540     }
541     { \bool_gset_true:N \g_@@_large_nodes_bool }

```

Consider the following code:

```

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

```

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

```

542     \int_compare:nNnT \c@jCol > \g_@@_last_vdotted_col_int
543     {
544         \int_gset_eq:NN \g_@@_last_vdotted_col_int \c@jCol
545         \tl_gput_right:Nx \g_@@_code_after_tl

```

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

```

546         { \@@_vdottedline:n { \int_use:N \c@jCol } }
547     }
548 }
549 }
550 \int_gzero_new:N \g_@@_last_vdotted_col_int
551 \bool_if:NT \c_@@_siunitx_loaded_bool \@@_renew_NC@rewrite@S:
552 \int_gset:Nn \g_@@_last_vdotted_col_int { -1 }
553 \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.

```

554 \tl_gclear_new:N \g_@@_Cdots_lines_tl
555 \tl_gclear_new:N \g_@@_Ldots_lines_tl
556 \tl_gclear_new:N \g_@@_Vdots_lines_tl
557 \tl_gclear_new:N \g_@@_Ddots_lines_tl
558 \tl_gclear_new:N \g_@@_Iddots_lines_tl
559 \tl_gclear_new:N \g_@@_Hdotsfor_lines_tl
560 }

```

14.5 The environment `{NiceArrayWithDelims}`

```

561 \NewDocumentEnvironment { NiceArrayWithDelims } { m m O { } m ! O { } }
562 {
563     \str_if_empty:NT \g_@@_type_env_str
564     {
565         \str_gset:Nn \g_@@_type_env_str
566         { environment ~ { NiceArrayWithDelims } }
567     }
568     \@@_adapt_S_column:
569     \@@_test_if_math_mode:
570     \bool_if:NT \l_@@_in_env_bool { \@@_fatal:n { Yet~in~env } }
571     \bool_set_true:N \l_@@_in_env_bool

```

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

```

572     \cs_if_exist:NT \tikz@library@external@loaded
573     {
574         \tikzset { external / export = false }

```

```

575     \cs_if_exist:NT \ifstandalone
576     { \tikzset { external / optimize = false } }
577 }

```

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

```

578 \int_gincr:N \g_@@_env_int
579 \bool_if:NF \l_@@_block_auto_columns_width_bool
580 { \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.

```

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

```

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

```

582 \bool_if:NTF \l_@@_NiceArray_bool
583 { \keys_set:nn { NiceMatrix / NiceArray } }
584 { \keys_set:nn { NiceMatrix / pNiceArray } }
585 { #3 , #5 }

```

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

```

586 \int_compare:nNnT \l_@@_last_row_int = { -1 }
587 {
588     \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.

```

589 \str_if_empty:NTF \l_@@_name_str
590 {
591     \cs_if_exist:cT { @@_last_row_ \int_use:N \g_@@_env_int }
592     {
593         \int_set:Nn \l_@@_last_row_int
594         { \use:c { @@_last_row_ \int_use:N \g_@@_env_int } }
595     }
596 }
597 {
598     \cs_if_exist:cT { @@_last_row_ \l_@@_name_str }
599     {
600         \int_set:Nn \l_@@_last_row_int
601         { \use:c { @@_last_row_ \l_@@_name_str } }
602     }
603 }
604 }

```

The code in `\@@_pre_array:` is used only by `{NiceArrayWithDelims}`. It exists only for historical reasons. We should change that.

```

605 \@@_pre_array:

```

We compute the width of the two delimiters.

```

606 \dim_gzero_new:N \g_@@_left_delim_dim
607 \dim_gzero_new:N \g_@@_right_delim_dim
608 \bool_if:NTF \l_@@_NiceArray_bool
609 {
610     \dim_gset:Nn \g_@@_left_delim_dim { 2 \arraycolsep }
611     \dim_gset:Nn \g_@@_right_delim_dim { 2 \arraycolsep }
612 }
613 {
614     \group_begin:
615     \dim_set_eq:NN \nulldelimiterspace \c_zero_dim
616     \hbox_set:Nn \l_tmpa_box
617     {
618         \c_math_toggle_token
619         \left #1 \vcenter to 3 cm { } \right.
620         \c_math_toggle_token
621     }
622     \dim_gset:Nn \g_@@_left_delim_dim { \box_wd:N \l_tmpa_box }

```

```

623     \hbox_set:Nn \l_tmpa_box
624     {
625         \dim_set_eq:NN \nulldelimiterspace \c_zero_dim
626         \c_math_toggle_token
627         \left. \vcenter to 3 cm { } \right #2
628         \c_math_toggle_token
629     }
630     \dim_gset:Nn \g_@@_right_delim_dim { \box_wd:N \l_tmpa_box }
631     \group_end:
632 }

```

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

```

633     \box_clear_new:N \l_@@_the_array_box

```

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

```

634     \tl_set:Nn \l_tmpa_tl { #4 }
635     \int_compare:nNnTF \l_@@_first_col_int = 0
636     { \tl_put_left:NV \l_tmpa_tl \c_@@_preamble_first_col_tl }
637     {
638         \bool_if:NT \l_@@_NiceArray_bool
639         {
640             \bool_if:NF \l_@@_exterior_arraycolsep_bool
641             { \tl_put_left:Nn \l_tmpa_tl { @ { } } }
642         }
643     }
644     \int_compare:nNnTF \l_@@_last_col_int > { -1 }
645     { \tl_put_right:NV \l_tmpa_tl \c_@@_preamble_last_col_tl }
646     {
647         \bool_if:NT \l_@@_NiceArray_bool
648         {
649             \bool_if:NF \l_@@_exterior_arraycolsep_bool
650             { \tl_put_right:Nn \l_tmpa_tl { @ { } } }
651         }
652     }

```

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).

```

653     \hbox_set:Nw \l_@@_the_array_box
654     \skip_horizontal:n \l_@@_left_margin_dim
655     \skip_horizontal:n \l_@@_extra_left_margin_dim
656     \c_math_toggle_token
657     \bool_if:NTF \l_@@_light_syntax_bool
658     { \begin { @@-light-syntax } }
659     { \begin { @@-normal-syntax } }
660 }
661 {
662     \bool_if:NTF \l_@@_light_syntax_bool
663     { \end { @@-light-syntax } }
664     { \end { @@-normal-syntax } }
665     \c_math_toggle_token
666     \skip_horizontal:n \l_@@_right_margin_dim
667     \skip_horizontal:n \l_@@_extra_right_margin_dim
668     \hbox_set_end:

```

Now, the box containing the array is constructed (in `\l_@@_the_array_box`). However, we have some computations to do before inserting that box in the TeX flow (mainly because the exterior columns are in overlapping positions).

```

669     \int_compare:nNnT \l_@@_last_row_int > { -2 }
670     {
671         \bool_if:NF \l_@@_last_row_without_value_bool
672         {

```

```

673         \int_compare:nNnF \l_@@_last_row_int = \c@iRow
674         {
675             \@@_error:n { Wrong~last~row }
676             \int_gset_eq:NN \l_@@_last_row_int \c@iRow
677         }
678     }
679 }

```

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. 48).

```

680     \int_compare:nNnT \l_@@_first_col_int = 0
681     {
682         \skip_horizontal:n \arraycolsep
683         \skip_horizontal:n \g_@@_width_first_col_dim
684     }

```

The construction of the real box is different in `{NiceArray}` and in its variants (`{pNiceArray}`, etc.) because, in `{NiceArray}`, we have to take into account the option of position (t, c or b). We begin with `{NiceArray}`.

```

685     \bool_if:NTF \l_@@_NiceArray_bool
686     {
687         \int_compare:nNnTF \l_@@_first_row_int = 0
688         {
689             \str_if_eq:VnT \l_@@_pos_env_str { t }
690             {
691                 \box_move_up:nn { \g_@@_dp_row_zero_dim + \g_@@_ht_row_one_dim }
692             }
693         }
694         {
695             \int_compare:nNnT \l_@@_last_row_int > 0
696             {
697                 \str_if_eq:VnT \l_@@_pos_env_str { b }
698                 {
699                     \box_move_down:nn
700                     { \g_@@_ht_last_row_dim + \g_@@_dp_ante_last_row_dim }
701                 }
702             }
703         }
704         { \box_use_drop:N \l_@@_the_array_box }
705     }

```

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).

```

706     {
707         \int_compare:nNnTF \l_@@_first_row_int = 0
708         {
709             \dim_set:Nn \l_tmpa_dim
710             { \g_@@_dp_row_zero_dim + \g_@@_ht_row_zero_dim }
711         }
712         { \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”.²⁸

```

713         \int_compare:nNnTF \l_@@_last_row_int > { -2 }
714         {
715             \dim_set:Nn \l_tmpb_dim
716             { \g_@@_ht_last_row_dim + \g_@@_dp_last_row_dim }
717         }
718         { \dim_zero:N \l_tmpb_dim }
719         \hbox_set:Nn \l_tmpa_box
720         {

```

²⁸A value of `-1` for `\l_@@_last_row_int` means that there is a “last row” but the number of that row is unknown (the user have not set the value with the option `last row`).

```

721         \c_math_toggle_token
722         \left #1
723         \vcenter
724         {

```

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

```

725         \skip_vertical:n { - \l_tmpa_dim }
726         \hbox:n
727         {
728             \skip_horizontal:n { - \arraycolsep }
729             \box_use_drop:N \l_@@_the_array_box
730             \skip_horizontal:n { - \arraycolsep }
731         }

```

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

```

732         \skip_vertical:n { - \l_tmpb_dim }
733     }
734     \right #2
735     \c_math_toggle_token
736 }
737 \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }
738 \box_set_dp:Nn \l_tmpa_box { \box_dp:N \l_tmpa_box + \l_tmpb_dim }
739 \box_use_drop:N \l_tmpa_box
740 }

```

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. 49).

```

741 \bool_if:NT \g_@@_last_col_found_bool
742 {
743     \skip_horizontal:n \g_@@_width_last_col_dim
744     \skip_horizontal:n \arraycolsep
745 }
746 \@@_after_array:
747 }

```

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

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).

```

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

```

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

```

749 { \exp_args:NV \@@_array: \l_tmpa_tl }
750 {

```

If all the columns must have the same width (if the user has used the option `columns-width` or the option `auto-column-width` of the environment `{NiceMatrixBlock}`), we will add a row in the array to fix the width of the columns and construct the “col” nodes `nm-a-col-j` (these nodes will be used by the horizontal open dotted lines and by the commands `\@@_vdottedline:n`). We have written a dedicated function for that job.

```

751     \@@_create_row_of_col_nodes:
752     \endarray
753 }

```

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`).

```

754 \NewDocumentEnvironment { @@-light-syntax } { b }
755 {

```

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

```

756 \exp_args:NV \@@_array: \l_tmpa_tl

```

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

```

757 \seq_gclear_new:N \g_@@_rows_seq
758 \tl_set_rescan:Nno \l_@@_end_of_row_tl { } \l_@@_end_of_row_tl
759 \exp_args:NNV \seq_gset_split:Nnn \g_@@_rows_seq \l_@@_end_of_row_tl { #1 }

```

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

```

760 \seq_gpop_left:NN \g_@@_rows_seq \l_tmpa_tl
761 \exp_args:NV \@@_line_with_light_syntax_i:n \l_tmpa_tl
762 \seq_map_function:NN \g_@@_rows_seq \@@_line_with_light_syntax:n
763 \@@_create_row_of_col_nodes:
764 \endarray
765 }

```

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`.

```

766 { }
767 \cs_new_protected:Npn \@@_line_with_light_syntax_i:n #1
768 {
769   \seq_gclear_new:N \g_@@_cells_seq
770   \seq_gset_split:Nnn \g_@@_cells_seq { ~ } { #1 }
771   \seq_gpop_left:NN \g_@@_cells_seq \l_tmpa_tl
772   \l_tmpa_tl
773   \seq_map_function:NN \g_@@_cells_seq \@@_cell_with_light_syntax:n
774 }
775 \cs_new_protected:Npn \@@_line_with_light_syntax:n #1
776 {
777   \tl_if_empty:nF { #1 }
778   { \ \ \@@_line_with_light_syntax_i:n { #1 } }
779 }
780 \cs_new_protected:Npn \@@_cell_with_light_syntax:n #1 { & #1 }

```

The command `\@@_create_row_of_col_nodes:` will construct the potential last row. That last row (when it is created) is a false row used to fix the width of the columns (when the array is constructed with an option which specify the width of the columns) and create the `col`-nodes (that nodes will be used by `\vdottedline` for instance).

```

781 \cs_new:Npn \@@_create_row_of_col_nodes:
782 {
783   \bool_if:nT
784   {
785     \l_@@_auto_columns_width_bool
786     || \dim_compare_p:nNn \l_@@_columns_width_dim > \c_zero_dim
787   }
788   {
789     \crrc
790     \int_compare:nNnT \l_@@_first_col_int = 0 { \omit & }
791     \omit

```

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

```

792   \skip_horizontal:N \arraycolsep
793   \tikz [ remember~picture , overlay ]
794   \coordinate [ name = nm - \int_use:N \g_@@_env_int - col - 0 ] ;
795   \skip_horizontal:n { - \arraycolsep }

```

We compute in `\g_tmpa_dim` the common width of the columns. We use a global variable because we are in a cell of an `\halign` and because we have to use this variable in other cells (of the same row). The affectation of `\g_tmpa_dim`, like all the affectations, must be done after the `\omit` of the cell.

```

796   \bool_if:nTF
797   {
798     \l_@@_auto_columns_width_bool
799     && ! \l_@@_block_auto_columns_width_bool

```

```

800     }
801     {
802         \dim_gset:Nn \g_tmpa_dim
803         { \g_@@_max_cell_width_dim + 2 \arraycolsep }
804     }
805     {
806         \dim_gset:Nn \g_tmpa_dim
807         { \l_@@_columns_width_dim + 2 \arraycolsep }
808     }
809     \skip_horizontal:N \g_tmpa_dim
810     \tikz [ remember~picture , overlay ]
811     \coordinate [ name = nm - \int_use:N \g_@@_env_int - col - 1 ] ;

```

We begin a loop over the columns. The integer `\g_tmpa_int` will be the number of the current column. This integer is not used to fix the width of the column (since all the columns have the same width equal to `\g_@@_tmpa_dim`) but for the Tikz nodes.

```

812     \int_gset:Nn \g_tmpa_int 1
813     \bool_if:nTF \g_@@_last_col_found_bool
814     { \prg_replicate:nn { \g_@@_col_total_int - 3 } }
815     { \prg_replicate:nn { \g_@@_col_total_int - 2 } }
816     {
817         &
818         \omit

```

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

```

819         \int_gincr:N \g_tmpa_int
820         \skip_horizontal:N \g_tmpa_dim

```

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

```

821         \@@_create_col_node:
822     }

```

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

```

823     &
824     \omit
825     \int_gincr:N \g_tmpa_int
826     \skip_horizontal:N \g_tmpa_dim
827     \skip_horizontal:n { - \arraycolsep }
828     \@@_create_col_node:
829     \skip_horizontal:N \arraycolsep
830 }
831 }

```

```

832 \cs_new_protected:Npn \@@_create_col_node:
833 {
834     \tikz [ remember~picture , overlay ]
835     \coordinate
836     [
837         name = nm - \int_use:N \g_@@_env_int -
838         col - \int_use:N \g_tmpa_int
839     ] ;
840 }

```

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

```

841 \tl_const:Nn \c_@@_preamble_first_col_tl
842 {
843     >
844     {
845         \@@_begin_of_row:

```

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

```

846         \hbox_set:Nw \l_tmpa_box
847         \c_math_toggle_token
848         \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".

```

849     \bool_if:nT
850     {
851         \int_compare_p:nNn \c@iRow > 0
852         &&
853         (
854             \int_compare_p:nNn \l_@@_last_row_int < 0
855             ||
856             \int_compare_p:nNn \c@iRow < \l_@@_last_row_int
857         )
858     }
859     { \l_@@_code_for_first_col_tl }
860 }
861 l
862 <
863 {
864     \c_math_toggle_token
865     \hbox_set_end:
866     \@@_actualization_for_first_and_last_row:

```

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

```

867     \dim_gset:Nn \g_@@_width_first_col_dim
868     {
869         \dim_max:nn
870         \g_@@_width_first_col_dim
871         { \box_wd:N \l_tmpa_box }
872     }

```

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

```

873     \hbox_overlap_left:n
874     {
875         \tikz
876         [
877             remember~picture ,
878             inner~sep = \c_zero_dim ,
879             minimum~width = \c_zero_dim ,
880             baseline
881         ]
882         \node
883         [
884             anchor = base ,
885             name =
886             nm -
887             \int_use:N \g_@@_env_int -
888             \int_use:N \c@iRow -
889             0 ,
890             alias =
891             \str_if_empty:NF \l_@@_name_str
892             {
893                 \l_@@_name_str -
894                 \int_use:N \c@iRow -
895                 0
896             }
897         ]
898         { \box_use:N \l_tmpa_box } ;
899         \skip_horizontal:n
900         {
901             \g_@@_left_delim_dim +
902             \l_@@_left_margin_dim +
903             \l_@@_extra_left_margin_dim
904         }
905     }
906     \skip_horizontal:n { - 2 \arraycolsep }

```



```

907     }
908 }

```

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

```

909 \tl_const:Nn \c_@@_preamble_last_col_tl
910 {
911   >
912   {

```

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

```

913     \bool_gset_true:N \g_@@_last_col_found_bool
914     \int_gincr:N \c@jCol
915     \int_gset:Nn \g_@@_col_total_int
916     { \int_max:nn \g_@@_col_total_int \c@jCol }

```

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

```

917     \hbox_set:Nw \l_tmpa_box
918     \c_math_toggle_token
919     \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”.

```

920     \bool_if:nT
921     {
922       \int_compare_p:nNn \c@iRow > 0
923       &&
924       (
925         \int_compare_p:nNn \l_@@_last_row_int < 0
926         ||
927         \int_compare_p:nNn \c@iRow < \l_@@_last_row_int
928       )
929     }
930     { \l_@@_code_for_last_col_tl }
931   }
932   l
933   <
934   {
935     \c_math_toggle_token
936     \hbox_set_end:
937     \@@_actualization_for_first_and_last_row:

```

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

```

938     \dim_gset:Nn \g_@@_width_last_col_dim
939     {
940       \dim_max:nn
941       \g_@@_width_last_col_dim
942       { \box_wd:N \l_tmpa_box }
943     }
944     \skip_horizontal:n { - 2 \arraycolsep }

```

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

```

945     \hbox_overlap_right:n
946     {
947       \skip_horizontal:n
948       {
949         \g_@@_right_delim_dim +
950         \l_@@_right_margin_dim +
951         \l_@@_extra_right_margin_dim
952       }
953       \tikz
954       [
955         remember~picture ,
956         inner~sep = \c_zero_dim ,
957         minimum~width = \c_zero_dim ,
958         baseline

```

```

959         ]
960     \node
961     [
962         anchor = base ,
963         name =
964             nm -
965             \int_use:N \g_@@_env_int -
966             \int_use:N \c@iRow -
967             \int_use:N \c@jCol ,
968         alias =
969             \str_if_empty:NF \l_@@_name_str
970             {
971                 \l_@@_name_str -
972                 \int_use:N \c@iRow -
973                 \int_use:N \c@jCol
974             }
975     ]
976     { \box_use:N \l_tmpa_box } ;
977 }
978 }
979 }

```

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.

```

980 \NewDocumentEnvironment { NiceArray } { }
981 {
982     \bool_set_true:N \l_@@_NiceArray_bool
983     \str_if_empty:NT \g_@@_type_env_str
984     { \str_gset:Nn \g_@@_type_env_str { environment ~ { NiceArray } } }

```

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

```

985     \NiceArrayWithDelims . .
986 }
987 { \endNiceArrayWithDelims }

```

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

```

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

996 \NewDocumentEnvironment { bNiceArray } { }
997 {
998     \str_if_empty:NT \g_@@_type_env_str
999     { \str_gset:Nn \g_@@_type_env_str { environment ~ { bNiceArray } } }
1000     \@@_test_if_math_mode:
1001     \NiceArrayWithDelims [ ]
1002 }
1003 { \endNiceArrayWithDelims }

1004 \NewDocumentEnvironment { BNiceArray } { }
1005 {
1006     \str_if_empty:NT \g_@@_type_env_str
1007     { \str_gset:Nn \g_@@_type_env_str { environment ~ { BNiceArray } } }
1008     \@@_test_if_math_mode:
1009     \NiceArrayWithDelims \{ \}

```

```

1010 }
1011 { \endNiceArrayWithDelims }
1012 \NewDocumentEnvironment { vNiceArray } { }
1013 {
1014   \str_if_empty:NT \g_@@_type_env_str
1015     { \str_gset:Nn \g_@@_type_env_str { environment ~ { vNiceArray } } }
1016   \@@_test_if_math_mode:
1017   \NiceArrayWithDelims | |
1018 }
1019 { \endNiceArrayWithDelims }
1020 \NewDocumentEnvironment { VNiceArray } { }
1021 {
1022   \str_if_empty:NT \g_@@_type_env_str
1023     { \str_gset:Nn \g_@@_type_env_str { environment ~ { VNiceArray } } }
1024   \@@_test_if_math_mode:
1025   \NiceArrayWithDelims \| \|
1026 }
1027 { \endNiceArrayWithDelims }

```

14.6 The environment `{NiceMatrix}` and its variants

```

1028 \cs_new_protected:Npn \@@_define_env:n #1
1029 {
1030   \NewDocumentEnvironment { #1 NiceMatrix } { ! O { } }
1031   {
1032     \str_gset:Nn \g_@@_type_env_str { environment ~ { #1 NiceMatrix } }
1033     \keys_set:nn { NiceMatrix / NiceMatrix } { ##1 }
1034     \begin { #1 NiceArray }
1035       {
1036         *
1037         {
1038           \int_compare:nNnTF \l_@@_last_col_int = { -1 }
1039             \c@MaxMatrixCols
1040             { \int_eval:n { \l_@@_last_col_int - 1 } }
1041         }
1042         C
1043       }
1044     }
1045     { \end { #1 NiceArray } }
1046   }
1047   \@@_define_env:n { }
1048   \@@_define_env:n p
1049   \@@_define_env:n b
1050   \@@_define_env:n B
1051   \@@_define_env:n v
1052   \@@_define_env:n V

```

14.7 How to know whether a cell is “empty”

The conditionnal `\@@_if_not_empty_cell:nnT` tests whether a cell is empty. The first two arguments must be LaTeX3 counters for the row and the column of the considered cell.

```

1053 \prg_set_conditional:Npnn \@@_if_not_empty_cell:nn #1 #2 { T , TF }
1054 {

```

First, we want to test whether the cell is in the virtual sequence of “non-empty” cells. There are several important remarks:

- we don’t use a `expl3` sequence for efficiency;
- the “non-empty” cells in this sequence are not, in fact, all the non-empty cells of the array: on the contrary they are only cells declared as non-empty for a special reason (as of now, there are only cells which are on a dotted line which is already drawn or which will be drawn “just after”);

- the flag `\l_tmpa_bool` will be raised when the cell is actually on this virtual sequence.

```

1055 \bool_set_false:N \l_tmpa_bool
1056 \cs_if_exist:cTF
1057 { @@ _ dotted _ \int_use:N #1 - \int_use:N #2 }
1058 \prg_return_true:
1059 {

```

We know that the cell is not in the virtual sequence of the “non-empty” cells. Now, we test whether the cell is a “virtual cell”, that is to say a cell after the `\\` of the line of the array. It’s easy to know whether a cell is virtual: the cell is virtual if, and only if, the corresponding Tikz node doesn’t exist.

```

1060 \cs_if_free:cTF
1061 {
1062     pgf@sh@ns@nm -
1063     \int_use:N \g_@@_env_int -
1064     \int_use:N #1 -
1065     \int_use:N #2
1066 }
1067 { \prg_return_false: }
1068 {

```

Now, we want to test whether the cell is in the virtual sequence of “empty” cells. There are several important remarks:

- we don’t use a `expl3` sequence for efficiency ;
- the “empty” cells in this sequence are not, in fact, all the non-empty cells of the array: on the contrary they are only cells declared as non-empty for a special reason ;
- the flag `\l_tmpa_bool` will be raised when the cell is actually on this virtual sequence.

```

1069 \bool_set_false:N \l_tmpa_bool
1070 \cs_if_exist:cT
1071 { @@ _ empty _ \int_use:N #1 - \int_use:N #2 }
1072 {
1073     \int_compare:nNnT
1074     { \use:c { @@ _ empty _ \int_use:N #1 - \int_use:N #2 } }
1075     =
1076     \g_@@_env_int
1077     { \bool_set_true:N \l_tmpa_bool }
1078 }
1079 \bool_if:NNTF \l_tmpa_bool
1080 \prg_return_false:

```

In the general case, we consider the width of the Tikz node corresponding to the cell. In order to compute this width, we have to extract the coordinate of the west and east anchors of the node. This extraction needs a command environment `{pgfpicture}` but, in fact, nothing is drawn.

```

1081 {
1082     \begin { pgfpicture }

```

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

```

1083 \tl_set:Nx \l_tmpa_tl
1084 {
1085     nm -
1086     \int_use:N \g_@@_env_int -
1087     \int_use:N #1 -
1088     \int_use:N #2
1089 }
1090 \pgfpointanchor \l_tmpa_tl { east }
1091 \dim_gset:Nn \g_tmpa_dim \pgf@x
1092 \pgfpointanchor \l_tmpa_tl { west }
1093 \dim_gset:Nn \g_tmpb_dim \pgf@x
1094 \end { pgfpicture }
1095 \dim_compare:nNnTF
1096 { \dim_abs:n { \g_tmpb_dim - \g_tmpa_dim } } < { 0.5 pt }
1097 \prg_return_false:

```

```

1098         \prg_return_true:
1099     }
1100 }
1101 }
1102 }

```

14.8 After the construction of the array

```

1103 \cs_new_protected:Nn \@@_after_array:
1104 {
1105     \int_compare:nNnTF \c@iRow > 0
1106         \@@_after_array_i:
1107     {
1108         \@@_error:n { Zero~row }
1109         \@@_restore_iRow_jCol:
1110     }
1111 }

```

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

```

1112 \cs_new_protected:Nn \@@_after_array_i:
1113 {
1114     \group_begin:
1115     \cs_if_exist:NT \tikz@library@external@loaded
1116         { \tikzset { external / export = false } }

```

Now, the definition of `\c@jCol` and `\g_@@_col_total_int` change: `\c@jCol` will be the number of columns without the “last column”; `\g_@@_col_total_int` will be the number of columns with this “last column”.²⁹

```

1117     \int_gset_eq:NN \c@jCol \g_@@_col_total_int
1118     \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.

```

1119     \int_gset_eq:NN \g_@@_row_total_int \c@iRow
1120     \int_compare:nNnT \l_@@_last_row_int > { -1 }
1121         { \int_gsub:Nn \c@iRow \c_one_int }

```

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

```

1122     \bool_if:NT \l_@@_last_row_without_value_bool
1123     {
1124         \iow_now:Nn \@mainaux \ExplSyntaxOn
1125         \iow_now:Nx \@mainaux
1126         {
1127             \cs_gset:cpn { @@_last_row_ \int_use:N \g_@@_env_int }
1128             { \int_use:N \g_@@_row_total_int }
1129         }

```

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.

```

1130         \str_if_empty:NF \l_@@_name_str
1131         {
1132             \iow_now:Nx \@mainaux
1133             {
1134                 \cs_gset:cpn { @@_last_row_ \l_@@_name_str }
1135                 { \int_use:N \g_@@_row_total_int }
1136             }
1137         }
1138         \iow_now:Nn \@mainaux \ExplSyntaxOff
1139     }

```

²⁹We remind that the potential “first column” has the number 0.

By default, the diagonal lines will be parallelized³⁰. 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.

```

1140   \bool_if:NT \l_@@_parallelize_diags_bool
1141   {
1142     \int_zero_new:N \l_@@_ddots_int
1143     \int_zero_new:N \l_@@_iddots_int

```

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

```

1144     \dim_zero_new:N \l_@@_delta_x_one_dim
1145     \dim_zero_new:N \l_@@_delta_y_one_dim
1146     \dim_zero_new:N \l_@@_delta_x_two_dim
1147     \dim_zero_new:N \l_@@_delta_y_two_dim
1148   }

```

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

```

1149   \bool_if:nTF \g_@@_medium_nodes_bool
1150   {
1151     \bool_if:nTF \g_@@_large_nodes_bool
1152     \@@_create_medium_and_large_nodes:
1153     \@@_create_medium_nodes:
1154   }
1155   { \bool_if:NT \g_@@_large_nodes_bool \@@_create_large_nodes: }
1156   \int_zero_new:N \l_@@_initial_i_int
1157   \int_zero_new:N \l_@@_initial_j_int
1158   \int_zero_new:N \l_@@_final_i_int
1159   \int_zero_new:N \l_@@_final_j_int
1160   \bool_set_false:N \l_@@_initial_open_bool
1161   \bool_set_false:N \l_@@_final_open_bool

```

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

```

1162   \bool_if:NT \l_@@_small_bool
1163   {
1164     \dim_set:Nn \l_@@_radius_dim { 0.37 pt }
1165     \dim_set:Nn \l_@@_inter_dots_dim { 0.25 em }
1166   }

```

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

```

1167   \g_@@_Hdotsfor_lines_tl
1168   \g_@@_Vdots_lines_tl
1169   \g_@@_Ddots_lines_tl
1170   \g_@@_Iddots_lines_tl
1171   \g_@@_Cdots_lines_tl
1172   \g_@@_Ldots_lines_tl

```

Now, the code-after.

```

1173   \tikzset
1174   {
1175     every~picture / .style =
1176     {
1177       overlay ,
1178       remember~picture ,

```

³⁰It's possible to use the option `parallelize-diags` to disable this parallelization.

```

1179         name-prefix = nm - \int_use:N \g_@@_env_int -
1180     }
1181 }
1182 \cs_set_eq:NN \line \@@_line:nn
1183 \g_@@_code_after_tl
1184 \tl_gclear:N \g_@@_code_after_tl
1185 \group_end:
1186 \str_gclear:N \g_@@_type_env_str
1187 \@@_restore_iRow_jCol:
1188 }
1189 \cs_new_protected:Nn \@@_restore_iRow_jCol:
1190 {
1191     \cs_if_exist:NT \theiRow { \int_gset_eq:NN \c@iRow \l_@@_save_iRow_int }
1192     \cs_if_exist:NT \thejCol { \int_gset_eq:NN \c@jCol \l_@@_save_jCol_int }
1193 }

```

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

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

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

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

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.

```

1194 \cs_new_protected:Nn \@@_find_extremities_of_line:nnnn
1195 {

```

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

```

1196     \cs_set:cpn { @@ _ dotted _ #1 - #2 } { }

```

Initialization of variables.

```

1197     \int_set:Nn \l_@@_initial_i_int { #1 }
1198     \int_set:Nn \l_@@_initial_j_int { #2 }
1199     \int_set:Nn \l_@@_final_i_int { #1 }
1200     \int_set:Nn \l_@@_final_j_int { #2 }

```

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

```

1201 \bool_set_false:N \l_@@_stop_loop_bool
1202 \bool_do_until:Nn \l_@@_stop_loop_bool
1203 {
1204   \int_add:Nn \l_@@_final_i_int { #3 }
1205   \int_add:Nn \l_@@_final_j_int { #4 }

```

We test if we are still in the matrix.

```

1206 \bool_set_false:N \l_@@_final_open_bool
1207 \int_compare:nNnTF \l_@@_final_i_int > \c@iRow
1208 {
1209   \int_compare:nNnT { #3 } = 1
1210   { \bool_set_true:N \l_@@_final_open_bool }
1211 }
1212 {
1213   \int_compare:nNnTF \l_@@_final_j_int < 1
1214   {
1215     \int_compare:nNnT { #4 } = { -1 }
1216     { \bool_set_true:N \l_@@_final_open_bool }
1217   }
1218   {
1219     \int_compare:nNnT \l_@@_final_j_int > \c@jCol
1220     {
1221       \int_compare:nNnT { #4 } = 1
1222       { \bool_set_true:N \l_@@_final_open_bool }
1223     }
1224   }
1225 }
1226 \bool_if:NTF \l_@@_final_open_bool

```

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

```

1227 {

```

We do a step backwards because we will draw the dotted line upon the last cell in the matrix (we will use the “medium node” of this cell).

```

1228 \int_sub:Nn \l_@@_final_i_int { #3 }
1229 \int_sub:Nn \l_@@_final_j_int { #4 }
1230 \bool_set_true:N \l_@@_stop_loop_bool
1231 }

```

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`.

```

1232 {
1233   \@@_if_not_empty_cell:nnTF \l_@@_final_i_int \l_@@_final_j_int
1234   { \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.

```

1235 {
1236   \cs_set:cpn
1237   {
1238     @@ _ dotted _
1239     \int_use:N \l_@@_final_i_int -
1240     \int_use:N \l_@@_final_j_int
1241   }
1242   { }
1243 }
1244 }
1245 }

```

We test whether the initial extremity of the dotted line is an implicit cell already dotted (by another dotted line). In this case, we can't draw the line because we have no Tikz node at the extremity

of the arrow (and we can't use the "medium node" or the "large node" because we should use the normal node since the extremity is not open).

```

1246 \cs_if_free:cT
1247 {
1248   pgf@sh@ns@nm -
1249   \int_use:N \g_@@_env_int -
1250   \int_use:N \l_@@_final_i_int -
1251   \int_use:N \l_@@_final_j_int
1252 }
1253 {
1254   \bool_if:NF \l_@@_final_open_bool
1255   {
1256     \msg_error:nmx { nicematrix } { Impossible~line }
1257     { \int_use:N \l_@@_final_i_int }
1258     \bool_set_true:N \l_@@_impossible_line_bool
1259   }
1260 }

```

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

```

1261 \bool_set_false:N \l_@@_stop_loop_bool
1262 \bool_do_until:Nn \l_@@_stop_loop_bool
1263 {
1264   \int_sub:Nn \l_@@_initial_i_int { #3 }
1265   \int_sub:Nn \l_@@_initial_j_int { #4 }
1266   \bool_set_false:N \l_@@_initial_open_bool
1267   \int_compare:nNnTF \l_@@_initial_i_int < 1
1268   {
1269     \int_compare:nNnT { #3 } = 1
1270     { \bool_set_true:N \l_@@_initial_open_bool }
1271   }
1272   {
1273     \int_compare:nNnTF \l_@@_initial_j_int < 1
1274     {
1275       \int_compare:nNnT { #4 } = 1
1276       { \bool_set_true:N \l_@@_initial_open_bool }
1277     }
1278     {
1279       \int_compare:nNnT \l_@@_initial_j_int > \c@jCol
1280       {
1281         \int_compare:nNnT { #4 } = { -1 }
1282         { \bool_set_true:N \l_@@_initial_open_bool }
1283       }
1284     }
1285   }
1286   \bool_if:NTF \l_@@_initial_open_bool
1287   {
1288     \int_add:Nn \l_@@_initial_i_int { #3 }
1289     \int_add:Nn \l_@@_initial_j_int { #4 }
1290     \bool_set_true:N \l_@@_stop_loop_bool
1291   }
1292   {
1293     \@@_if_not_empty_cell:nnTF
1294     \l_@@_initial_i_int \l_@@_initial_j_int
1295     { \bool_set_true:N \l_@@_stop_loop_bool }
1296     {
1297       \cs_set:cpn
1298       {
1299         @@ _ dotted _
1300         \int_use:N \l_@@_initial_i_int -
1301         \int_use:N \l_@@_initial_j_int
1302       }

```

```

1303         { }
1304     }
1305 }
1306 }

```

We test whether the initial extremity of the dotted line is an implicit cell already dotted (by another dotted line). In this case, we can't draw the line because we have no Tikz node at the extremity of the arrow (and we can't use the “medium node” or the “large node” because we should use the normal node since the extremity is not open).

```

1307 \cs_if_free:cT
1308 {
1309     pgf@sh@ns@nm -
1310     \int_use:N \g_@@_env_int -
1311     \int_use:N \l_@@_initial_i_int -
1312     \int_use:N \l_@@_initial_j_int
1313 }
1314 {
1315     \bool_if:NF \l_@@_initial_open_bool
1316     {
1317         \msg_error:nmx { nicematrix } { Impossible~line }
1318         { \int_use:N \l_@@_initial_i_int }
1319         \bool_set_true:N \l_@@_impossible_line_bool
1320     }
1321 }

```

If we have at least one open extremity, we create the “medium nodes” in the matrix³¹. We remind that, when used once, the command `\@@_create_medium_nodes:` becomes no-op in the current TeX group.

```

1322 \bool_if:nT \l_@@_initial_open_bool \@@_create_medium_nodes:
1323 \bool_if:NT \l_@@_final_open_bool \@@_create_medium_nodes:
1324 }

```

The command `\@@_retrieve_coords:nn` retrieves the Tikz coordinates of the two extremities of the dotted line we will have to draw³². This command has four implicit arguments which are `\l_@@_initial_i_int`, `\l_@@_initial_j_int`, `\l_@@_final_i_int` and `\l_@@_final_j_int`. The two arguments of the command `\@@_retrieve_coords:nn` are the suffix and the anchor that must be used for the two nodes.

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

```

1325 \cs_new_protected:Nn \@@_retrieve_coords:nn
1326 {
1327     \dim_gzero_new:N \g_@@_x_initial_dim
1328     \dim_gzero_new:N \g_@@_y_initial_dim
1329     \dim_gzero_new:N \g_@@_x_final_dim
1330     \dim_gzero_new:N \g_@@_y_final_dim
1331     \begin { tikzpicture } [ remember~picture ]
1332         \tikz@parse@node \pgfutil@firstofone
1333         ( nm - \int_use:N \g_@@_env_int -
1334             \int_use:N \l_@@_initial_i_int -
1335             \int_use:N \l_@@_initial_j_int #1 )
1336         \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1337         \dim_gset:Nn \g_@@_y_initial_dim \pgf@y
1338         \tikz@parse@node \pgfutil@firstofone
1339         ( nm - \int_use:N \g_@@_env_int -
1340             \int_use:N \l_@@_final_i_int -
1341             \int_use:N \l_@@_final_j_int #2 )
1342         \dim_gset:Nn \g_@@_x_final_dim \pgf@x

```

³¹We should change this. Indeed, for an open extremity of an *horizontal* dotted line, we use the w-node, if, it exists, and not the “medium node”.

³²In fact, with diagonal lines, or vertical lines in columns of type L or R, an adjustment of one of the coordinates may be done.

```

1343     \dim_gset:Nn \g_@@_y_final_dim \pgf@y
1344   \end { tikzpicture }
1345 }
1346 \cs_generate_variant:Nn \@@_retrieve_coords:nn { x x }

```

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

```

1347 \cs_new_protected:Nn \@@_adjust_with_col_nodes:
1348 {
1349   \bool_if:NT \l_@@_initial_open_bool
1350   {
1351     \cs_if_exist:cT
1352     { pgf@sh@ns@nm - \int_use:N \g_@@_env_int - col - 0 }
1353     {
1354       \begin { tikzpicture } [ remember~picture ]
1355       \tikz@parse@node \pgfutil@firstofone
1356       ( nm - \int_use:N \g_@@_env_int - col - 0 )
1357       \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1358       \end { tikzpicture }
1359     }
1360   }
1361   \bool_if:NT \l_@@_final_open_bool
1362   {
1363     \cs_if_exist:cT
1364     {
1365       pgf@sh@ns@nm - \int_use:N \g_@@_env_int - col -
1366       \int_use:N \c@jCol
1367     }
1368     {
1369       \begin { tikzpicture } [ remember~picture ]
1370       \tikz@parse@node \pgfutil@firstofone
1371       ( nm - \int_use:N \g_@@_env_int - col - \int_use:N \c@jCol )
1372       \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1373       \end { tikzpicture }
1374     }
1375   }
1376 }

1377 \cs_new_protected:Nn \@@_draw_Ldots:nn
1378 {
1379   \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1380   {
1381     \bool_set_false:N \l_@@_impossible_line_bool
1382     \@@_find_extremities_of_line:nnnn { #1 } { #2 } 0 \c_one_int
1383     \bool_if:NF \l_@@_impossible_line_bool \@@_actually_draw_Ldots:
1384   }
1385 }

```

The command `\@@_actually_draw_Ldots:` draws the `Ldots` line using `\l_@@_initial_i_int`, `\l_@@_initial_j_int`, `\l_@@_initial_open_bool`, `\l_@@_final_i_int`, `\l_@@_final_j_int` and `\l_@@_final_open_bool`. We have a dedicated command because it is used also by `\Hdotsfor`.

```

1386 \cs_new_protected:Nn \@@_actually_draw_Ldots:
1387 {
1388   \@@_retrieve_coords:xx
1389   {
1390     \bool_if:NTF \l_@@_initial_open_bool
1391     {

```

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

```

1392     \cs_if_exist:cTF
1393     {
1394         pgf@sh@ns@nm
1395         - \int_use:N \g_@@_env_int
1396         - \int_use:N \l_@@_initial_i_int
1397         - \int_use:N \l_@@_initial_j_int - w
1398     }
1399     { - w.base~west }
1400     { - medium.base~west }
1401 }
1402 { .base~east }
1403 }
1404 {
1405     \bool_if:NTF \l_@@_final_open_bool
1406     {
1407         \cs_if_exist:cTF
1408         {
1409             pgf@sh@ns@nm
1410             - \int_use:N \g_@@_env_int
1411             - \int_use:N \l_@@_final_i_int
1412             - \int_use:N \l_@@_final_j_int - w
1413         }
1414         { - w.base~east }
1415         { - medium.base~east }
1416     }
1417     { .base~west }
1418 }
1419 \@@_adjust_with_col_nodes:
1420 \bool_if:NT \l_@@_initial_open_bool
1421 { \dim_gset_eq:NN \g_@@_y_initial_dim \g_@@_y_final_dim }
1422 \bool_if:NT \l_@@_final_open_bool
1423 { \dim_gset_eq:NN \g_@@_y_final_dim \g_@@_y_initial_dim }

```

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

```

1424     \dim_gadd:Nn \g_@@_y_initial_dim { 0.53 pt }
1425     \dim_gadd:Nn \g_@@_y_final_dim { 0.53 pt }
1426     \@@_draw_tikz_line:
1427 }

1428 \cs_new_protected:Nn \@@_draw_Cdots:nn
1429 {
1430     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1431     {
1432         \bool_set_false:N \l_@@_impossible_line_bool
1433         \@@_find_extremities_of_line:nnnn { #1 } { #2 } 0 \c_one_int
1434         \bool_if:NF \l_@@_impossible_line_bool
1435         {
1436             \@@_retrieve_coords:xx
1437             {
1438                 \bool_if:NTF \l_@@_initial_open_bool
1439                 {
1440                     \cs_if_exist:cTF
1441                     {
1442                         pgf@sh@ns@nm
1443                         - \int_use:N \g_@@_env_int
1444                         - \int_use:N \l_@@_initial_i_int
1445                         - \int_use:N \l_@@_initial_j_int - w
1446                     }
1447                     { - w.mid~west }
1448                     { - medium.mid~west }
1449                 }

```

```

1450         { .mid~east }
1451     }
1452     {
1453         \bool_if:NTF \l_@@_final_open_bool
1454         {
1455             \cs_if_exist:cTF
1456             {
1457                 pgf@sh@ns@nm
1458                 - \int_use:N \g_@@_env_int
1459                 - \int_use:N \l_@@_final_i_int
1460                 - \int_use:N \l_@@_final_j_int - w
1461             }
1462             { - w.mid~east }
1463             { - medium.mid~east }
1464         }
1465         { .mid~west }
1466     }
1467     \@@_adjust_with_col_nodes:
1468     \bool_if:NT \l_@@_initial_open_bool
1469     { \dim_gset_eq:NN \g_@@_y_initial_dim \g_@@_y_final_dim }
1470     \bool_if:NT \l_@@_final_open_bool
1471     { \dim_gset_eq:NN \g_@@_y_final_dim \g_@@_y_initial_dim }
1472     \@@_draw_tikz_line:
1473 }
1474 }
1475 }

```

For the vertical dots, we have to distinguish different instances because we want really vertical lines. Be careful: it's not possible to insert the command `\@@_retrieve_coords:nn` in the arguments T and F of the `expl3` commands (why?).

```

1476 \cs_new_protected:Nn \@@_draw_Vdots:nn
1477 {
1478     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1479     {
1480         \bool_set_false:N \l_@@_impossible_line_bool
1481         \@@_find_extremities_of_line:nnnn { #1 } { #2 } \c_one_int 0
1482         \bool_if:NF \l_@@_impossible_line_bool
1483         {
1484             \@@_retrieve_coords:xx
1485             {
1486                 \bool_if:NTF \l_@@_initial_open_bool
1487                 { - medium.north~west }
1488                 { .south~west }
1489             }
1490             {
1491                 \bool_if:NTF \l_@@_final_open_bool
1492                 { - medium.south~west }
1493                 { .north~west }
1494             }
1495         }
1496     }
1497 }

```

The boolean `\l_tmpa_bool` indicates whether the column is of type 1 (L of `{NiceArray}`) or may be considered as if.

```

1495     \bool_set:Nn \l_tmpa_bool
1496     { \dim_compare_p:nNn \g_@@_x_initial_dim = \g_@@_x_final_dim }
1497     \@@_retrieve_coords:xx
1498     {
1499         \bool_if:NTF \l_@@_initial_open_bool
1500         { - medium.north }
1501         { .south }
1502     }
1503     {
1504         \bool_if:NTF \l_@@_final_open_bool
1505         { - medium.south }
1506     }
1507 }

```

```

1506         { .north }
1507     }

```

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

```

1508     \bool_set:Nn \l_tmpb_bool
1509     { \dim_compare_p:nNn \g_@@_x_initial_dim = \g_@@_x_final_dim }
1510     \bool_if:NF \l_tmpb_bool
1511     {
1512         \dim_gset:Nn \g_@@_x_initial_dim
1513         {
1514             \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
1515             \g_@@_x_initial_dim \g_@@_x_final_dim
1516         }
1517         \dim_gset_eq:NN \g_@@_x_final_dim \g_@@_x_initial_dim
1518     }
1519     \@@_draw_tikz_line:
1520 }
1521 }
1522 }

```

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.

```

1523 \cs_new_protected:Nn \@@_draw_Ddots:nn
1524 {
1525     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1526     {
1527         \bool_set_false:N \l_@@_impossible_line_bool
1528         \@@_find_extremities_of_line:nnnn { #1 } { #2 } \c_one_int \c_one_int
1529         \bool_if:NF \l_@@_impossible_line_bool
1530         {
1531             \@@_retrieve_coords:xx
1532             {
1533                 \bool_if:NTF \l_@@_initial_open_bool
1534                 { - medium.north~west }
1535                 { .south~east }
1536             }
1537             {
1538                 \bool_if:NTF \l_@@_final_open_bool
1539                 { - medium.south~east }
1540                 { .north~west }
1541             }
1542         }
1543     }
1544 }

```

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

```

1542     \bool_if:NT \l_@@_parallelize_diags_bool
1543     {
1544         \int_incr:N \l_@@_ddots_int

```

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

```

1545         \int_compare:nNnTF \l_@@_ddots_int = \c_one_int

```

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

```

1546     {
1547         \dim_set:Nn \l_@@_delta_x_one_dim
1548         { \g_@@_x_final_dim - \g_@@_x_initial_dim }
1549         \dim_set:Nn \l_@@_delta_y_one_dim
1550         { \g_@@_y_final_dim - \g_@@_y_initial_dim }
1551     }

```

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

```

1552         {
1553             \dim_gset:Nn \g_@@_y_final_dim
1554             {
1555                 \g_@@_y_initial_dim +
1556                 ( \g_@@_x_final_dim - \g_@@_x_initial_dim ) *
1557                 \dim_ratio:nn \l_@@_delta_y_one_dim \l_@@_delta_x_one_dim
1558             }
1559         }
1560     }

```

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

```

1561         \@@_draw_tikz_line:
1562     }
1563 }
1564 }

```

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

```

1565 \cs_new_protected:Nn \@@_draw_Iddots:nn
1566 {
1567     \cs_if_free:cT { @@ _ dotted _ #1 - #2 }
1568     {
1569         \bool_set_false:N \l_@@_impossible_line_bool
1570         \@@_find_extremities_of_line:nnnn { #1 } { #2 } 1 { -1 }
1571         \bool_if:NF \l_@@_impossible_line_bool
1572         {
1573             \@@_retrieve_coords:xx
1574             {
1575                 \bool_if:NTF \l_@@_initial_open_bool
1576                 { - medium.north-east }
1577                 { .south~west }
1578             }
1579             {
1580                 \bool_if:NTF \l_@@_final_open_bool
1581                 { - medium.south-west }
1582                 { .north~east }
1583             }
1584             \bool_if:NT \l_@@_parallelize_diags_bool
1585             {
1586                 \int_incr:N \l_@@_iddots_int
1587                 \int_compare:nNnTF \l_@@_iddots_int = \c_one_int
1588                 {
1589                     \dim_set:Nn \l_@@_delta_x_two_dim
1590                     { \g_@@_x_final_dim - \g_@@_x_initial_dim }
1591                     \dim_set:Nn \l_@@_delta_y_two_dim
1592                     { \g_@@_y_final_dim - \g_@@_y_initial_dim }
1593                 }
1594                 {
1595                     \dim_gset:Nn \g_@@_y_final_dim
1596                     {
1597                         \g_@@_y_initial_dim +
1598                         ( \g_@@_x_final_dim - \g_@@_x_initial_dim ) *
1599                         \dim_ratio:nn \l_@@_delta_y_two_dim \l_@@_delta_x_two_dim
1600                     }
1601                 }
1602             }
1603             \@@_draw_tikz_line:
1604         }
1605     }
1606 }

```

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).

```
1607 \NewExpandableDocumentCommand \NiceMatrixLastEnv { }
1608 { \int_use:N \g_@@_env_int }
```

14.9 The actual instructions for drawing the dotted line with Tikz

The command `\@@_draw_tikz_line`: draws the line using four implicit arguments:

`\g_@@_x_initial_dim`, `\g_@@_y_initial_dim`, `\g_@@_x_final_dim` and `\g_@@_y_final_dim`. These variables are global for technical reasons: their first affectation was in an instruction `\tikz`.

```
1609 \cs_new_protected:Nn \@@_draw_tikz_line:
1610 {
```

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.

```
1611   \dim_zero_new:N \l_@@_l_dim
1612   \dim_set:Nn \l_@@_l_dim
1613   {
1614     \fp_to_dim:n
1615     {
1616       sqrt
1617       (
1618         ( \dim_use:N \g_@@_x_final_dim
1619           - \dim_use:N \g_@@_x_initial_dim
1620         ) ^ 2
1621         +
1622         ( \dim_use:N \g_@@_y_final_dim
1623           - \dim_use:N \g_@@_y_initial_dim
1624         ) ^ 2
1625       )
1626     }
1627   }
```

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

```
1628   \dim_compare:nNnF \l_@@_l_dim = \c_zero_dim
```

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

```
1629   {
1630     \bool_if:NTF \l_@@_initial_open_bool
1631     {
1632       \bool_if:NTF \l_@@_final_open_bool
1633       {
1634         \int_set:Nn \l_tmpa_int
1635         { \dim_ratio:nn \l_@@_l_dim \l_@@_inter_dots_dim }
1636       }
1637       {
1638         \int_set:Nn \l_tmpa_int
1639         {
1640           \dim_ratio:nn
1641           { \l_@@_l_dim - \l_@@_dotted_lines_margin_dim }
1642           \l_@@_inter_dots_dim
1643         }
1644       }
1645     }
1646     {
1647       \bool_if:NTF \l_@@_final_open_bool
1648       {
1649         \int_set:Nn \l_tmpa_int
```



```

1650         {
1651             \dim_ratio:nn
1652             { \l_@@_l_dim - \l_@@_dotted_lines_margin_dim }
1653             \l_@@_inter_dots_dim
1654         }
1655     }
1656     {
1657         \int_set:Nn \l_tmpa_int
1658         {
1659             \dim_ratio:nn
1660             { \l_@@_l_dim - ( \l_@@_dotted_lines_margin_dim * 2 ) }
1661             \l_@@_inter_dots_dim
1662         }
1663     }
1664 }

```

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

```

1665     \dim_set:Nn \l_tmpa_dim
1666     {
1667         ( \g_@@_x_final_dim - \g_@@_x_initial_dim ) *
1668         \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
1669     }
1670     \dim_set:Nn \l_tmpb_dim
1671     {
1672         ( \g_@@_y_final_dim - \g_@@_y_initial_dim ) *
1673         \dim_ratio:nn \l_@@_inter_dots_dim \l_@@_l_dim
1674     }

```

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`.

```

1675     \int_set:Nn \l_tmpb_int
1676     {
1677         \bool_if:NTF \l_@@_initial_open_bool
1678         { \bool_if:NTF \l_@@_final_open_bool 1 0 }
1679         { \bool_if:NTF \l_@@_final_open_bool 2 1 }
1680     }

```

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

```

1681     \dim_gadd:Nn \g_@@_x_initial_dim
1682     {
1683         ( \g_@@_x_final_dim - \g_@@_x_initial_dim ) *
1684         \dim_ratio:nn
1685         { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
1686         { \l_@@_l_dim * 2 }
1687         * \l_tmpb_int
1688     }

```

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

```

1689     \dim_gadd:Nn \g_@@_y_initial_dim
1690     {
1691         ( \g_@@_y_final_dim - \g_@@_y_initial_dim ) *
1692         \dim_ratio:nn
1693         { \l_@@_l_dim - \l_@@_inter_dots_dim * \l_tmpa_int }
1694         { \l_@@_l_dim * 2 } *
1695         \l_tmpb_int
1696     }
1697     \begin { tikzpicture } [ overlay ]
1698         \int_step_inline:nnn 0 \l_tmpa_int
1699         {

```

```

1700         \pgfpathcircle
1701         { \pgfpoint { \g_@@_x_initial_dim } { \g_@@_y_initial_dim } }
1702         { \l_@@_radius_dim }
1703         \pgfusepath { fill }
1704         \dim_gadd:Nn \g_@@_x_initial_dim \l_tmpa_dim
1705         \dim_gadd:Nn \g_@@_y_initial_dim \l_tmpb_dim
1706     }
1707     \end { tikzpicture }
1708 }
1709 }

```

14.10 User commands available in the new environments

We give new names for the commands `\ldots`, `\cdots`, `\vdots` and `\ddots` because these commands will be redefined (if the option `renew-dots` is used).

```

1710 \cs_set_eq:NN \@@_ldots \ldots
1711 \cs_set_eq:NN \@@_cdots \cdots
1712 \cs_set_eq:NN \@@_vdots \vdots
1713 \cs_set_eq:NN \@@_ddots \ddots
1714 \cs_set_eq:NN \@@_iddots \iddots

```

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

```

1715 \cs_new_protected:Nn \@@_add_to_empty_cells:
1716 {
1717     \cs_gset:cpx
1718     { @@ _ empty _ \int_use:N \c@iRow - \int_use:N \c@jCol }
1719     { \int_use:N \g_@@_env_int }
1720 }

```

The commands `\@@_Ldots`, `\@@_Cdots`, `\@@_Vdots`, `\@@_Ddots` and `\@@_Iddots` will be linked to `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots` and `\Iddots` in the environments `{NiceArray}` (the other environments of `nicematrix` rely upon `{NiceArray}`).

The starred versions of these commands are deprecated since version 3.1 but they are still available.

```

1721 \NewDocumentCommand \@@_Ldots { s }
1722 {
1723     \bool_if:nF { #1 } { \@@_instruction_of_type:n { Ldots } }
1724     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_ldots }
1725     \@@_add_to_empty_cells:
1726 }

```

```

1727 \NewDocumentCommand \@@_Cdots { s }
1728 {
1729     \bool_if:nF { #1 } { \@@_instruction_of_type:n { Cdots } }
1730     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_cdots }
1731     \@@_add_to_empty_cells:
1732 }

```

```

1733 \NewDocumentCommand \@@_Vdots { s }
1734 {
1735     \bool_if:nF { #1 } { \@@_instruction_of_type:n { Vdots } }
1736     \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_vdots }
1737     \@@_add_to_empty_cells:
1738 }

```

```

1739 \NewDocumentCommand \@@_Ddots { s }
1740 {
1741   \bool_if:nF { #1 } { \@@_instruction_of_type:n { Ddots } }
1742   \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_ddots }
1743   \@@_add_to_empty_cells:
1744 }

1745 \NewDocumentCommand \@@_Iddots { s }
1746 {
1747   \bool_if:nF { #1 } { \@@_instruction_of_type:n { Iddots } }
1748   \bool_if:NF \l_@@_nullify_dots_bool { \phantom \@@_iddots }
1749   \@@_add_to_empty_cells:
1750 }

```

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

```

1751 \cs_new_protected:Nn \@@_Hspace:
1752 {
1753   \@@_add_to_empty_cells:
1754   \hspace
1755 }

```

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

```

1756 \cs_set_eq:NN \@@_old_multicolumn \multicolumn
1757 \cs_new:Npn \@@_multicolumn:nnn #1 #2 #3
1758 {
1759   \@@_old_multicolumn { #1 } { #2 } { #3 }
1760   \int_compare:nNnT #1 > 1
1761   {
1762     \seq_gput_left:Nx \g_@@_multicolumn_cells_seq
1763     { \int_eval:n \c@iRow - \int_use:N \c@jCol }
1764     \seq_gput_left:Nn \g_@@_multicolumn_sizes_seq { #1 }
1765   }
1766   \int_gadd:Nn \c@jCol { #1 - 1 }
1767 }

```

The command `\@@_Hdotsfor` will be linked to `\Hdotsfor` in `{NiceArray}`. This command uses an optional argument like `\hdotsfor` but this argument is discarded (in `\hdotsfor`, this argument is used for fine tuning of the space between two consecutive dots). Tikz nodes are created for all the cells of the array, even the implicit cells of the `\Hdotsfor`.

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

```

1768 \cs_new:Npn \@@_Hdotsfor:
1769 {
1770   \multicolumn { 1 } { C } { }
1771   \@@_Hdotsfor_i
1772 }

```

The command `\@@_Hdotsfor_i` is defined with the tools of `xparse` because it has an optionnal argument. Note that such a command defined by `\NewDocumentCommand` is protected and that's why we have put the `\multicolumn` before (in the definition of `\@@_Hdotsfor:`).

```

1773 \bool_if:NTF \c_@@_draft_bool
1774 {
1775   \NewDocumentCommand \@@_Hdotsfor_i { 0 { } m }
1776   { \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { C } { } } }
1777 }
1778 {
1779   \NewDocumentCommand \@@_Hdotsfor_i { 0 { } m }
1780   {
1781     \tl_gput_right:Nx \g_@@_Hdotsfor_lines_tl
1782     {

```

```

1783         \@@_draw_Hdotsfor:nnn
1784         { \int_use:N \c@iRow }
1785         { \int_use:N \c@jCol }
1786         { #2 }
1787     }
1788     \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { C } { } }
1789 }
1790 }

1791 \cs_new_protected:Nn \@@_draw_Hdotsfor:nnn
1792 {
1793     \bool_set_false:N \l_@@_initial_open_bool
1794     \bool_set_false:N \l_@@_final_open_bool

```

For the row, it's easy.

```

1795     \int_set:Nn \l_@@_initial_i_int { #1 }
1796     \int_set:Nn \l_@@_final_i_int { #1 }

```

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

```

1797     \int_compare:nNnTF #2 = 1
1798     {
1799         \int_set:Nn \l_@@_initial_j_int 1
1800         \bool_set_true:N \l_@@_initial_open_bool
1801     }
1802     {
1803         \int_set:Nn \l_tmpa_int { #2 - 1 }
1804         \@@_if_not_empty_cell:nnTF \l_@@_initial_i_int \l_tmpa_int
1805         { \int_set:Nn \l_@@_initial_j_int { #2 - 1 } }
1806         {
1807             \int_set:Nn \l_@@_initial_j_int {#2}
1808             \bool_set_true:N \l_@@_initial_open_bool
1809         }
1810     }
1811     \int_compare:nNnTF { #2 + #3 - 1 } = \c@jCol
1812     {
1813         \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
1814         \bool_set_true:N \l_@@_final_open_bool
1815     }
1816     {
1817         \int_set:Nn \l_tmpa_int { #2 + #3 }
1818         \@@_if_not_empty_cell:nnTF \l_@@_final_i_int \l_tmpa_int
1819         { \int_set:Nn \l_@@_final_j_int { #2 + #3 } }
1820         {
1821             \int_set:Nn \l_@@_final_j_int { #2 + #3 - 1 }
1822             \bool_set_true:N \l_@@_final_open_bool
1823         }
1824     }
1825     \bool_if:nT { \l_@@_initial_open_bool || \l_@@_final_open_bool }
1826     \@@_create_medium_nodes:
1827     \@@_actually_draw_Ldots:

```

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

```

1828     \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
1829     { \cs_set:cpn { @@ _ dotted _ #1 - ##1 } { } }
1830 }

```

14.11 The command `\line` accessible in code-after

In the `code-after`, the command `\@@_line:nn` will be linked to `\line`. This command takes two arguments which are the specification of two cells in the array (in the format *i-j*) and draws a dotted line between these cells.

First, we write a command with an argument of the format $i-j$ and applies the command `\int_eval:n` to i and j ; this must *not* be protected (and is, of course fully expandable).³³

```
1831 \cs_new:Npn \@@_double_int_eval:n #1-#2 \q_stop
1832 { \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).

```
1833 \cs_new_protected:Npn \@@_line:nn #1 #2
1834 {
1835   \use:x
1836   {
1837     \@@_line_i:nn
1838     { \@@_double_int_eval:n #1 \q_stop }
1839     { \@@_double_int_eval:n #2 \q_stop }
1840   }
1841 }
1842 \cs_new_protected:Nn \@@_line_i:nn
1843 {
1844   \bool_if:NF \c_@@_draft_bool
1845   {
1846     \dim_zero_new:N \g_@@_x_initial_dim
1847     \dim_zero_new:N \g_@@_y_initial_dim
1848     \dim_zero_new:N \g_@@_x_final_dim
1849     \dim_zero_new:N \g_@@_y_final_dim
1850     \bool_set_false:N \l_@@_initial_open_bool
1851     \bool_set_false:N \l_@@_final_open_bool
1852     \bool_if:nTF
1853     {
1854       \cs_if_exist_p:c { pgf@sh@ns@nm - \int_use:N \g_@@_env_int - #1 }
1855       &&
1856       \cs_if_exist_p:c { pgf@sh@ns@nm - \int_use:N \g_@@_env_int - #2 }
1857     }
1858     {
1859       \begin { tikzpicture }
1860         \path~(#1)~---~(#2)~node[at~start]~(i)~{}~node[at~end]~(f)~{} ;
1861         \tikz@parse@node \pgfutil@firstofone ( i )
1862         \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1863         \dim_gset:Nn \g_@@_y_initial_dim \pgf@y
1864         \tikz@parse@node \pgfutil@firstofone ( f )
1865         \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1866         \dim_gset:Nn \g_@@_y_final_dim \pgf@y
1867       \end { tikzpicture }
1868       \@@_draw_tikz_line:
1869     }
1870     {
1871       \@@_error:nnn { unknown~cell~for~line~in~code~after }
1872       { #1 } { #2 }
1873     }
1874   }
1875 }
```

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).

14.12 The commands to draw dotted lines to separate columns and rows

The command `\hdottedline` draws an horizontal dotted line to separate two rows. Similarly, the letter “:” in the preamble draws a vertical dotted line (the letter can be changed with the option

³³Indeed, we want that the user may use the command `\line` in *code-after* with LaTeX counters in the arguments — with the command `\value`.

letter-for-dotted-lines). Both mechanisms write instructions in the `code-after`. The actual instructions in the `code-after` use the commands `\@@_hdottedline:n` and `\@@_vdottedline:n`.

We want the horizontal lines at the same position³⁴ as the line created by `\hline` (or `\hdashline` of `arydshln`). That's why we use a `\noalign` to insert a box with a `\dotfill`.

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

```

1876 \bool_if:NTF \c_@@_draft_bool
1877 { \cs_set_eq:NN \@@_dotfill: \prg_do_nothing: }
1878 {
1879   \cs_set:Npn \@@_dotfill:
1880   {

```

If the option `small` is used, we change the space between dots (we can't use `\l_@@_inter_dots_dim` which will be set after the construction of the array). We can't put the `\bool_if:NT` in the first argument of `\hbox_to_wd:nn` because `\cleaders` is a special TeX primitive.

```

1881   \bool_if:NT \l_@@_small_bool
1882   { \dim_set:Nn \l_@@_inter_dots_dim { 0.25 em } }
1883   \cleaders
1884   \hbox_to_wd:nn
1885   { \l_@@_inter_dots_dim }
1886   {
1887     \c_math_toggle_token
1888     \bool_if:NT \l_@@_small_bool \scriptstyle
1889     \hss . \hss
1890     \c_math_toggle_token
1891   }
1892   \hfill
1893 }
1894 }

```

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

```

1895 \cs_new:Npn \@@_hdottedline:
1896 {
1897   \noalign
1898   {
1899     \bool_gset_true:N \g_@@_large_nodes_bool
1900     \cs_if_exist:cTF { @@_width_ \int_use:N \g_@@_env_int }
1901     { \dim_set_eq:Nc \l_tmpa_dim { @@_width_ \int_use:N \g_@@_env_int } }
1902     { \dim_set:Nn \l_tmpa_dim { 5 mm } }
1903     \hbox_overlap_right:n
1904     {
1905       \bool_if:nT
1906       {
1907         \l_@@_NiceArray_bool
1908         &&
1909         ! \l_@@_exterior_arraycolsep_bool
1910         &&
1911         \int_compare_p:nNn \l_@@_first_col_int > 0
1912       }
1913       { \skip_horizontal:n { - \arraycolsep } }
1914       \hbox_to_wd:nn
1915       {
1916         \l_tmpa_dim + 2 \arraycolsep
1917         - \l_@@_left_margin_dim - \l_@@_right_margin_dim
1918       }
1919       \@@_dotfill:
1920     }
1921   }
1922 }

```

³⁴In fact, almost the same position because of the width of the line: the width of a dotted line is not the same as the width of a line created by `\hline`.

```

1923 \cs_new_protected:Nn \@@_vdottedline:n
1924 {

```

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

```

1925 \int_compare:nNnTF #1 = 0
1926 { \@@_error:n { Use~of~.:~in~first~position } }
1927 {
1928   \bool_if:NF \c_@@_draft_bool
1929   {
1930     \dim_zero_new:N \g_@@_x_initial_dim
1931     \dim_zero_new:N \g_@@_y_initial_dim
1932     \dim_zero_new:N \g_@@_x_final_dim
1933     \dim_zero_new:N \g_@@_y_final_dim
1934     \bool_set_true:N \l_@@_initial_open_bool
1935     \bool_set_true:N \l_@@_final_open_bool

```

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

```

1936 \cs_if_exist:cTF
1937 { pgf@sh@ns@nm -\int_use:N \g_@@_env_int - col - #1 }
1938 {
1939   \begin { tikzpicture } [ remember~picture ]
1940     \tikz@parse@node\pgfutil@firstofone
1941     ( col - #1 )
1942     \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1943     \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1944     \dim_gset:Nn \g_@@_y_final_dim \pgf@y
1945   \end { tikzpicture }
1946   \dim_gset:Nn \g_@@_y_initial_dim { - \c_max_dim }
1947   \int_step_inline:nn \c@jCol
1948   {
1949     \begin { tikzpicture } [ remember~picture ]
1950       \tikz@parse@node\pgfutil@firstofone
1951       ( 1 - #1 . north~east )
1952       \dim_gset:Nn \g_@@_y_initial_dim
1953       { \dim_max:nn \g_@@_y_initial_dim \pgf@y }
1954     \end { tikzpicture }
1955   }
1956 }

```

If not, we use the “large node”.

```

1957 {
1958   \begin { tikzpicture } [ remember~picture ]
1959     \tikz@parse@node\pgfutil@firstofone
1960     ( 1 - #1 - large .north~east )
1961     \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1962     \dim_gset:Nn \g_@@_y_initial_dim \pgf@y
1963     \tikz@parse@node\pgfutil@firstofone
1964     ( \int_use:N \c@iRow - #1 - large .south~east )
1965     \dim_gset:Nn \g_@@_x_final_dim \pgf@x
1966     \dim_gset:Nn \g_@@_y_final_dim \pgf@y
1967   \end { tikzpicture }

```

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

```

1968 \cs_if_exist:cT
1969 { pgf@sh@ns@nm -\int_use:N \g_@@_env_int - 1 - #1 - w }
1970 {
1971   \begin { tikzpicture } [ remember~picture ]
1972     \tikz@parse@node\pgfutil@firstofone
1973     ( 1 - #1 - w .north~east )
1974     \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
1975     \tikz@parse@node\pgfutil@firstofone
1976     ( \int_use:N \c@iRow - #1 - w .south~east )
1977     \dim_gset:Nn \g_@@_x_final_dim \pgf@x

```

```

1978         \end { tikzpicture }
1979         \dim_gadd:Nn \g_@@_x_initial_dim \arraycolsep
1980         \dim_gadd:Nn \g_@@_x_final_dim \arraycolsep
1981     }
1982 }
1983 \@@_draw_tikz_line:
1984 }
1985 }
1986 }

```

14.13 The vertical rules

We don't want that a vertical rule drawn by the specifier “|” extends in the eventual “first row” and “last row” of the array.

The natural way to do that would be to redefine the specifier “|” with `\newcolumnntype`:

```

\newcolumnntype { | }
{ ! { \int_compare:nNnF \c@iRow = 0 \vline } }

```

However, this code fails if the user uses `\DefineShortVerb{||}` of `fancyvrb`. Moreover, it would not be able to deal correctly with two consecutive specifiers “|” (in a preamble like `ccc||ccc`).

That's why we will do a redefinition of the macro `\@arrayrule` of `array` and this redefinition will add `\@@_vline:` instead of `\vline` to the preamble.

Here is the definition of `\@@_vline:`. This definition *must* be protected because you don't want that macro expanded during the construction of the preamble (the tests must be effective in each row and not once when the preamble is constructed).

```

1987 \cs_new_protected:Npn \@@_vline:
1988 {
1989     \int_compare:nNnTF \l_@@_first_col_int = 0
1990     {
1991         \int_compare:nNnTF \c@jCol = 0
1992         {
1993             \int_compare:nNnTF \l_@@_first_row_int = 0
1994             {
1995                 \int_compare:nNnF \c@iRow = 0
1996                 {
1997                     \int_compare:nNnF \c@iRow = \l_@@_last_row_int
1998                     \@@_vline_i:
1999                 }
2000             }
2001             {
2002                 \int_compare:nNnF \c@iRow = 0
2003                 {
2004                     \int_compare:nNnF \c@iRow = \l_@@_last_row_int
2005                     \@@_vline_i:
2006                 }
2007             }
2008         }
2009         {
2010             \int_compare:nNnF \c@iRow = 0
2011             {
2012                 \int_compare:nNnF \c@iRow = \l_@@_last_row_int
2013                 \@@_vline_i:
2014             }
2015         }
2016     }
2017     {
2018         \int_compare:nNnTF \c@jCol = 0
2019         {
2020             \int_compare:nNnF \c@iRow = { -1 }
2021             {

```



```

2022         \int_compare:nNnF \c@iRow = { \l_@@_last_row_int - 1 }
2023         \@@_vline_i:
2024     }
2025 }
2026 {
2027     \int_compare:nNnF \c@iRow = 0
2028     {
2029         \int_compare:nNnF \c@iRow = \l_@@_last_row_int
2030         \@@_vline_i:
2031     }
2032 }
2033 }
2034 }

```

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

```

2035 \cs_set_eq:NN \@@_vline_i: \vline

```

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

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

```

2036 \cs_set_eq:NN \OnlyMainNiceMatrix \use:n

```

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

```

2037 \cs_new:Npn \@@_OnlyMainNiceMatrix:n #1
2038 {
2039     \int_compare:nNnF \c@iRow = 0
2040     { \int_compare:nNnF \c@iRow = \l_@@_last_row_int { #1 } }
2041 }

```

14.14 The environment `{NiceMatrixBlock}`

The following flag will be raised when all the columns of the environments of the block must have the same width in “auto” mode.

```

2042 \bool_new:N \l_@@_block_auto_columns_width_bool

```

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

```

2043 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
2044 {
2045     auto-columns-width .code:n =
2046     {
2047         \bool_set_true:N \l_@@_block_auto_columns_width_bool
2048         \dim_gzero_new:N \g_@@_max_cell_width_dim
2049         \bool_set_true:N \l_@@_auto_columns_width_bool
2050     }
2051 }

```

```

2052 \NewDocumentEnvironment { NiceMatrixBlock } { ! 0 { } }
2053 {
2054     \int_gincr:N \g_@@_NiceMatrixBlock_int
2055     \dim_zero:N \l_@@_columns_width_dim
2056     \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
2057     \bool_if:NT \l_@@_block_auto_columns_width_bool

```

```

2058 {
2059   \cs_if_exist:cT { @@_max_cell_width_ \int_use:N \g_@@_NiceMatrixBlock_int }
2060   {
2061     \dim_set:Nx \l_@@_columns_width_dim
2062     { \use:c { @@_max_cell_width _ \int_use:N \g_@@_NiceMatrixBlock_int } }
2063   }
2064 }
2065 }

```

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).

```

2066 {
2067   \bool_if:NT \l_@@_block_auto_columns_width_bool
2068   {
2069     \iow_now:Nn \@mainaux \ExplSyntaxOn
2070     \iow_now:Nx \@mainaux
2071     {
2072       \cs_gset:cpn
2073       { @@ _ max _ cell _ width _ \int_use:N \g_@@_NiceMatrixBlock_int }
2074       { \dim_use:N \g_@@_max_cell_width_dim }
2075     }
2076     \iow_now:Nn \@mainaux \ExplSyntaxOff
2077   }
2078 }

```

14.15 The extra nodes

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

```

2079 \cs_generate_variant:Nn \dim_min:nn { v n }
2080 \cs_generate_variant:Nn \dim_max:nn { v n }

```

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

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

The command \@@_computations_for_medium_nodes: must be used in a {tikzpicture}. For each row i , we compute two dimensions $l_@@_row_i_min_dim$ and $l_@@_row_i_max_dim$. The dimension $l_@@_row_i_min_dim$ is the minimal y -value of all the cells of the row i . The dimension $l_@@_row_i_max_dim$ is the maximal y -value of all the cells of the row i .

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

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

```

2081 \cs_new_protected:Npn \@@_computations_for_medium_nodes:
2082 {
2083   \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
2084   {
2085     \dim_zero_new:c { l_@@_row\_@@_i: _min_dim }
2086     \dim_set_eq:cN { l_@@_row\_@@_i: _min_dim } \c_max_dim
2087     \dim_zero_new:c { l_@@_row\_@@_i: _max_dim }
2088     \dim_set:cn { l_@@_row\_@@_i: _max_dim } { - \c_max_dim }
2089   }

```

```

2090 \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:
2091 {
2092   \dim_zero_new:c { l_@@_column_\@@_j: _min_dim }
2093   \dim_set_eq:cN { l_@@_column_\@@_j: _min_dim } \c_max_dim
2094   \dim_zero_new:c { l_@@_column_\@@_j: _max_dim }
2095   \dim_set:cn { l_@@_column_\@@_j: _max_dim } { - \c_max_dim }
2096 }

```

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

```

2097 \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
2098 {
2099   \int_step_variable:nnNn
2100   \l_@@_first_col_int \g_@@_col_total_int \@@_j:

```

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

```

2101   { \cs_if_exist:cT
2102     { pgf@sh@ns@nm - \int_use:N \g_@@_env_int - \@@_i: - \@@_j: }

```

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

```

2103   {
2104     \tikz@parse@node \pgfutil@firstofone
2105     ( nm - \int_use:N \g_@@_env_int
2106       - \@@_i: - \@@_j: .south-west )
2107     \dim_set:cn { l_@@_row_\@@_i: _min_dim }
2108     { \dim_min:vn { l_@@_row _ \@@_i: _min_dim } \pgf@y }
2109     \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
2110     {
2111       \dim_set:cn { l_@@_column _ \@@_j: _min_dim }
2112       { \dim_min:vn { l_@@_column _ \@@_j: _min_dim } \pgf@x }
2113     }

```

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`.

```

2114     \tikz@parse@node \pgfutil@firstofone
2115     ( nm - \int_use:N \g_@@_env_int - \@@_i: - \@@_j: .north-east )
2116     \dim_set:cn { l_@@_row _ \@@_i: _ max_dim }
2117     { \dim_max:vn { l_@@_row _ \@@_i: _ max_dim } \pgf@y }
2118     \seq_if_in:NxF \g_@@_multicolumn_cells_seq { \@@_i: - \@@_j: }
2119     {
2120       \dim_set:cn { l_@@_column _ \@@_j: _ max_dim }
2121       { \dim_max:vn { l_@@_column _ \@@_j: _ max_dim } \pgf@x }
2122     }
2123   }
2124 }
2125 }
2126 }

```

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

```

2127 \cs_new_protected:Npn \@@_create_medium_nodes:
2128 {
2129   \begin { tikzpicture } [ remember-picture , overlay ]
2130     \@@_computations_for_medium_nodes:
2131     \tikzset { name~suffix = -medium }

```

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

```

2132   \@@_create_nodes:
2133   \end { tikzpicture }
2134   \cs_set_protected:Npn \@@_create_medium_nodes: { }
2135   \cs_set_protected:Npn \@@_create_medium_and_large_nodes:

```

```

2136     { \@@_create_large_nodes: }
2137 }

```

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

```

2138 \cs_new_protected:Npn \@@_create_large_nodes:
2139 {
2140   \begin { tikzpicture } [ remember-picture , overlay ]
2141     \@@_computations_for_medium_nodes:
2142     \@@_computations_for_large_nodes:
2143     \tikzset { name~suffix = -large }
2144     \@@_create_nodes:
2145   \end { tikzpicture }
2146   \@@_compute_width_of_array:
2147   \cs_set_protected:Npn \@@_create_large_nodes: { }
2148   \cs_set_protected:Npn \@@_create_medium_and_large_nodes:
2149     { \@@_create_medium_nodes: }
2150 }
2151 \cs_new_protected:Npn \@@_create_medium_and_large_nodes:
2152 {
2153   \begin { tikzpicture } [ remember-picture , overlay ]
2154     \@@_computations_for_medium_nodes:

```

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

```

2155     \tikzset { name~suffix = -medium }
2156     \@@_create_nodes:
2157     \@@_computations_for_large_nodes:
2158     \tikzset { name~suffix = -large }
2159     \@@_create_nodes:
2160     \@@_compute_width_of_array:
2161   \end { tikzpicture }
2162   \@@_compute_width_of_array:
2163   \cs_set_protected:Npn \@@_create_medium_and_large_nodes: { }
2164   \cs_set_protected:Npn \@@_create_medium_nodes: { }
2165   \cs_set_protected:Npn \@@_create_large_nodes: { }
2166 }

```

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.

```

2167 \cs_new_protected:Npn \@@_computations_for_large_nodes:
2168 {
2169   \int_set:Nn \l_@@_first_row_int 1
2170   \int_set:Nn \l_@@_first_col_int 1

```

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

```

2171   \int_step_variable:nNn { \c@iRow - 1 } \@@_i:
2172   {
2173     \dim_set:cn { l_@@_row _ \@@_i: _ min _ dim }
2174     {
2175       (
2176         \dim_use:c { l_@@_row _ \@@_i: _ min _ dim } +
2177         \dim_use:c { l_@@_row _ \int_eval:n { \@@_i: + 1 } _ max _ dim }
2178       )
2179       / 2
2180     }
2181     \dim_set_eq:cc { l_@@_row _ \int_eval:n { \@@_i: + 1 } _ max _ dim }

```

```

2182         { l_@@_row_ \@@_i: _min_dim }
2183     }
2184     \int_step_variable:nNn { \c@jCol - 1 } \@@_j:
2185     {
2186         \dim_set:cn { l_@@_column _ \@@_j: _ max _ dim }
2187         {
2188             (
2189                 \dim_use:c
2190                 { l_@@_column _ \@@_j: _ max _ dim } +
2191                 \dim_use:c
2192                 { l_@@_column _ \int_eval:n { \@@_j: + 1 } _ min _ dim }
2193             )
2194             / 2
2195         }
2196         \dim_set_eq:cc { l_@@_column _ \int_eval:n { \@@_j: + 1 } _ min _ dim }
2197         { l_@@_column _ \@@_j: _ max _ dim }
2198     }
2199     \dim_sub:cn
2200     { l_@@_column _ 1 _ min _ dim }
2201     \l_@@_left_margin_dim
2202     \dim_add:cn
2203     { l_@@_column _ \int_use:N \c@jCol _ max _ dim }
2204     \l_@@_right_margin_dim
2205 }

```

The control sequence `\@@_create_nodes:` is used twice: for the construction of the “medium nodes” and for the construction of the “large nodes”. The nodes are constructed with the value of all the dimensions `l_@@_row_i_min_dim`, `l_@@_row_i_max_dim`, `l_@@_column_j_min_dim` and `l_@@_column_j_max_dim`. Between the construction of the “medium nodes” and the “large nodes”, the values of these dimensions are changed.

```

2206 \cs_new_protected:Npn \@@_create_nodes:
2207 {
2208     \int_step_variable:nnNn \l_@@_first_row_int \g_@@_row_total_int \@@_i:
2209     {
2210         \int_step_variable:nnNn \l_@@_first_col_int \g_@@_col_total_int \@@_j:

```

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

```

2211     {
2212         \coordinate ( @@~south~west )
2213         at ( \dim_use:c { l_@@_column _ \@@_j: _min_dim } ,
2214             \dim_use:c { l_@@_row _ \@@_i: _min_dim } ) ;
2215         \coordinate ( @@~north~east )
2216         at ( \dim_use:c { l_@@_column _ \@@_j: _max_dim } ,
2217             \dim_use:c { l_@@_row _ \@@_i: _max_dim } ) ;

```

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

```

2218     \node
2219     [
2220         node~contents = { } ,
2221         fit = ( @@~south~west ) ( @@~north~east ) ,
2222         inner~sep = \c_zero_dim ,
2223         name = nm - \int_use:N \g_@@_env_int - \@@_i: - \@@_j: ,
2224         alias =
2225             \str_if_empty:NF \l_@@_name_str
2226             { \l_@@_name_str - \@@_i: - \@@_j: }
2227     ]
2228     ;
2229 }

```

```
2230 }
```

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 .

```
2231 \seq_mapthread_function:NNN
2232   \g_@@_multicolumn_cells_seq
2233   \g_@@_multicolumn_sizes_seq
2234   \@@_node_for_multicolumn:nn
2235 }
```

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

```
2236 \cs_new_protected:Npn \@@_compute_width_of_array:
2237 {
2238   \begin { tikzpicture } [ remember~picture , overlay ]
2239   \tikz@parse@node \pgfutil@firstofone
2240     ( nm - \int_use:N \g_@@_env_int - 1 - 1 - large .north~west )
2241   \dim_gset:Nn \g_tmpa_dim \pgf@x
2242   \tikz@parse@node \pgfutil@firstofone
2243     ( nm - \int_use:N \g_@@_env_int - 1 -
2244       \int_use:N \c@jCol - large .north~east )
2245   \dim_gset:Nn \g_tmpb_dim \pgf@x
2246   \end { tikzpicture }
2247   \iow_now:Nn \@mainaux \ExplSyntaxOn
2248   \iow_now:Nx \@mainaux
2249   {
2250     \cs_gset:cpn { @@_width_ \int_use:N \g_@@_env_int }
2251       { \dim_eval:n { \g_tmpb_dim - \g_tmpa_dim } }
2252   }
2253   \iow_now:Nn \@mainaux \ExplSyntaxOff
2254 }
```

```
2255 \cs_new_protected:Npn \@@_extract_coords: #1 - #2 \q_stop
2256 {
2257   \cs_set:Npn \@@_i: { #1 }
2258   \cs_set:Npn \@@_j: { #2 }
2259 }
```

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”).

```
2260 \cs_new_protected:Nn \@@_node_for_multicolumn:nn
2261 {
2262   \@@_extract_coords: #1 \q_stop
2263   \coordinate ( @@~south~west ) at
2264     (
2265       \dim_use:c { l_@@_column _ \@@_j: _ min _ dim } ,
2266       \dim_use:c { l_@@_row _ \@@_i: _ min _ dim }
2267     ) ;
2268   \coordinate ( @@~north~east ) at
2269     (
2270       \dim_use:c { l_@@_column _ \int_eval:n { \@@_j: + #2 - 1 } _ max _ dim } ,
2271       \dim_use:c { l_@@_row _ \@@_i: _ max _ dim }
2272     ) ;
2273   \node
2274     [
2275     node~contents = { } ,
2276     fit = ( @@~south~west ) ( @@~north~east ) ,
2277     inner~sep = \c_zero_dim ,
2278     name = nm - \int_use:N \g_@@_env_int - \@@_i: - \@@_j: ,
2279     alias =
2280       \str_if_empty:NF \l_@@_name_str
```

```

2281         { \l_@@_name_str - \@@_i: - \@@_j: }
2282     ]
2283     ;
2284 }

```

14.16 Block matrices

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

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

```

2285 \NewDocumentCommand \@@_Block: { m D < > { } m }
2286 { \@@_Block_i #1 \q_stop { #2 } { #3 } }

```

The first argument of `\@@_Block:` (which is required) has a special syntax. It must be of the form $i-j$ where i and j are the size (in rows and columns) of the block.

```

2287 \cs_new:Npn \@@_Block_i #1-#2 \q_stop { \@@_Block_ii:nnnn { #1 } { #2 } }

```

Now, the arguments have been extracted: `#1` is i (the number of rows of the block), `#2` is j (the number of columns of the block), `#3` are the tokens to put before the math mode and `#4` is the label of the block. The following command must *not* be protected because it contains a command `\multicolumn` (in the case of a block of only one row).

```

2288 \cs_new:Npn \@@_Block_ii:nnnn #1 #2 #3 #4
2289 {

```

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

```

2290     \int_compare:nNnT { #1 } = 1
2291     {
2292         \begin { tikzpicture } [ remember~picture , baseline ]
2293             \coordinate
2294                 ( nm - \int_use:N \g_@@_env_int
2295                   - Block
2296                   - \int_use:N \c@iRow
2297                   - \int_use:N \c@jCol ) ;
2298             \end { tikzpicture }
2299         }
2300     \bool_gset_true:N \g_@@_medium_nodes_bool

```

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

```

2301     \tl_gput_left:Nx \g_@@_code_after_tl
2302     {
2303         \@@_Block_iii:nnnnn
2304         { \int_use:N \c@iRow }
2305         { \int_use:N \c@jCol }
2306         { \int_eval:n { \c@iRow + #1 - 1 } }
2307         { \int_eval:n { \c@jCol + #2 - 1 } }
2308         \exp_not:n { { #3 $ #4 $ } }
2309     }
2310 }

```

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

```

2311 \cs_new_protected:Npn \@@_Block_iii:nnnnn #1 #2 #3 #4 #5
2312 {
2313     \bool_if:nTF
2314     {
2315         \int_compare_p:nNn { #3 } > \c@iRow

```

```

2316      || \int_compare_p:nNn { #4 } > \c@jCol
2317    }
2318    { \msg_error:nnnn { nicematrix } { Block-too-large } { #1 } { #2 } }
2319  {

```

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

```

2320      \int_compare:nNnTF { #1 } = { #3 }
2321      {
2322        \begin { tikzpicture }

```

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

```

2323      \tikz@parse@node \pgfutil@firstofone (Block-#1-#2)
2324      \dim_set:Nn \l_tmpa_dim \pgf@y
2325      \node
2326      [
2327        fit = ( #1 - #2 - medium . north-west )
2328              ( #3 - #4 - medium . south-east ) ,
2329        inner-sep = 0 pt ,
2330      ]
2331      (#1-#2) { } ;

```

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

```

2332      \tikz@parse@node \pgfutil@firstofone (#1-#2)
2333      \path (\pgf@x,\l_tmpa_dim) node [ anchor = base ] { #5 } ;
2334    \end { tikzpicture }
2335  }

```

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

```

2336      {
2337        \begin { tikzpicture }
2338        \node
2339        [
2340          fit = ( #1 - #2 - medium . north-west )
2341                ( #3 - #4 - medium . south-east ) ,
2342          inner-sep = 0 pt ,
2343        ]

```

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

```

2344          (#1-#2) { } ;
2345          \node at (#1-#2.center) { #5 } ;
2346        \end { tikzpicture }
2347      }
2348    }
2349  }

```

14.17 How to draw the dotted lines transparently

```

2350 \cs_set_protected:Npn \@@_renew_matrix:
2351 {
2352   \RenewDocumentEnvironment { pmatrix } { } {
2353     { \pNiceMatrix }
2354     { \endpNiceMatrix }
2355   \RenewDocumentEnvironment { vmatrix } { } {
2356     { \vNiceMatrix }
2357     { \endvNiceMatrix }
2358   \RenewDocumentEnvironment { Vmatrix } { } {
2359     { \VNiceMatrix }
2360     { \endVNiceMatrix }
2361   \RenewDocumentEnvironment { bmatrix } { } {
2362     { \bNiceMatrix }

```



```

2363     { \endbNiceMatrix }
2364 \RenewDocumentEnvironment { Bmatrix } { }
2365     { \BNiceMatrix }
2366     { \endBNiceMatrix }
2367 }

```

14.18 Automatic arrays

```

2368 \cs_new_protected:Npn \@@_set_size:n #1-#2 \q_stop
2369 {
2370     \int_set:Nn \l_@@_nb_rows_int { #1 }
2371     \int_set:Nn \l_@@_nb_cols_int { #2 }
2372 }
2373 \NewDocumentCommand \AutoNiceMatrixWithDelims { m m O { } m O { } m ! O { } }
2374 {
2375     \int_zero_new:N \l_@@_nb_rows_int
2376     \int_zero_new:N \l_@@_nb_cols_int
2377     \@@_set_size:n #4 \q_stop
2378     \begin { NiceArrayWithDelims } { #1 } { #2 }
2379         { * { \l_@@_nb_cols_int } { C } } [ #3 , #5 , #7 ]
2380     \int_compare:nNnT \l_@@_first_row_int = 0
2381     {
2382         \int_compare:nNnT \l_@@_first_col_int = 0 { & }
2383         \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
2384         \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
2385     }
2386     \prg_replicate:nn \l_@@_nb_rows_int
2387     {
2388         \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`).

```

2389     \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { { } #6 & } #6
2390     \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
2391 }
2392 \int_compare:nNnT \l_@@_last_row_int > { -2 }
2393 {
2394     \int_compare:nNnT \l_@@_first_col_int = 0 { & }
2395     \prg_replicate:nn { \l_@@_nb_cols_int - 1 } { & }
2396     \int_compare:nNnT \l_@@_last_col_int > { -1 } { & } \\
2397 }
2398 \end { NiceArrayWithDelims }
2399 }
2400 \cs_set_protected:Npn \@@_define_com:nnn #1 #2 #3
2401 {
2402     \cs_set_protected:cpn { #1 AutoNiceMatrix }
2403     {
2404         \str_gset:Nx \g_@@_type_env_str
2405         { command ~ \c_backslash_str #1 AutoNiceMatrix }
2406         \AutoNiceMatrixWithDelims { #2 } { #3 }
2407     }
2408 }
2409 \@@_define_com:nnn p ( )
2410 \@@_define_com:nnn b [ ]
2411 \@@_define_com:nnn v | |
2412 \@@_define_com:nnn V \l \l
2413 \@@_define_com:nnn B \{ \}

```

14.19 We process the options

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

We must process these options after the definition of the environment `{NiceMatrix}` because the option `renew-matrix` executes the code `\cs_set_eq:NN \env@matrix \NiceMatrix`. Of course, the command `\NiceMatrix` must be defined before such an instruction is executed.

```

2414 \keys_define:nn { NiceMatrix / Package }
2415 {
2416   renew-dots .bool_set:N = \l_@@_renew_dots_bool ,
2417   renew-dots .value_forbidden:n = true ,
2418   renew-matrix .code:n = \@@_renew_matrix: ,
2419   renew-matrix .value_forbidden:n = true ,
2420   transparent .meta:n = { renew-dots , renew-matrix } ,
2421   transparent .value_forbidden:n = true,
2422   obsolete-environments .code:n =
2423     \@@_msg_redirect_name:nn { Obsolete-environment } { none }
2424 }
2425 \ProcessKeysOptions { NiceMatrix / Package }

```

14.20 Error messages of the package

```

2426 \@@_msg_new:nn { unknown-cell-for-line-in-code-after }
2427 {
2428   Your-command-\token_to_str:N\line\{#1\}\{#2\}-in-the-'code-after'-
2429   can't-be-executed-because-a-Tikz-node-doesn't-exist.\\
2430   If-you-go-on-this-command-will-be-ignored.
2431 }
2432 \@@_msg_new:nn { last-col-non-empty-for-NiceArray }
2433 {
2434   In-the-\g_@@_type_env_str,~you-must-use-the-option~
2435   'last-col'~without-value.\\
2436   However,~you-can-go-on-for-this-time~
2437   (the-value-\l_keys_value_tl~will-be-ignored).
2438 }
2439 \@@_msg_new:nn { last-col-empty-for-NiceMatrix }
2440 {
2441   In-the-\g_@@_type_env_str, you-can't-use-the-option~
2442   'last-col'~without-value.~You-must-give-the-number-of-that-last-column.\\
2443   If-you-go-on-this-option-will-be-ignored.
2444 }
2445 \@@_msg_new:nn { Block-too-large }
2446 {
2447   You-try-to-draw-a-block-in-the-cell~#1~#2~of-your-matrix-but-the-matrix-is~
2448   too-small-for-that-block.\\
2449   If-you-go-on,~this-command-will-be-ignored.
2450 }
2451 \@@_msg_new:nn { Impossible-line }
2452 {
2453   A-dotted-line-can't-be-drawn-because-you-have-not-put~
2454   all-the-ampersands-required-on-the-row~#1.\\
2455   If-you-go-on,~this-dotted-line-will-be-ignored.
2456 }
2457 \@@_msg_new:nn { Wrong-last-row }
2458 {
2459   You-have-used-'last-row=\int_use:N \l_@@_last_row_int'~but-your~
2460   \g_@@_type_env_str\ seems-to-have-\int_use:N \c@iRow \ rows.~
2461   If-you-go-on,~the-value-of-\int_use:N \c@iRow \ will-be-used-for~
2462   last-row.~You-can-avoid-this-problem-by-using-'last-row'~
2463   without-value~(more-compilations-might-be-necessary).
2464 }
2465 \@@_msg_new:nn { Yet-in-env }
2466 {
2467   Environments-\{NiceArray\}~(or-\{NiceMatrix\},~etc.)~can't-be-nested.\\

```

```

2468     This-error-is-fatal.
2469 }

2470 \@@_msg_new:nn { Outside-math-mode }
2471 {
2472     The~\g_@@_type_env_str\ can-be-used-only-in-math-mode~
2473     (and-not-in-\token_to_str:N \vcenter).\
2474     This-error-is-fatal.
2475 }

2476 \@@_msg_new:nn { Option-Transparent-suppressed }
2477 {
2478     The-option-'Transparent'-has-been-renamed-'transparent'.\
2479     If-you-go-on-this-command-will-be-ignored.
2480 }

2481 \@@_msg_new:nn { Option-RenewMatrix-suppressed }
2482 {
2483     The-option-'RenewMatrix'-has-been-renamed-'renew-matrix'.\
2484     If-you-go-on-this-command-will-be-ignored.
2485 }

2486 \@@_msg_new:nn { Bad-value-for-letter-for-dotted-lines }
2487 {
2488     The-value-of-key~'\tl_use:N\l_keys_key_tl'~must-be-of-length~1.\
2489     If-you-go-on,~it-will-be-ignored.
2490 }

2491 \@@_msg_new:nnn { Unknown-key-for-NiceMatrixOptions }
2492 {
2493     The-key~'\tl_use:N\l_keys_key_tl'~is-unknown-for-the-command~
2494     \token_to_str:N \NiceMatrixOptions. \
2495     If-you-go-on,~it-will-be-ignored. \
2496     For-a-list-of-the-available-keys,~type-H~<return>.
2497 }
2498 {
2499     The-available-options-are~(in-alphabetic-order):~
2500     allow-duplicate-names,~
2501     code-for-first-col,~
2502     code-for-first-row,~
2503     code-for-last-col,~
2504     code-for-last-row,~
2505     create-extra-nodes,~
2506     create-medium-nodes,~
2507     create-large-nodes,~
2508     dotted-lines-margin,~
2509     end-of-row,~
2510     exterior-arraycolsep,~
2511     hlines,~
2512     left-margin,~
2513     letter-for-dotted-lines,~
2514     light-syntax,~
2515     nullify-dots,~
2516     parallelize-diags,~
2517     renew-dots,~
2518     renew-matrix,~
2519     right-margin,~
2520     small,~
2521     and-transparent
2522 }

2523 \@@_msg_new:nnn { Unknown-option-for-NiceArray }
2524 {
2525     The-option~'\tl_use:N\l_keys_key_tl'~is-unknown-for-the-environment~
2526     \{NiceArray\}. \
2527     If-you-go-on,~it-will-be-ignored. \
2528     For-a-list-of-the-available-options,~type-H~<return>.

```

```

2529 }
2530 {
2531   The~available~options~are~(in~alphabetic~order):~
2532   b,~
2533   c,~
2534   code-after,~
2535   code-for-first-col,~
2536   code-for-first-row,~
2537   code-for-last-col,~
2538   code-for-last-row,~
2539   columns-width,~
2540   create-extra-nodes,~
2541   create-medium-nodes,~
2542   create-large-nodes,~
2543   dotted-lines-margin,~
2544   end-of-row,~
2545   extra-left-margin,~
2546   extra-right-margin,~
2547   first-col,~
2548   first-row,~
2549   hlines,~
2550   last-col,~
2551   last-row,~
2552   left-margin,~
2553   light-syntax,~
2554   name,~
2555   nullify-dots,~
2556   parallelize-diags,~
2557   renew-dots,~
2558   right-margin,~
2559   small,~
2560   and~t.
2561 }

```

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).

```

2562 \@@_msg_new:nnn { Unknown~option~for~NiceMatrix }
2563 {
2564   The~option~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~
2565   \g_@@_type_env_str. \\\
2566   If~you~go~on,~it~will~be~ignored. \\\
2567   For~a~list~of~the~available~options,~type~H~<return>.
2568 }
2569 {
2570   The~available~options~are~(in~alphabetic~order):~
2571   code-after,~
2572   code-for-first-col,~
2573   code-for-first-row,~
2574   code-for-last-col,~
2575   code-for-last-row,~
2576   columns-width,~
2577   create-extra-nodes,~
2578   create-medium-nodes,~
2579   create-large-nodes,~
2580   dotted-lines-margin,~
2581   end-of-row,~
2582   extra-left-margin,~
2583   extra-right-margin,~
2584   first-col,~
2585   first-row,~
2586   hlines,~
2587   last-col,~
2588   last-row,~
2589   left-margin,~

```

```

2590   light-syntax,~
2591   name,~
2592   nullify-dots,~
2593   parallelize-diags,~
2594   renew-dots,~
2595   right-margin,~
2596   and~small.
2597 }
2598 \@@_msg_new:nnn { Duplicate~name }
2599 {
2600   The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
2601   the~same~environment~name~twice.~You~can~go~on,~but,~
2602   maybe,~you~will~have~incorrect~results~especially~
2603   if~you~use~'columns-width=auto'.~If~you~don't~want~to~see~this~
2604   message~again,~use~the~option~'allow-duplicate-names'.\\
2605   For~a~list~of~the~names~already~used,~type~H~<return>. \\
2606 }
2607 {
2608   The~names~already~defined~in~this~document~are:~
2609   \seq_use:Nnnn \g_@@_names_seq { ,~ } { ,~ } { ~and~ }.
2610 }
2611 \@@_msg_new:nn { Option~auto~for~columns~width }
2612 {
2613   You~can't~give~the~value~'auto'~to~the~option~'columns-width'~here.~
2614   If~you~go~on,~the~option~will~be~ignored.
2615 }
2616 \@@_msg_new:nn { Zero~row }
2617 {
2618   There~is~a~problem.~Maybe~your~\g_@@_type_env_str\ is~empty.~
2619   Maybe~you~have~used~l,~c~and~r~instead~of~L,~C~and~R~in~the~preamble~
2620   of~your~environment. \\
2621   If~you~go~on,~the~result~may~be~incorrect.
2622 }
2623 \@@_msg_new:nn { Use~of~::~in~first~position }
2624 {
2625   You~can't~use~the~column~specifier~'\l_@@_letter_for_dotted_lines_str'~in~the~
2626   first~position~of~the~preamble~of~the~\g_@@_type_env_str. \\
2627   If~you~go~on,~this~dotted~line~will~be~ignored.
2628 }

```

14.21 Obsolete environments

```

2629 \@@_msg_new:nn { Obsolete~environment }
2630 {
2631   The~environment~\{\@currenvir\}~is~obsolete.~We~should~use~#1~instead.~
2632   However,~you~can~go~on~for~this~time.~
2633   If~you~don't~want~to~see~this~error~again,~you~should~load~'nicematrix'~
2634   with~the~option~'obsolete-environments'.
2635 }
2636 \NewDocumentEnvironment { pNiceArrayC } { }
2637 {
2638   \@@_error:nn { Obsolete~environment }
2639     { the~option~'last-col' }
2640   \int_zero:N \l_@@_last_col_int
2641   \pNiceArray
2642 }
2643 { \endpNiceArray }
2644 \NewDocumentEnvironment { bNiceArrayC } { }
2645 {
2646   \@@_error:nn { Obsolete~environment }
2647     { the~option~'last-col' }
2648   \int_zero:N \l_@@_last_col_int

```

```

2649     \bNiceArray
2650   }
2651   { \endbNiceArray }

2652 \NewDocumentEnvironment { BNiceArrayC } { }
2653 {
2654   \@@_error:nn { Obsolete-environment }
2655     { the~option~'last-col' }
2656   \int_zero:N \l_@@_last_col_int
2657   \BNiceArray
2658 }
2659 { \endBNiceArray }

2660 \NewDocumentEnvironment { vNiceArrayC } { }
2661 {
2662   \@@_error:nn { Obsolete-environment }
2663     { the~option~'last-col' }
2664   \int_zero:N \l_@@_last_col_int
2665   \vNiceArray
2666 }
2667 { \endvNiceArray }

2668 \NewDocumentEnvironment { VNiceArrayC } { }
2669 {
2670   \@@_error:nn { Obsolete-environment }
2671     { the~option~'last-col' }
2672   \int_zero:N \l_@@_last_col_int
2673   \VNiceArray
2674 }
2675 { \endVNiceArray }

2676 \NewDocumentEnvironment { pNiceArrayRC } { }
2677 {
2678   \@@_error:nn { Obsolete-environment }
2679     { the~options~'last-col'~and~'first-row' }
2680   \int_zero:N \l_@@_last_col_int
2681   \int_zero:N \l_@@_first_row_int
2682   \pNiceArray
2683 }
2684 { \endpNiceArray }

2685 \NewDocumentEnvironment { bNiceArrayRC } { }
2686 {
2687   \@@_error:nn { Obsolete-environment }
2688     { the~options~'last-col'~and~'first-row' }
2689   \int_zero:N \l_@@_last_col_int
2690   \int_zero:N \l_@@_first_row_int
2691   \bNiceArray
2692 }
2693 { \endbNiceArray }

2694 \NewDocumentEnvironment { BNiceArrayRC } { }
2695 {
2696   \@@_error:nn { Obsolete-environment }
2697     { the~options~'last-col'~and~'first-row' }
2698   \int_zero:N \l_@@_last_col_int
2699   \int_zero:N \l_@@_first_row_int
2700   \BNiceArray
2701 }
2702 { \endBNiceArray }

2703 \NewDocumentEnvironment { vNiceArrayRC } { }
2704 {
2705   \@@_error:nn { Obsolete-environment }
2706     { the~options~'last-col'~and~'first-row' }
2707   \int_zero:N \l_@@_last_col_int
2708   \int_zero:N \l_@@_first_row_int
2709   \vNiceArray

```

```

2710 }
2711 { \endvNiceArray }
2712 \NewDocumentEnvironment { VNiceArrayRC } { }
2713 {
2714   \@@_error:nn { Obsolete-environment }
2715   { the~options~'last-col'~and~'first-row' }
2716   \int_zero:N \l_@@_last_col_int
2717   \int_zero:N \l_@@_first_row_int
2718   \VNiceArray
2719 }
2720 { \endVNiceArray }
2721 \NewDocumentEnvironment { NiceArrayCwithDelims } { }
2722 {
2723   \@@_error:nn { Obsolete-environment }
2724   { the~option~'last-col' }
2725   \int_zero:N \l_@@_last_col_int
2726   \NiceArrayWithDelims
2727 }
2728 { \endNiceArrayWithDelims }
2729 \NewDocumentEnvironment { NiceArrayRCwithDelims } { }
2730 {
2731   \@@_error:nn { Obsolete-environment }
2732   { the~options~'last-col'~and~'first-row' }
2733   \int_zero:N \l_@@_last_col_int
2734   \int_zero:N \l_@@_first_row_int
2735   \NiceArrayWithDelims
2736 }
2737 { \endNiceArrayWithDelims }

```

15 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 StackExchange³⁵, Tikz externalization is now deactivated in the environments of the extension `nicematrix`.³⁶

Changes between version 2.1 and 2.1.2

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

Changes between version 2.1.2 and 2.1.3

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

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

Changes between version 2.1.3 and 2.1.4

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

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

Changes between version 2.1.4 and 2.1.5

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

Option `allow-duplicate-names`.

Changes between version 2.1.5 and 2.2

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

³⁵cf. tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package

³⁶Before this version, there was an error when using `nicematrix` with Tikz externalization. In any case, it’s not possible to externalize the Tikz elements constructed by `nicematrix` because they use the options `overlay` and `remember picture`.

Changes between version 2.2 and 2.2.1

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

Changes between version 2.2.1 and 2.3

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

Changes between version 2.3 and 3.0

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

Changes between version 3.0 and 3.1

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

Changes between version 3.1 and 3.2 (and 3.2a)

Option `small`.

Changes between version 3.2 and 3.3

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

Changes between version 3.3 and 3.4

Following a discussion on TeX StackExchange³⁷, 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.

³⁷cf. 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`.

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>\@@_Hspace:</code> 505, 1756
<code>\@@_Block:</code>	508, 2290	<code>\@@_Iddots</code> 503, 516, 1750
<code>\@@_Block_i</code>	2291, 2292	<code>\g_@@_Iddots_lines_tl</code> 559, 1175
<code>\@@_Block_ii:nnnn</code>	2292, 2293	<code>\@@_Ldots</code> 499, 512, 517, 1726
<code>\@@_Block_iii:nnnnn</code>	2308, 2316	<code>\g_@@_Ldots_lines_tl</code> 556, 1177
<code>\@@_Cdots</code>	500, 513, 1732	<code>\l_@@_NiceArray_bool</code>
<code>\g_@@_Cdots_lines_tl</code>	555, 1176 54, 583, 609, 639, 648, 698, 987, 1912
<code>_@@_Cell:</code>	377	<code>\g_@@_NiceMatrixBlock_int</code>
<code>\@@_Cell:</code>	124, 292, 496, 497, 498 50, 2059, 2064, 2067, 2078
<code>\@@_Ddots</code>	502, 515, 1744	<code>\@@_OnlyMainNiceMatrix:n</code> 509, 2042
<code>\g_@@_Ddots_lines_tl</code>	558, 1174	<code>\@@_Vdots</code> 501, 514, 1738
<code>\@@_Hdotsfor:</code>	506, 518, 1773	<code>\g_@@_Vdots_lines_tl</code> 557, 1173
<code>\@@_Hdotsfor_i</code>	1776, 1780, 1784	<code>\@@_actualization_for_first_and_last_</code>
<code>\g_@@_Hdotsfor_lines_tl</code> ...	560, 1172, 1786	<code>row:</code> 319, 344, 871, 942
		<code>\@@_actually_draw_Ldots:</code> . 1388, 1391, 1832

<code>\@@adapt_S_column:</code>	99, 114, 569	<code>\@@draw_Hdotsfor:nnn</code>	1788, 1796
<code>\@@add_to_empty_cells:</code>		<code>\@@draw_Iddots:nn</code>	1570
. . .	1720, 1730, 1736, 1742, 1748, 1754, 1758	<code>\@@draw_Ldots:nn</code>	1382
<code>\@@adjust_with_col_nodes:</code>	1352, 1424, 1472	<code>\@@draw_Vdots:nn</code>	1481
<code>\@@after_array:</code>	751, 1108	<code>\@@draw_tikz_line:</code>	
<code>\@@after_array_i:</code>	1111, 1117	1431, 1477, 1524, 1566, 1608, 1614, 1873, 1988	
<code>\@@array:</code>	418, 754, 761	<code>_@@end_Cell:</code>	381
<code>\l_@@auto_columns_width_bool</code>		<code>\@@end_Cell:</code>	126, 338, 496, 497, 498
.	139, 194, 539, 790, 803, 2054	<code>\l_@@end_of_row_tl</code>	151, 152, 159, 763, 764
<code>\@@begin_of_row:</code>	298, 312, 850	<code>\g_@@env_int</code>	49,
<code>\l_@@block_auto_columns_width_bool</code>		355, 387, 579, 592, 595, 799, 816, 842,	
.	580, 804, 2047, 2052, 2062, 2072	892, 970, 1068, 1081, 1091, 1132, 1184,	
<code>\@@cdots</code>	1716, 1735	1254, 1315, 1338, 1344, 1357, 1361, 1370,	
<code>\@@cell_with_light_syntax:n</code>	778, 785	1376, 1400, 1415, 1448, 1463, 1613, 1724,	
<code>\g_@@cells_seq</code>	774, 775, 776, 778	1859, 1861, 1905, 1906, 1942, 1974, 2107,	
<code>\g_@@code_after_tl</code>		2110, 2120, 2228, 2245, 2248, 2255, 2283, 2299	
.	79, 206, 546, 1188, 1189, 2306	<code>\@@error:n</code>	17, 239, 242, 246, 255, 258,
<code>\l_@@code_for_first_col_tl</code>	161, 864	267, 269, 277, 279, 285, 290, 676, 1113, 1931	
<code>\l_@@code_for_first_row_tl</code>	165, 306	<code>\@@error:nn</code>	
<code>\l_@@code_for_last_col_tl</code>	163, 935	18, 201, 2643, 2651, 2659, 2667,
<code>\l_@@code_for_last_row_tl</code>	167, 309	2675, 2683, 2692, 2701, 2710, 2719, 2728, 2736	
<code>\g_@@col_total_int</code>	300, 301,	<code>\@@error:nnn</code>	19, 1876
524, 819, 820, 920, 921, 1122, 2095, 2105, 2215		<code>\@@everycr:</code>	431, 471, 474
<code>\c_@@colortbl_loaded_bool</code>	62, 67, 466	<code>\@@everycr_i:</code>	432, 433
<code>\l_@@columns_width_dim</code>		<code>\l_@@exterior_arraycolsep_bool</code>	
.	51, 195, 247, 540, 791, 812, 2060, 2066	134, 243, 641, 650, 1914
<code>\@@computations_for_large_nodes:</code>		<code>\l_@@extra_left_margin_dim</code>	
.	2147, 2162, 2172	149, 185, 656, 908
<code>\@@computations_for_medium_nodes:</code>		<code>\l_@@extra_right_margin_dim</code>	
.	2086, 2135, 2146, 2159	150, 186, 668, 956
<code>\@@compute_width_of_array:</code>		<code>\@@extract_coords:</code>	2260, 2267
.	2151, 2165, 2167, 2241	<code>\@@fatal:n</code>	20, 58, 571
<code>\@@create_col_node:</code>	826, 833, 837	<code>\@@fatal:nn</code>	21
<code>\@@create_large_nodes:</code>		<code>\l_@@final_i_int</code>	
.	1160, 2141, 2143, 2152, 2170	1163, 1204, 1209, 1212, 1233, 1238,
<code>\@@create_medium_and_large_nodes:</code>		1244, 1255, 1262, 1345, 1416, 1464, 1801, 1823	
.	1157, 2140, 2153, 2156, 2168	<code>\l_@@final_j_int</code>	
<code>\@@create_medium_nodes:</code>	1164, 1205, 1210, 1218, 1224, 1234, 1238,
.	1158, 1327, 1328, 1831, 2132, 2139, 2154, 2169	1245, 1256, 1346, 1417, 1465, 1818, 1824, 1826	
<code>\@@create_nodes:</code>	2137, 2149, 2161, 2164, 2211	<code>\l_@@final_open_bool</code>	
<code>\@@create_row_of_col_nodes:</code>	756, 768, 786	1166, 1211, 1215, 1221, 1227,
<code>\@@ddots</code>	1718, 1747	1231, 1259, 1328, 1366, 1410, 1427, 1458,	
<code>\l_@@ddots_int</code>	1147, 1549, 1550	1475, 1496, 1509, 1543, 1585, 1637, 1652,	
<code>\@@define_com:nnn</code>		1683, 1684, 1799, 1819, 1827, 1830, 1856, 1940	
.	2405, 2414, 2415, 2416, 2417, 2418	<code>\@@find_extremities_of_line:nnnn</code>	
<code>\@@define_env:n</code>	1199, 1387, 1438, 1486, 1533, 1575
.	1033, 1052, 1053, 1054, 1055, 1056, 1057	<code>\l_@@first_col_int</code>	84,
<code>\l_@@delta_x_one_dim</code>	1149, 1552, 1562	85, 208, 283, 297, 636, 693, 795, 1916,
<code>\l_@@delta_x_two_dim</code>	1151, 1594, 1604	1994, 2095, 2105, 2175, 2215, 2387, 2393, 2399	
<code>\l_@@delta_y_one_dim</code>	1150, 1554, 1562	<code>\l_@@first_row_int</code>	82, 83, 209,
<code>\l_@@delta_y_two_dim</code>	1152, 1596, 1604	287, 522, 681, 700, 1998, 2088, 2102, 2174,
<code>\@@dotfill:</code>	1882, 1884, 1924	2213, 2385, 2686, 2695, 2704, 2713, 2722, 2739	
<code>\l_@@dotted_lines_margin_dim</code>		<code>\@@hdottedline:</code>	504, 1900
.	74, 75, 155, 1646, 1657, 1665	<code>\l_@@hlines_bool</code>	137, 170, 436
<code>\@@double_int_eval:n</code>	1836, 1843, 1844	<code>\g_@@ht_last_row_dim</code>	
<code>\g_@@dp_ante_last_row_dim</code>	315, 482, 483, 716	317, 333, 334, 484, 485, 690
<code>\g_@@dp_last_row_dim</code>		<code>\g_@@ht_row_one_dim</code>	330, 331, 480, 481, 705
.	315, 316, 335, 336, 486, 487, 690, 716	<code>\g_@@ht_row_zero_dim</code>	
<code>\g_@@dp_row_zero_dim</code>	323, 324, 476, 477, 684	325, 326, 478, 479, 684, 705
<code>\c_@@draft_bool</code>		<code>\@@i:</code>	2088, 2090,
.	10, 11, 35, 404, 1778, 1849, 1881, 1933	2091, 2092, 2093, 2102, 2107, 2111, 2112,	
<code>\@@draw_Cdots:nn</code>	1433	2113, 2114, 2120, 2121, 2122, 2123, 2176,	
<code>\@@draw_Ddots:nn</code>	1528		

2178, 2181, 2182, 2186, 2187, 2213, 2219, 2222, 2228, 2231, 2262, 2271, 2276, 2283, 2286	\g_@@_medium_nodes_bool 142, 462, 1154, 2305
\@@_iddots 1719, 1753	\l_@@_medium_nodes_bool 141, 175, 462
\l_@@_iddots_int 1148, 1591, 1592	\@@_msg_new:nn 22, 33, 2431, 2437, 2444, 2450, 2456, 2462, 2470, 2475, 2481, 2486, 2491, 2616, 2621, 2628, 2634
\@@_if_not_empty_cell:nn 1058	\@@_msg_new:nnn .. 23, 2496, 2528, 2567, 2603
\@@_if_not_empty_cell:nnTF 1238, 1298, 1809, 1823	\@@_msg_redirect_name:nn 24, 249, 2428
\l_@@_impossible_line_bool 61, 1263, 1324, 1386, 1388, 1437, 1439, 1485, 1487, 1532, 1534, 1574, 1576	\@@_multicolumn:nnn 507, 1762
\l_@@_in_env_bool 53, 571, 572	\g_@@_multicolumn_cells_seq 520, 1767, 2114, 2123, 2237
\l_@@_initial_i_int 1161, 1202, 1269, 1272, 1293, 1299, 1305, 1316, 1323, 1339, 1401, 1449, 1800, 1809	\g_@@_multicolumn_sizes_seq 521, 1769, 2238
\l_@@_initial_j_int 1162, 1203, 1270, 1278, 1284, 1294, 1299, 1306, 1317, 1340, 1402, 1450, 1804, 1810, 1812	\l_@@_name_str 140, 203, 359, 361, 391, 393, 590, 599, 602, 896, 898, 974, 976, 1135, 1139, 2230, 2231, 2285, 2286
\l_@@_initial_open_bool 1165, 1271, 1275, 1281, 1287, 1291, 1320, 1327, 1354, 1395, 1425, 1443, 1473, 1491, 1504, 1538, 1580, 1635, 1682, 1798, 1805, 1813, 1830, 1855, 1939	\g_@@_names_seq 52, 200, 202, 2614
\@@_instruction_of_type:n 405, 407, 1728, 1734, 1740, 1746, 1752	\l_@@_nb_cols_int 2376, 2381, 2384, 2388, 2394, 2400
\l_@@_inter_dots_dim 72, 73, 1170, 1640, 1647, 1658, 1666, 1673, 1678, 1690, 1698, 1887, 1890	\l_@@_nb_rows_int 2375, 2380, 2391
\@@_j: 2095, 2097, 2098, 2099, 2100, 2105, 2107, 2111, 2114, 2116, 2117, 2120, 2123, 2125, 2126, 2189, 2191, 2195, 2197, 2201, 2202, 2215, 2218, 2221, 2228, 2231, 2263, 2270, 2275, 2283, 2286	\@@_node_for_multicolumn:nn 2239, 2265
\l_@@_l_dim . 1616, 1617, 1633, 1640, 1646, 1657, 1665, 1673, 1678, 1690, 1691, 1698, 1699	\l_@@_nullify_dots_bool 138, 174, 1729, 1735, 1741, 1747, 1753
\g_@@_large_nodes_bool 144, 463, 542, 1156, 1160, 1904	\@@_old_multicolumn 1761, 1764
\l_@@_large_nodes_bool 143, 176, 463	\l_@@_parallelize_diags_bool 135, 136, 171, 1145, 1547, 1589
\l_@@_last_col_bool 2712	\l_@@_pos_env_str 132, 133, 273, 274, 275, 429, 702, 711
\g_@@_last_col_found_bool 91, 554, 746, 818, 918, 1123	\@@_pre_array: 448, 606
\l_@@_last_col_int 89, 90, 268, 278, 286, 645, 1043, 1045, 2389, 2395, 2401, 2645, 2653, 2661, 2669, 2677, 2685, 2694, 2703, 2721, 2730, 2738	\c_@@_preamble_first_col_tl 637, 846
\l_@@_last_row_int 86, 87, 210, 288, 308, 440, 587, 594, 601, 670, 674, 677, 687, 709, 859, 861, 930, 932, 1125, 2002, 2009, 2017, 2027, 2034, 2045, 2397, 2464	\c_@@_preamble_last_col_tl 646, 914
\l_@@_last_row_without_value_bool 88, 589, 672, 1127	\l_@@_radius_dim 76, 77, 1169, 1707
\g_@@_last_vdotted_col_int 543, 545, 551, 553	\@@_renew_NC@rewrite@S: 117, 552
\@@_ldots 1715, 1729	\l_@@_renew_dots_bool 172, 510, 2421
\g_@@_left_delim_dim 607, 611, 623, 906	\@@_renew_matrix: 237, 2355, 2423
\l_@@_left_margin_dim 145, 179, 655, 907, 1922, 2206	\@@_renewcolumntype:nn 371, 528, 529
\l_@@_letter_for_dotted_lines_str 254, 260, 261, 531, 532, 2630	\@@_restore_iRow_jCol: ... 1114, 1192, 1194
\l_@@_light_syntax_bool . 131, 157, 658, 663	\@@_retrieve_coords:nn 1330, 1351, 1393, 1441, 1489, 1502, 1536, 1578
\@@_line:nn 1187, 1838	\c_@@_revtex_bool 26, 28, 31, 420
\@@_line_i:nn 1842, 1847	\g_@@_right_delim_dim ... 608, 612, 631, 954
\@@_line_with_light_syntax:n 767, 780	\l_@@_right_margin_dim 146, 181, 667, 955, 1922, 2209
\@@_line_with_light_syntax_i:n 766, 772, 783	\g_@@_row_total_int 523, 1124, 1133, 1140, 2088, 2102, 2213
\g_@@_max_cell_width_dim 342, 343, 581, 808, 2053, 2079	\g_@@_rows_seq 762, 764, 765, 767
	\l_@@_save_iRow_int 80, 451, 1196
	\l_@@_save_jCol_int 81, 454, 1197
	\@@_set_size:n 2373, 2382
	\c_@@_siunitx_loaded_bool . 92, 96, 101, 552
	\l_@@_small_bool 169, 304, 457, 853, 924, 1167, 1886, 1893
	\l_@@_stop_loop_bool 1206, 1207, 1235, 1239, 1266, 1267, 1295, 1300
	\c_@@_table_collect_begin_tl . 109, 111, 124
	\c_@@_table_print_tl 112, 113, 126
	\@@_test_if_math_mode: 55, 570, 997, 1005, 1013, 1021, 1029
	\l_@@_the_array_box 634, 654, 721, 734
	\g_@@_type_env_str 78, 564, 566, 988, 989, 995, 996, 1003, 1004, 1011, 1012, 1019, 1020, 1027, 1028, 1037, 1191, 2409, 2439, 2446, 2465, 2477, 2570, 2623, 2631
	\@@_vdots 1717, 1741
	\@@_vdottedline:n 547, 1928

<code>\hfil</code>	529
<code>\hfill</code>	1897
<code>\hrule</code>	442
<code>\Hspace</code>	505
<code>\hspace</code>	1759
<code>\hss</code>	529, 1894

I

<code>\ialign</code>	464, 488
<code>\iddots</code>	503
<code>\iddots</code>	37, 516, 1719
if commands:	
<code>\if_mode_math:</code>	57
<code>\ifstandalone</code>	576
int commands:	
<code>\int_add:Nn</code>	1209, 1210, 1293, 1294
<code>\int_compare:nNnTF</code>	295,
297, 305, 306, 308, 321, 328, 438, 440, 543,	
587, 636, 645, 670, 674, 681, 687, 693, 700,	
709, 795, 1043, 1078, 1110, 1125, 1212,	
1214, 1218, 1220, 1224, 1226, 1272, 1274,	
1278, 1280, 1284, 1286, 1550, 1592, 1765,	
1802, 1816, 1930, 1994, 1996, 1998, 2000,	
2002, 2007, 2009, 2015, 2017, 2023, 2025,	
2027, 2032, 2034, 2044, 2045, 2295, 2325,	
2385, 2387, 2389, 2393, 2395, 2397, 2399, 2401	
<code>\int_compare:nTF</code>	253
<code>\int_compare_p:nNn</code>	856,
859, 861, 927, 930, 932, 1916, 2320, 2321	
<code>\int_eval:n</code>	1045, 1768,
1837, 2182, 2186, 2197, 2201, 2275, 2311, 2312	
<code>\int_gadd:Nn</code>	1771
<code>\int_gdecr:N</code>	1123
<code>\int_gincr:N</code> 294, 314, 579, 824, 830, 919, 2059	
<code>\int_gset:Nn</code>	300, 522, 553, 817, 920
<code>\int_gset_eq:NN</code>	
545, 677, 1122, 1124, 1196, 1197	
<code>\int_gsub:Nn</code>	1126
<code>\int_gzero:N</code>	435
<code>\int_gzero_new:N</code> 452, 455, 523, 524, 551	
<code>\int_incr:N</code>	1549, 1591
<code>\int_max:nn</code>	301, 921
<code>\int_new:N</code>	49, 50, 80, 81, 82, 84, 86, 89
<code>\int_set:Nn</code>	83, 85, 87,
90, 268, 594, 601, 1202, 1203, 1204, 1205,	
1639, 1643, 1654, 1662, 1680, 1800, 1801,	
1804, 1808, 1810, 1812, 1818, 1822, 1824,	
1826, 2174, 2175, 2375, 2376, 2645, 2653,	
2661, 2669, 2677, 2685, 2686, 2694, 2695,	
2703, 2704, 2713, 2721, 2722, 2730, 2738, 2739	
<code>\int_set_eq:NN</code>	451, 454
<code>\int_step_inline:nn</code>	1952
<code>\int_step_inline:nnn</code>	1703, 1833
<code>\int_step_variable:nNn</code>	2176, 2189
<code>\int_step_variable:nnNn</code>	
2088, 2095, 2102, 2104, 2213, 2215	
<code>\int_sub:Nn</code>	1233, 1234, 1269, 1270
<code>\int_use:N</code> 355, 356, 357, 362, 363, 387, 388,	
389, 394, 395, 413, 414, 547, 592, 595, 799,	
816, 842, 843, 892, 893, 899, 970, 971, 972,	
977, 978, 1062, 1068, 1069, 1070, 1076,	
1079, 1091, 1092, 1093, 1132, 1133, 1140,	
1184, 1244, 1245, 1254, 1255, 1256, 1262,	
1305, 1306, 1315, 1316, 1317, 1323, 1338,	

1339, 1340, 1344, 1345, 1346, 1357, 1361,	
1370, 1371, 1376, 1400, 1401, 1402, 1415,	
1416, 1417, 1448, 1449, 1450, 1463, 1464,	
1465, 1613, 1723, 1724, 1768, 1789, 1790,	
1859, 1861, 1905, 1906, 1942, 1969, 1974,	
1981, 2064, 2067, 2078, 2107, 2110, 2120,	
2208, 2228, 2245, 2248, 2249, 2255, 2283,	
2299, 2301, 2302, 2309, 2310, 2464, 2465, 2466	
<code>\int_zero:N</code>	208, 209, 278, 283, 286, 287
<code>\int_zero_new:N</code>	
1147, 1148, 1161, 1162, 1163, 1164, 2380, 2381	
<code>\c_one_int</code>	253, 295, 297,
328, 1126, 1387, 1438, 1486, 1533, 1550, 1592	
<code>\g_tmpa_int</code>	817, 824, 830, 843
<code>\l_tmpa_int</code>	1639, 1643, 1654,
1662, 1690, 1698, 1703, 1808, 1809, 1822, 1823	
<code>\l_tmpb_int</code>	1680, 1692, 1700
<code>\c_zero_int</code>	305, 306, 321, 636,
681, 693, 700, 856, 927, 1110, 1387, 1438,	
1486, 1916, 1930, 1994, 1996, 1998, 2000,	
2007, 2015, 2023, 2032, 2044, 2385, 2387,	
2393, 2399, 2686, 2695, 2704, 2713, 2722, 2739	
iow commands:	
<code>\iow_now:Nn</code>	1129, 1130,
1137, 1143, 2074, 2075, 2081, 2252, 2253, 2258	

K

<code>\kern</code>	46
keys commands:	
<code>\keys_define:nn</code>	
153, 190, 213, 235, 264, 271, 281, 2048, 2419	
<code>\l_keys_key_tl</code>	2493, 2498, 2530, 2569
<code>\keys_set:nn</code>	263, 584, 585, 1038, 2061
<code>\l_keys_value_tl</code>	2442, 2605

L

<code>\ldots</code>	499
<code>\ldots</code>	512, 1715
<code>\left</code>	620, 628, 727
<code>\line</code>	1187, 2433
<code>\lineskip</code>	684, 690

M

<code>\makebox</code>	399
math commands:	
<code>\c_math_toggle_token</code>	
303, 340, 619, 621, 627, 629, 657,	
666, 726, 740, 852, 869, 923, 940, 1892, 1895	
<code>\mathinner</code>	39
<code>\mkern</code>	41, 43, 45, 46
msg commands:	
<code>\msg_error:nn</code>	17
<code>\msg_error:nnn</code>	18, 1261, 1322
<code>\msg_error:nnnn</code>	19, 2323
<code>\msg_fatal:nn</code>	20, 21
<code>\msg_new:nnn</code>	22
<code>\msg_new:nnnn</code>	23
<code>\msg_redirect_name:nnn</code>	25
<code>\msg_warning:nn</code>	36
<code>\multicolumn</code>	507, 1761, 1775, 1781, 1793
<code>\myfiledate</code>	7
<code>\myfileversion</code>	8

N

<code>\newcolumntype</code>	373, 496, 497, 498, 532
-----------------------------------	-------------------------

`\NewDocumentCommand` 262, 1726,
 1732, 1738, 1744, 1750, 1780, 1784, 2290, 2378
`\NewDocumentEnvironment`
 562, 753, 759, 985, 993, 1001, 1009, 1017,
 1025, 1035, 2057, 2641, 2649, 2657, 2665,
 2673, 2681, 2690, 2699, 2708, 2717, 2726, 2734
`\NewExpandableDocumentCommand` 1612
`\NiceArrayWithDelims`
 990, 998, 1006, 1014, 1022, 1030, 2731, 2740
`\NiceMatrixLastEnv` 1612
`\NiceMatrixOptions` 262, 2499
`\noalign` 432, 470, 1902
`\node` 352,
 385, 887, 965, 2223, 2278, 2330, 2343, 2350
`\normalbaselines` 456
`\nulldelimiterspace` 616, 626

O

`\omit` 795, 796, 823, 829
`\OnlyMainNiceMatrix` 509, 2041

P

`\path` 1865, 2338
`\pgfpathcircle` 1705
`\pgfpoint` 1706
`\pgfpointanchor` 1095, 1097
`\pgfusepath` 1708
`\phantom` 1729, 1735, 1741, 1747, 1753
`\pNiceArray` 2646, 2687
`\pNiceMatrix` 2358
 prg commands:
`\prg_do_nothing:` 105, 114, 470, 1882
`\prg_replicate:nn`
 819, 820, 1781, 1793, 2388, 2391, 2394, 2400
`\prg_return_false:` 1072, 1085, 1102
`\prg_return_true:` 1063, 1103
`\prg_set_conditional:Npnn` 1058
`\ProcessKeysOptions` 2430
`\ProcessOptions` 13
`\ProvideDocumentCommand` 37
`\ProvidesExplPackage` 5

Q

quark commands:
`\q_stop` 1836,
 1843, 1844, 2260, 2267, 2291, 2292, 2373, 2382

R

`\raise` 42, 44, 46
`\relax` 13, 526, 527
`\renewcommand` 119
`\RenewDocumentEnvironment`
 2357, 2360, 2363, 2366, 2369
`\RequirePackage` 1, 3, 4, 14, 15, 16
`\right` 620, 628, 739

S

`\scriptstyle` 304, 853, 924, 1893
 seq commands:
`\seq_gclear_new:N` 520, 521, 762, 774
`\seq_gpop_left:NN` 765, 776
`\seq_gput_left:Nn` 202, 1767, 1769
`\seq_gset_split:Nnn` 764, 775
`\seq_if_in:NnTF` 200, 2114, 2123

`\seq_map_function:NN` 767, 778
`\seq_mapthread_function:NNN` 2236
`\seq_new:N` 52
`\seq_use:Nnnn` 2614

skip commands:

`\skip_horizontal:N` .. 797, 814, 825, 831, 834
`\skip_horizontal:n` 536,
 655, 656, 667, 668, 695, 696, 733, 735,
 748, 749, 800, 832, 904, 911, 949, 952, 1918
`\skip_vertical:n` 443, 730, 737
`\c_zero_skip` 475, 491

str commands:

`\c_backslash_str` 2410
`\c_colon_str` 261
`\str_gclear:N` 1191
`\str_gset:Nn` 566,
 989, 996, 1004, 1012, 1020, 1028, 1037, 2409
`\str_if_empty:NTF`
 359, 391, 564, 590, 896, 974, 988,
 995, 1003, 1011, 1019, 1027, 1135, 2230, 2285
`\str_if_eq:nnTF` 193, 245, 702, 711
`\str_new:N` 78, 132, 140, 260
`\str_set:Nn` 133, 199, 254, 273, 274, 275
`\str_set_eq:NN` 203, 261
`\l_tmpa_str` 199, 200, 202, 203

T

`\tabskip` 475, 491

TeX and L^AT_EX commands:

`\@acol` 424
`\@acoll` 422
`\@acolr` 423
`\@addtopreamble` 582
`\@array@array` 426
`\@arrayacol` 422, 423, 424
`\@arrayrule` 582
`\@arstrutbox` 477, 479, 481, 483, 485, 487
`\@currenvir` 2636
`\@halignto` 425
`\@height` 442
`\@ifclassloaded` 27, 30
`\@ifnextchar` 525
`\@ifpackageloaded` 65, 95
`\@mainaux` 1129, 1130,
 1137, 1143, 2074, 2075, 2081, 2252, 2253, 2258
`\@temptokena` 104, 107, 121, 123
`\c@iRow` . 305, 308, 314, 321, 328, 356, 362,
 388, 394, 413, 438, 440, 451, 452, 522, 674,
 677, 856, 861, 893, 899, 927, 932, 971, 977,
 1110, 1124, 1126, 1196, 1212, 1723, 1768,
 1789, 1969, 1981, 2000, 2002, 2007, 2009,
 2015, 2017, 2025, 2027, 2032, 2034, 2044,
 2045, 2176, 2301, 2309, 2311, 2320, 2465, 2466
`\c@jCol` 294,
 295, 301, 306, 357, 363, 389, 395, 414, 435,
 454, 455, 543, 545, 547, 919, 921, 972, 978,
 1122, 1123, 1197, 1224, 1284, 1371, 1376,
 1723, 1768, 1771, 1790, 1816, 1952, 1996,
 2023, 2189, 2208, 2249, 2302, 2310, 2312, 2321
`\c@MaxMatrixCols` 1044
`\CT@arc@` 68
`\CT@everycr` 468
`\CT@row@color` 470
`\ifmeasuring@` 198

<code>\NC@find</code>	105, 128	<code>\tl_gset:Nn</code>	107, 111, 113
<code>\NC@find@W</code>	527	<code>\tl_if_empty:nTF</code>	266, 276, 284, 782
<code>\NC@find@w</code>	526	<code>\tl_item:Nn</code>	110, 111, 113
<code>\NC@rewrite@S</code>	106, 119	<code>\tl_new:N</code>	79, 109, 112, 151
<code>\new@ifnextchar</code>	525	<code>\tl_put_left:Nn</code>	637, 642
<code>\p@</code>	42, 44, 46	<code>\tl_put_right:Nn</code>	646, 651
<code>\pgf@x</code>	1096, 1098, 1341, 1347, 1362, 1377, 1867, 1870, 1947, 1948, 1966, 1970, 1979, 1982, 2117, 2126, 2246, 2250, 2338	<code>\tl_set:Nn</code>	110, 152, 635, 1088
<code>\pgf@y</code>	1342, 1348, 1868, 1871, 1949, 1958, 1967, 1971, 2113, 2122, 2329	<code>\tl_set_rescan:Nnn</code>	530, 763
<code>\pgfutil@firstofone</code> ...	1337, 1343, 1360, 1375, 1866, 1869, 1945, 1955, 1964, 1968, 1977, 1980, 2109, 2119, 2244, 2247, 2328, 2337	<code>\tl_use:N</code>	2493, 2498, 2530, 2569
<code>\tikz@library@external@loaded</code> ...	573, 1120	<code>\g_tmpa_tl</code>	107, 110, 113
<code>\tikz@parse@node</code>	1337, 1343, 1360, 1375, 1866, 1869, 1945, 1955, 1964, 1968, 1977, 1980, 2109, 2119, 2244, 2247, 2328, 2337	<code>\l_tmpa_tl</code> 110, 111, 635, 637, 642, 646, 651, 754, 761, 765, 766, 776, 777, 1088, 1095, 1097	
tex commands:		token commands:	
<code>\tex_the:D</code>	123	<code>\token_to_str:N</code>	2433, 2478, 2499
<code>\theiRow</code>	450, 1196		
<code>\thejCol</code>	453, 1197		
<code>\tikz</code>	345, 384, 798, 815, 839, 880, 958		
<code>\tikzset</code>	575, 577, 1121, 1178, 2136, 2148, 2160, 2163		
tl commands:			
<code>\tl_const:Nn</code>	846, 914		
<code>\tl_count:n</code>	253		
<code>\tl_gclear:N</code>	1189		
<code>\tl_gclear_new:N</code>	555, 556, 557, 558, 559, 560		
<code>\tl_gput_left:Nn</code>	2306		
<code>\tl_gput_right:Nn</code>	409, 546, 1786		

11	Advanced features	12
11.1	The option <code>small</code>	12
11.2	The counters <code>iRow</code> and <code>jCol</code>	12
11.3	The option <code>hlines</code>	13
11.4	The option <code>light-syntax</code>	13
11.5	Utilisation of the column type <code>S</code> of <code>siunitx</code>	14
12	Technical remarks	14
12.1	Definition of new column types	14
12.2	Intersections of dotted lines	14
12.3	The names of the Tikz nodes created by <code>nicematrix</code>	15
12.4	Diagonal lines	15
12.5	The “empty” cells	16
12.6	The option <code>exterior-arraycolsep</code>	16
12.7	The class option <code>draft</code>	16
12.8	A technical problem with the argument of <code>\</code>	17
12.9	Obsolete environments	17
13	Examples	17
13.1	Dotted lines	17
13.2	Width of the columns	19
13.3	How to highlight cells of the matrix	20
13.4	Direct utilisation of the Tikz nodes	23
14	Implementation	24
14.1	Declaration of the package and extensions loaded	24
14.2	Technical definitions	25
14.2.1	Variables for the exterior rows and columns	27
14.2.2	The column <code>S</code> of <code>siunitx</code>	28
14.3	The options	30
14.4	Important code used by <code>{NiceArrayWithDelims}</code>	34
14.5	The environment <code>{NiceArrayWithDelims}</code>	41
14.6	The environment <code>{NiceMatrix}</code> and its variants	51
14.7	How to know whether a cell is “empty”	51
14.8	After the construction of the array	53
14.9	The actual instructions for drawing the dotted line with Tikz	64
14.10	User commands available in the new environments	66
14.11	The command <code>\line</code> accessible in code-after	68
14.12	The commands to draw dotted lines to separate columns and rows	69
14.13	The vertical rules	72
14.14	The environment <code>{NiceMatrixBlock}</code>	73
14.15	The extra nodes	74
14.16	Block matrices	79
14.17	How to draw the dotted lines transparently	80
14.18	Automatic arrays	81
14.19	We process the options	81
14.20	Error messages of the package	82
14.21	Obsolete environments	85
15	History	87
	Index	90