

The package `nicematrix`*

F. Pantigny
 fpantigny@wanadoo.fr

August 15, 2019

Abstract

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

1 Presentation

This package can be used with `xelatex`, `lualatex`, `pdflatex` but also by the classical workflow `latex-dvips-ps2pdf` (or Adobe Distiller). Two or three compilations may be necessary. This package requires and loads the packages `expl3`, `l3keys2e`, `xparse`, `array`, `amsmath` and `tikz`. It also loads the Tikz library `fit`.

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

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

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

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 3.0 of `nicematrix`, at the date of 2019/08/15.

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. The environment `{NiceArray}` and its variants provide also options to draw exterior rows and columns. 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.

```
\begin{bNiceMatrix}
a_1      & \Cdots &      & & a_1 \\
\Vdots   & a_2      & \Cdots & & a_2 \\
          & \Vdots & \Ddots \\
\\
a_1      & a_2      &      & & a_n \\
\end{bNiceMatrix}
```

$$\left[\begin{array}{cccc} a_1 & \cdots & \cdots & a_1 \\ \vdots & & & \\ \vdots & a_2 & \cdots & a_2 \\ \vdots & \vdots & \ddots & \\ a_1 & a_2 & & a_n \end{array} \right]$$

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

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

$$\left[\begin{array}{cccc} 0 & \cdots & \cdots & 0 \\ \vdots & & & \\ \vdots & & & \\ 0 & \cdots & \cdots & 0 \end{array} \right]$$

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: `\iddots`. 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. 12).

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

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

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

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

However, useless computations are performed by TeX before detecting that both instructions would eventually yield the same dotted line. That's why the package `nicematrix` provides starred versions of `\Ldots`, `\Cdots`, etc.: `\Ldots*`, `\Cdots*`, etc. These versions are simply equivalent to `\hphantom{\ldots}`, `\hphantom{\cdots}`, etc. The user should use these starred versions whenever a classical version has already been used for the same dotted line.

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

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

In fact, in this example, it would be possible to draw the same matrix without starred commands with the following code:

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

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

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

$$\begin{bmatrix} 0 & \cdots & \hspace{1cm} & 0 \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ 0 & \cdots & & 0 \end{bmatrix}$$

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

⁶Nevertheless, the best way to fix the width of a column is to use the environment `{NiceArray}` (or one of its variants) with a column of type `w` or `W`: see p. 10

3.1 The option `nullify-dots`

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

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

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

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

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

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

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).

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

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.

```

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

```

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

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

```

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

```

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

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 `{matrix}`. This option can be set as option of `\usepackage` or with the command `\NiceMatrixOptions`.

In fact, this option is an alias for the conjunction of two options: `renew-dots` and `renew-matrix`.

- The option `renew-dots`

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

- The option `renew-matrix`

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

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

```

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

```

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

4 The Tikz nodes created by `nicematrix`

The package `nicematrix` creates a Tikz node for each cell of the considered array. These nodes are used to draw the dotted lines between the cells of the matrix. However, the user may wish to use directly these nodes. It's possible. First, the user have to give a name to the array (with the key

called **name**). Then, the nodes are accessible through the names “*name-i-j*” where *name* is the name given to the array and *i* and *j* the numbers of the row and the column of the considered cell.

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

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

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

```
7 & 8 & 9
```

```
\end{pNiceMatrix}
```

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

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

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

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

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

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

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

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.⁷

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

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**.⁸

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

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.

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

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

⁸The options **left-margin** and **right-margin** take dimensions as values but, if no value is given, the default value is used, which is `\arraycolsep`. There is also an option **margin** to fix both **left-margin** and **right-margin** to the same value.

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. 16).

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`⁹ 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.¹⁰

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]
```

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

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

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

⁹The 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.

¹⁰In a command `\multicolumn`, one should also use the letters `L`, `C`, `R`.

```

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

```

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

With the environment `{NiceArray}` and its the variants, it's possible to compose exterior rows and columns with the options `first-row`, `last-row`, `first-col` and `last-col`. There is no specification of column to provide for the potential “first column” (it will automatically be a **R** column) and for the potential “last column” (it will automatically be a **L** column).

```

 $\begin{pNiceArray}{CCCC}[first-row,last-row,first-col,last-col]$ 
& 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 \\
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}

```

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

However, there is a particularity for the option `last-row`: when LaTeX composes an array (with the TeX command `\halign`) it composes it row by row and there is no direct way to know if we are at the last row before the composition of that row. That's why `nicematrix` writes in the `aux` file the number of rows of the array in order to use it at the next run. Nevertheless, it's possible to give directly the number of rows as the value of the key `last-row`. It that way, `nicematrix` will know the correct value by the first compilation.

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

```

\NiceMatrixOptions{code-for-first-row = \color{red},
                  code-for-first-col = \color{blue},
                  code-for-last-row = \color{green},
                  code-for-last-col = \color{magenta}}
 $\begin{pNiceArray}{CC|CC}[first-row,last-row=5,first-col,last-col]$ 
& C_1 & & C_2 & & C_3 & & C_4 & & \\
L_1 & a_{11} & & a_{12} & & a_{13} & & a_{14} & & \\
L_2 & a_{21} & & a_{22} & & a_{23} & & a_{24} & & \\
\hline
L_3 & a_{31} & & a_{32} & & a_{33} & & a_{34} & & \\
L_4 & a_{41} & & a_{42} & & a_{43} & & a_{44} & & \\
& C_1 & & C_2 & & C_3 & & C_4 & & \\
\end{pNiceArray}

```

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

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.¹¹
- The “first row” of an environment `{pNiceArray}` has the number 0, and not 1. Idem for the “first column”. This number is used for the names of the Tikz nodes (the names of these nodes are used, for example, by the command `\line` in `code-after`).
- 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).

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.

```


$$\begin{array}{ccc|c} 1 & 2 & 3 & L_1 \\ 4 & 5 & 6 & L_2 \\ 7 & 8 & 9 & L_3 \end{array}$$


```

7 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`).

```


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


```

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

```


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


```

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

¹¹The 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`.

```

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

```

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)

```

8 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`.

```

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

```

It's also possible to fix the width of all the columns of a matrix directly with the option `columns-width` (in all the environments of `nicematrix`).

```

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

```

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 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. **Two or three compilations may be necessary.**

```

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

```

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`.¹²

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

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

9 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` 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>

10 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`. The `d` columns of `dcolumn` are not supported by `nicematrix`.

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

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

¹²At this time, this is the only usage of the environment `\NiceMatrixBlock` but it may have other usages in the future.

11 Technical remarks

11.1 Diagonal lines

By default, all the diagonal lines¹³ 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):

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

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

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

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

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

The same example without parallelization:

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

11.2 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.

¹³We speak of the lines created by `\Ddots` and not the lines created by a command `\line` in `code-after`.

- A cell which contains a command `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots` or `\Iddots` and their starred versions 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`.

11.3 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 idea.¹⁴

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

However, the user can change this behaviour with the boolean option `exterior-arraycolsep` of the command `\NiceMatrixOptions`. With this option, `{NiceArray}` will insert the same horizontal spaces as the environment `{array}`.

This option is only for “compatibility” since the package `nicematrix` provides a more precise control with the options `left-margin`, `right-margin`, `extra-left-margin` and `extra-right-margin`.

11.4 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 `:`).¹⁵

That’s why, when the class option `draft` is used, the dotted lines are not drawn, for a faster compilation.

11.5 A technical problem with the argument of `\`

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

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

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

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

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

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

¹⁴In 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@{}}`.

¹⁵The 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.

11.6 Obsolete environments

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

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

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

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

12 Examples

12.1 Dotted lines

A tridiagonal matrix:

```

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

```

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

A permutation matrix:

```

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

```

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

An example with `\Iddots`:

```

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

```

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

An example with `\multicolumn`:

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

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

An example with `\Hdotsfor`:

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

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

An example for the resultant of two polynomials:

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

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

An example for a linear system:

```

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

```

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

12.2 Width of the columns

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

```

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

```

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

12.3 How to highlight cells of the matrix

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

nodes” of the diagonal of the matrix (with the Tikz key “`name suffix`”, it’s easy to use the “large nodes”).

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

```

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

```

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

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

```

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

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

```

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

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

```

\ExplSyntaxOn
\makeatletter

```

```

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

```

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

```

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

```

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

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

```

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

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

```

We obtain the following matrix.

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

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

```

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

```

We obtain the following matrix.

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

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

```

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

```

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

12.4 Direct utilisation of the Tikz nodes

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

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

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

```
\NiceMatrixOptions{nullify-dots}
```

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

```

$\begin{array}{cc}
&

```

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

```

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

```

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

```

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

```

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

```
\begin{bNiceArray}{CC>{\strut}CCC}
      &      & & & \\\
      &      & & \Vdots & \\\
\Cdots &      & & c_{ij} & \\\
\\
\\
\end{bNiceArray}
\end{array}$

\end{NiceMatrixBlock}

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

$$L_i \begin{bmatrix} a_{11} & \dots & a_{nn} \\ \vdots & & \vdots \\ a_{i1} & \dots & a_{in} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} b_{11} & \dots & b_{1n} \\ \vdots & & \vdots \\ b_{k1} & \dots & b_{kn} \\ \vdots & & \vdots \\ b_{n1} & \dots & b_{nn} \end{bmatrix} \begin{bmatrix} c_{1j} \\ \vdots \\ c_{ij} \\ \vdots \\ c_{nj} \end{bmatrix}$$

12.5 Block matrices

In the following example, we use the “large nodes” to construct a block matrix (the dashed lines have been drawn with `arydashln`).

```
\NiceMatrixOptions{letter-for-dotted-lines = V}
\begin{pNiceArray}{CC:CC}[
  create-extra-nodes,
  code-after = { \tikz \node [fit = (1-1-large) (2-2-large), inner sep = 0 pt]
                  {$0_{22}$} ; } ]
      &      & a_{13} & a_{14} & \\\
      &      & a_{23} & a_{24} & \\\
\hdashline
a_{31} & a_{32} & a_{33} & a_{34} & \\\
a_{41} & a_{42} & a_{34} & a_{44} & \\
\end{pNiceArray}
```

$$D = \left(\begin{array}{cc|cc} 0_{22} & & a_{13} & a_{14} \\ & & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{34} & a_{44} \end{array} \right)$$

13 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 independant of its implementation. Unfortunately, it was not possible to be strictly independant: the package `nicematrix` relies upon the fact that the package `{array}` uses `\ialign` to begin the `\halign`.

The desire to do no modification to existing code leads to complications in the code of this extension.

13.1 Declaration of the package and extensions loaded

First, `tikz` and the Tikz library `fit` are loaded before the `\ProvidesExplPackage`. They are loaded this way because `\usetikzlibrary` in `expl3` code fails.¹⁶

```
1 <@@=nm>
2 \RequirePackage{tikz}
3 \usetikzlibrary{fit}
4 \RequirePackage{expl3}[2019/02/15]
```

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

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

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

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

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

We load `array` and `amsmath`.

```
15 \RequirePackage { array }
16 \RequirePackage { amsmath }
17 \RequirePackage { xparse } [ 2018-10-17 ]

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

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

¹⁶cf. tex.stackexchange.com/questions/57424/using-of-usetikzlibrary-in-an-expl3-package-fails

13.2 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__nm_revtex_bool
27 \ifclassloaded { revtex4-1 }
28   { \bool_set_true:N \c__nm_revtex_bool }
29   { }
30 \ifclassloaded { revtex4-2 }
31   { \bool_set_true:N \c__nm_revtex_bool }
32   { }
33 \__nm_msg_new:nn { Draft~mode }
34   { The~compilation-is-in~draft-mode:-the~dotted-lines~won't~be~drawn. }
35 \bool_if:NT \c__nm_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__nm_env_int

```

The dimension `\l_@@_columns_width_dim` will be used when the options specify that all the columns must have the same width.

```

50 \dim_new:N \l__nm_columns_width_dim

```

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

```

51 \seq_new:N \g__nm_names_seq

```

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

```

52 \bool_new:N \l__nm_in_env_bool

```

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

```

53 \bool_new:N \l__nm_NiceArray_bool
54 \bool_new:N \g__nm_NiceArray_bool

```

```

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

61 \__nm_msg_new:nn { Yet~in~env }
62 {
63   Environments~\{NiceArray\}~(or~\{NiceMatrix\},~etc.)~can't~be~
64   nested.\\
65   This~error~is~fatal.
66 }

67 \__nm_msg_new:nn { Outside~math~mode }
68 {
69   The~environment~\{@currentvir\}~can~be~used~only~in~math~mode~
70   (and~not~in~\token_to_str:N \vcenter).\\
71   This~error~is~fatal.
72 }

```

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

```

73 \bool_new:N \c__nm_colortbl_loaded_bool
74 \AtBeginDocument
75 {
76   \@ifpackageloaded { colortbl }
77     { \bool_set_true:N \c__nm_colortbl_loaded_bool }
78     { }
79 }

```

13.2.1 Variables for the exterior rows and columns

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

- **First row**

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

```

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

```

- **First column**

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

```

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

```

- **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 has not still been read in the `aux`).

```

86 \int_new:N \l__nm_last_row_int
87 \int_set:Nn \l__nm_last_row_int { -2 }
88 \int_new:N \g__nm_last_row_int

```

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”.¹⁷

```

89 \bool_new:N \l__nm_last_row_without_value_bool
90 \bool_new:N \g__nm_last_row_without_value_bool

```

- **Last column**

For the eventual “last column”, we have a boolean and not an integer.

```

91 \bool_new:N \l__nm_last_col_bool

```

However, we have also another 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.

```

92 \bool_new:N \g__nm_last_col_found_bool

```

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

13.2.2 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`.

```

93 \bool_new:N \c__nm_siunitx_loaded_bool
94 \AtBeginDocument
95 {
96   \@ifpackageloaded { siunitx }
97   { \bool_set_true:N \c__nm_siunitx_loaded_bool }
98   { }
99 }

```

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:

¹⁷We 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:`.

```

100 \cs_set_protected:Npn \_nm_adapt_S_column:
101 {

```

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.¹⁸

```

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

```

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

```

106     \cs_set_eq:NN \NC@find \prg_do_nothing:
107     \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`).

```

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

```

¹⁸Indeed, `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).

```

115     \cs_gset_eq:NN \_nm_adapt_S_column: \prg_do_nothing:
116   }
117 }

```

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).

```

118 \cs_new_protected:Npn \_nm_renew_NC@rewrite@S:
119 {
120   \renewcommand*{\NC@rewrite@S}[1] []
121   {
122     \@temptokena \exp_after:wN
123     {
124       \tex_the:D \@temptokena
125       > { \_nm_Cell: \_nm_table_collect_begin_tl S {##1} }
126       c
127       < { \_nm_table_print_tl \_nm_end_Cell: }
128     }
129     \NC@find
130   }
131 }

```

13.3 The options

```

132 \_nm_msg_new:nn { Option-Transparent-suppressed }
133 {
134   The~option~'Transparent'~has~been~renamed~'transparent'.\\
135   However,~you~can~go~on~for~this~time.
136 }
137 \_nm_msg_new:nn { Option-RenewMatrix-suppressed }
138 {
139   The~option~'RenewMatrix'~has~been~renamed~'renew-matrix'.\\
140   However,~you~can~go~on~for~this~time.
141 }

```

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.

```

142 \str_new:N \l__nm_pos_env_str
143 \str_set:Nn \l__nm_pos_env_str c

```

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

```

144 \bool_new:N \l__nm_exterior_arraycolsep_bool

```

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

```

145 \bool_new:N \l__nm_parallelize_diags_bool
146 \bool_set_true:N \l__nm_parallelize_diags_bool

```

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

```

147 \bool_new:N \l__nm_hlines_bool

```

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

```
148 \bool_new:N \l__nm_nullify_dots_bool
```

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

```
149 \bool_new:N \l__nm_auto_columns_width_bool
```

We don't want to patch any existing code. That's why some code must be executed in a `\group_insert_after:N`. That's why the parameters used in that code must be transferred outside the current group. To do this, we copy those quantities in global variables just before the `\group_insert_after:N`. Therefore, for those quantities, we have two parameters, one local and one global. For example, we have `\l_@@_name_str` and `\g_@@_name_str`.

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.

```
150 \str_new:N \l__nm_name_str
```

```
151 \str_new:N \g__nm_name_str
```

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

```
152 \bool_new:N \l__nm_extra_nodes_bool
```

```
153 \bool_new:N \g__nm_extra_nodes_bool
```

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

```
154 \dim_new:N \l__nm_left_margin_dim
```

```
155 \dim_new:N \l__nm_right_margin_dim
```

```
156 \dim_new:N \g__nm_left_margin_dim
```

```
157 \dim_new:N \g__nm_right_margin_dim
```

```
158 \dim_new:N \g__nm_width_last_col_dim
```

```
159 \dim_new:N \g__nm_width_first_col_dim
```

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

```
160 \dim_new:N \l__nm_extra_left_margin_dim
```

```
161 \dim_new:N \l__nm_extra_right_margin_dim
```

```
162 \dim_new:N \g__nm_extra_right_margin_dim
```

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

```
163 \keys_define:nn { NiceMatrix / Global }
```

```
164 {
```

```
165   hlines .bool_set:N = \l__nm_hlines_bool ,
```

```
166   parallelize-diags .bool_set:N = \l__nm_parallelize_diags_bool ,
```

```
167   parallelize-diags .default:n = true ,
```

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

```
168   renew-dots .bool_set:N = \l__nm_renew_dots_bool ,
```

```
169   renew-dots .default:n = true ,
```

```
170   nullify-dots .bool_set:N = \l__nm_nullify_dots_bool ,
```

```
171   nullify-dots .default:n = true ,
```

An option to test whether the extra nodes will be created (these nodes are the “medium nodes” and “large nodes”). In some circumstances, the extra nodes are created automatically, for example when a dotted line has an “open” extremity.¹⁹

```

172   create-extra-nodes .bool_set:N = \l__nm_extra_nodes_bool ,
173   create-extra-nodes .default:n = true,
174   left-margin .dim_set:N = \l__nm_left_margin_dim ,
175   left-margin .default:n = \arraycolsep ,
176   right-margin .dim_set:N = \l__nm_right_margin_dim ,
177   right-margin .default:n = \arraycolsep ,
178   margin .meta:n = { left-margin = #1 , right-margin = #1 } ,
179   margin .default:n = \arraycolsep ,
180   extra-left-margin .dim_set:N = \l__nm_extra_left_margin_dim ,
181   extra-right-margin .dim_set:N = \l__nm_extra_right_margin_dim ,
182   extra-margin .meta:n =
183     { extra-left-margin = #1 , extra-right-margin = #1 } ,
184 }

```

The following set of keys concerns the options to *customize* the exterior columns, that is to say the options `code-for-first-row`, etc.

```

185 \keys_define:nn { NiceMatrix / Code }
186 {
187   code-for-first-col .tl_set:N = \l__nm_code_for_first_col_tl ,
188   code-for-first-col .value_required:n = true ,
189   code-for-last-col .tl_set:N = \l__nm_code_for_last_col_tl ,
190   code-for-last-col .value_required:n = true ,
191   code-for-first-row .tl_set:N = \l__nm_code_for_first_row_tl ,
192   code-for-first-row .value_required:n = true ,
193   code-for-last-row .tl_set:N = \l__nm_code_for_last_row_tl ,
194   code-for-last-row .value_required:n = true ,
195 }

```

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

```

196 \keys_define:nn { NiceMatrix / Env }
197 {
198   columns-width .code:n =
199     \str_if_eq:nnTF { #1 } { auto }
200     { \bool_set_true:N \l__nm_auto_columns_width_bool }
201     { \dim_set:Nn \l__nm_columns_width_dim { #1 } } ,
202   columns-width .value_required:n = true ,
203   name .code:n =
204     \str_set:Nn \l_tmpa_str { #1 }
205     \seq_if_in:NVTF \g__nm_names_seq \l_tmpa_str
206     { \__nm_error:nn { Duplicate-name } { #1 } }
207     { \seq_gput_left:NV \g__nm_names_seq \l_tmpa_str }
208     \str_set_eq:NN \l__nm_name_str \l_tmpa_str ,
209   name .value_required:n = true ,
210   code-after .tl_gset:N = \g__nm_code_after_tl ,
211   code-after .initial:n = \c_empty_tl ,
212   code-after .value_required:n = true ,
213 }

```

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

```

214 \keys_define:nn { NiceMatrix }
215 {
216   NiceMatrixOptions .inherit:n =
217     {
218       NiceMatrix / Global ,
219       NiceMatrix / Code

```

¹⁹In fact, we should not because, if there is a `w` node, the `w` node is used instead of the “medium” node.

```

220     } ,
221     NiceMatrix .inherit:n =
222     {
223         NiceMatrix / Global ,
224         NiceMatrix / Env
225     } ,
226     NiceArray .inherit:n =
227     {
228         NiceMatrix / Global ,
229         NiceMatrix / Env ,
230         NiceMatrix / Code
231     } ,
232     pNiceArray .inherit:n =
233     {
234         NiceMatrix / Global ,
235         NiceMatrix / Env ,
236         NiceMatrix / Code
237     }
238 }

```

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

```

239 \keys_define:nn { NiceMatrix / NiceMatrixOptions }
240 {

```

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

```

241     renew-matrix .code:n = \__nm_renew_matrix: ,
242     renew-matrix .value_forbidden:n = true ,
243     RenewMatrix .code:n = \__nm_error:n { Option-RenewMatrix-suppressed }
244         \__nm_renew_matrix: ,
245     transparent .meta:n = { renew-dots , renew-matrix } ,
246     transparent .value_forbidden:n = true ,
247     Transparent .code:n = \__nm_error:n { Option-Transparent-suppressed }
248         \__nm_renew_matrix:
249         \bool_set_true:N \l__nm_renew_dots_bool ,

```

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

```

250     exterior-arraycolsep .bool_set:N = \l__nm_exterior_arraycolsep_bool ,
251     exterior-arraycolsep .default:n = true ,

```

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

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

```

252     columns-width .code:n =
253         \str_if_eq:nnTF { #1 } { auto }
254         { \__nm_error:n { Option-auto-for-columns-width } }
255         { \dim_set:Nn \l__nm_columns_width_dim { #1 } } ,

```

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

```

256     allow-duplicate-names .code:n =
257         \__nm_msg_redirect_name:nn { Duplicate-name } { none } ,
258     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`).

```

259     letter-for-dotted-lines .code:n =

```

```

260     {
261       \int_compare:nTF { \tl_count:n { #1 } = \c_one_int }
262       { \tl_set:Nx \l_nm_letter_for_dotted_lines_str { #1 } }
263       { \__nm_error:n { Bad-value-for-letter-for-dotted-lines } }
264     } ,
265     letter-for-dotted-lines .value_required:n = true ,
266     letter-for-dotted-lines .initial:n = \c_colon_str ,

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

268 \__nm_msg_new:nn { Bad-value-for-letter-for-dotted-lines }
269 {
270   The~value-of~key~'\tl_use:N\l_keys_key_tl'~must-be-of~length~1.\\
271   If~you-go-on,~it~will-be-ignored.
272 }

273 \__nm_msg_new:nnn { Unknown-key-for-NiceMatrixOptions }
274 {
275   The~key~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~command~
276   \token_to_str:N \NiceMatrixOptions. \\
277   If~you-go-on,~it~will-be-ignored. \\
278   For~a~list~of~the~available~keys,~type~H~<return>.
279 }
280 {
281   The~available~keys~are~(in~alphabetic~order):~
282   allow-duplicate-names,~
283   code-for-first-col,~
284   code-for-first-row,~
285   code-for-last-col,~
286   code-for-last-row,~
287   exterior-arraycolsep,~
288   hlines,~
289   left-margin,~
290   letter-for-dotted-lines,~
291   nullify-dots,~
292   parallelize-diags,~
293   renew-dots,~
294   renew-matrix,~
295   right-margin,~
296   and~transparent
297 }

298 \__nm_msg_new:nn { Option-auto-for-columns-width }
299 {
300   You~can't~give~the~value~'auto'~to~the~option~'columns-width'~here.~
301   If~you-go-on,~the~option~will~be~ignored.
302 }

```

`\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.

```

303 \NewDocumentCommand \NiceMatrixOptions { m }
304 { \keys_set:nn { NiceMatrix / NiceMatrixOptions } { #1 } }

```

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

```

305 \keys_define:nn { NiceMatrix / NiceMatrix }
306 { unknown .code:n = \__nm_error:n { Unknown-option-for-NiceMatrix } }

307 \__nm_msg_new:nnn { Unknown-option-for-NiceMatrix }
308 {
309   The~option~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~environment~
310   \{NiceMatrix\}~and~its~variants. \\

```

```

311   If~you~go~on,~it~will~be~ignored. \\
312   For~a~list~of~the~available~options,~type~H~<return>.
313 }
314 {
315   The~available~options~are~(in~alphabetic~order):~
316   code~after,~
317   columns~width,~
318   create~extra~nodes,~
319   extra~left~margin,~
320   extra~right~margin,~
321   hlines,~
322   left~margin,~
323   name,~
324   nullify~dots,~
325   parallelize~diags,~
326   renew~dots~
327   and~right~margin.
328 }

329 \__nm_msg_new:nnn { Duplicate~name }
330 {
331   The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
332   the~same~environment~name~twice.~You~can~go~on,~but,~
333   maybe,~you~will~have~incorrect~results~especially~
334   if~you~use~'columns~width=auto'. \\
335   For~a~list~of~the~names~already~used,~type~H~<return>. \\
336   If~you~don't~want~to~see~this~message~again,~use~the~option~
337   'allow~duplicate~names'.
338 }
339 {
340   The~names~already~defined~in~this~document~are:~
341   \seq~use:Nnnn \g__nm_names_seq { ,~ } { ,~ } { ~and~ }.
342 }

```

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

```

343 \keys_define:nn { NiceMatrix / NiceArray }
344 {

```

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

```

345   c .code:n = \str_set:Nn \l__nm_pos_env_str c ,
346   t .code:n = \str_set:Nn \l__nm_pos_env_str t ,
347   b .code:n = \str_set:Nn \l__nm_pos_env_str b ,
348   first~col .code:n = \int_zero:N \l__nm_first_col_int ,
349   last~col .bool_set:N = \l__nm_last_col_bool ,
350   first~row .code:n = \int_zero:N \l__nm_first_row_int ,
351   last~row .int_set:N = \l__nm_last_row_int ,
352   last~row .default:n = -1 ,
353   unknown .code:n = \__nm_error:n { Unknown~option~for~NiceArray }
354 }

355 \__nm_msg_new:nnn { Unknown~option~for~NiceArray }
356 {
357   The~option~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~environment~
358   \{NiceArray\}. \\
359   If~you~go~on,~it~will~be~ignored. \\
360   For~a~list~of~the~available~options,~type~H~<return>.
361 }
362 {
363   The~available~options~are~(in~alphabetic~order):~
364   b,~

```

```

365 c,~
366 code-after,~
367 code-for-first-col,~
368 code-for-first-row,~
369 code-for-last-col,~
370 code-for-last-row,~
371 columns-width,~
372 create-extra-nodes,~
373 extra-left-margin,~
374 extra-right-margin,~
375 hlines,~
376 left-margin,~
377 name,~
378 nullify-dots,~
379 parallelize-diags,~
380 renew-dots,~
381 right-margin,~
382 and~t.
383 }

384 \keys_define:nn { NiceMatrix / pNiceArray }
385 {
386   first-col .code:n = \int_zero:N \l__nm_first_col_int ,
387   last-col .bool_set:N = \l__nm_last_col_bool ,
388   first-row .code:n = \int_zero:N \l__nm_first_row_int ,
389   last-row .int_set:N = \l__nm_last_row_int ,
390   last-row .default:n = -1 ,
391   unknown .code:n = \__nm_error:n { Unknown~option~for~pNiceArray }
392 }

393 \__nm_msg_new:nnn { Unknown~option~for~pNiceArray }
394 {
395   The~option~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~environment~
396   \{\@currenvir\}. \\
397   If~you~go~on,~it~will~be~ignored. \\
398   For~a~list~of~the~available~options,~type~H~<return>.
399 }
400 {
401   The~available~options~are~(in~alphabetic~order):~
402   code-after,~
403   code-for-first-col,~
404   code-for-first-row,~
405   code-for-last-col,~
406   code-for-last-row,~
407   columns-width,~
408   create-extra-nodes,~
409   extra-left-margin,~
410   extra-right-margin,~
411   first-col,~
412   first-row,~
413   last-col,~
414   last-row,~
415   hlines,~
416   left-margin,~
417   name,~
418   nullify-dots,~
419   parallelize-diags,~
420   renew-dots,~
421   and~right-margin.
422 }

```

13.4 Code common to {NiceArrayWithDelims} and {NiceMatrix}

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}`).

```
423 \cs_new_protected:Nn \__nm_Cell:
424 {
```

We increment `\g_@@_col_int`, which is the counter of the columns.

```
425 \int_gincr:N \g__nm_col_int
```

Now, we increment the counter of the rows. We don't do this incrementation in the `\everycr` because some packages, like `arydshln`, create special rows in the `\halign` that we don't want to take into account.

```
426 \int_compare:nNnT \g__nm_col_int = \c_one_int
427 {
428   \int_compare:nNnT \l__nm_first_col_int = \c_one_int
429   \__nm_begin_of_row:
430 }
431 \int_gset:Nn \g__nm_col_total_int
432 { \int_max:nn \g__nm_col_total_int \g__nm_col_int }
```

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).

```
433 \hbox_set:Nw \l_tmpa_box
434 \c_math_toggle_token
435 \int_compare:nNnTF \g__nm_row_int = \c_zero_int
436 \l__nm_code_for_first_row_tl
437 {
438   \int_compare:nNnT \g__nm_row_int = \l__nm_last_row_int
439   \l__nm_code_for_last_row_tl
440 }
441 }
```

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.

```
442 \cs_new_protected:Nn \__nm_begin_of_row:
443 {
444   \int_gincr:N \g__nm_row_int
445   \dim_gset_eq:NN \g__nm_dp_ante_last_row_dim \g__nm_dp_last_row_dim
446   \dim_gzero:N \g__nm_dp_last_row_dim
447   \dim_gzero:N \g__nm_ht_last_row_dim
448 }
```

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.

```
449 \cs_new_protected:Npn \__nm_actualization_for_first_and_last_row:
450 {
451   \int_compare:nNnT \g__nm_row_int = \c_zero_int
452   {
453     \dim_gset:Nn \g__nm_dp_row_zero_dim
454     { \dim_max:nn \g__nm_dp_row_zero_dim { \box_dp:N \l_tmpa_box } }
455     \dim_gset:Nn \g__nm_ht_row_zero_dim
456     { \dim_max:nn \g__nm_ht_row_zero_dim { \box_ht:N \l_tmpa_box } }
457   }
458   \int_compare:nNnT \g__nm_row_int = \c_one_int
459   {
460     \dim_gset:Nn \g__nm_ht_row_one_dim
461     { \dim_max:nn \g__nm_ht_row_one_dim { \box_ht:N \l_tmpa_box } }
462   }
463   \dim_gset:Nn \g__nm_ht_last_row_dim
464   { \dim_max:nn \g__nm_ht_last_row_dim { \box_ht:N \l_tmpa_box } }
465   \dim_gset:Nn \g__nm_dp_last_row_dim
466   { \dim_max:nn \g__nm_dp_last_row_dim { \box_dp:N \l_tmpa_box } }
467 }
```

```

468 \cs_new_protected:Nn \__nm_end_Cell:
469 {
470   \c_math_toggle_token
471   \hbox_set_end:

```

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

```

472 \dim_gset:Nn \g__nm_max_cell_width_dim
473 { \dim_max:nn \g__nm_max_cell_width_dim { \box_wd:N \l_tmpa_box } }

```

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

```

474 \__nm_actualization_for_first_and_last_row:

```

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

```

475 \tikz
476 [
477   remember~picture ,
478   inner~sep = \c_zero_dim ,
479   minimum~width = \c_zero_dim ,
480   baseline
481 ]
482 \node
483 [
484   anchor = base ,
485   name = nm - \int_use:N \g__nm_env_int -
486           \int_use:N \g__nm_row_int -
487           \int_use:N \g__nm_col_int ,
488   alias =
489     \str_if_empty:NF \l__nm_name_str
490     {
491       \l__nm_name_str -
492       \int_use:N \g__nm_row_int -
493       \int_use:N \g__nm_col_int
494     }
495 ]
496 \bgroup
497 \box_use:N \l_tmpa_box
498 \egroup ;
499 }
500 \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.

```

501 \cs_new_protected:Nn \__nm_renewcolumnntype:nn
502 {
503   \newcolumnntype #1 [ 2 ]
504   {
505     > {
506       \hbox_set:Nw \l_tmpa_box
507       \__nm_Cell:
508     }
509     c
510     < {
511       \__nm_end_Cell:
512       \hbox_set_end:
513       #2
514       \hbox_set:Nn \l_tmpb_box
515       { \makebox [ ##2 ] [ ##1 ] { \box_use:N \l_tmpa_box } }
516       \dim_set:Nn \l_tmpa_dim { \box_dp:N \l_tmpb_box }

```

```

517 \box_move_down:nn \l_tmpa_dim
518 {
519   \vbox:n
520   {
521     \hbox_to_wd:nn { \box_wd:N \l_tmpb_box }
522     {
523       \hfil
524       \tikz [ remember-picture , overlay ]
525       \coordinate ( __nm~north~east ) ;
526     }
527     \hbox:n
528     {
529       \tikz [ remember-picture , overlay ]
530       \coordinate ( __nm~south~west ) ;
531       \box_move_up:nn \l_tmpa_dim { \box_use:N \l_tmpb_box }
532     }
533   }
534 }

```

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

```

535 \tikz [ remember-picture , overlay ]
536 \node
537 [
538   node~contents = { } ,
539   name = nm - \int_use:N \g__nm_env_int -
540           \int_use:N \g__nm_row_int -
541           \int_use:N \g__nm_col_int - w,
542   alias =
543     \str_if_empty:NF \l__nm_name_str
544     {
545       \l__nm_name_str -
546       \int_use:N \g__nm_row_int -
547       \int_use:N \g__nm_col_int - w
548     } ,
549   inner~sep = \c_zero_dim ,
550   fit = ( __nm~south~west ) ( __nm~north~east )
551 ]
552 ;
553 }
554 }
555 }

```

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 `\g_@@_lines_to_draw_tl` the instruction that 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 & \Hdotsfor{2} & & 
\end{pNiceMatrix}

```

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

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

```

\@@_draw_Cdots:nn {2}{2}
\@@_draw_Hdotsfor:nnn {3}{2}{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.

```

556 \bool_if:NTF \c__nm_draft_bool
557 { \cs_set_protected:Npn \__nm_instruction_of_type:n #1 { } }

```

```

558 {
559   \cs_new_protected:Npn \__nm_instruction_of_type:n #1
560   {
561     \tl_gput_right:Nx \g__nm_lines_to_draw_tl
562     {
563       \use:c { __nm _ draw _ #1 : nn }
564       { \int_use:N \g__nm_row_int }
565       { \int_use:N \g__nm_col_int }
566     }
567   }
568 }

```

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`.

```

569 \cs_new_protected:Npn \__nm_array:
570 {
571   \bool_if:NTF \c__nm_revtex_bool
572   {
573     \cs_set_eq:NN \@acoll \@arrayacol
574     \cs_set_eq:NN \@acolr \@arrayacol
575     \cs_set_eq:NN \@acol \@arrayacol
576     \cs_set:Npn \@halignto { }
577     \@array@array
578   }
579   \array

```

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

```

580 [ \l__nm_pos_env_str ]
581 }

```

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

```

582 \cs_new:Npn \__nm_everycr:
583 { \noalign { \__nm_everycr_i: } }
584 \cs_new_protected:Npn \__nm_everycr_i:
585 {
586   \int_gzero:N \g__nm_col_int
587   \bool_if:NT \l__nm_hlines_bool
588   {

```

The counter `\g_@@_row_int` 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.

```

589     \int_compare:nNnT \g__nm_row_int > { -1 }
590     {
591       \int_compare:nNnF \g__nm_row_int = \g__nm_last_row_int
592       {
593         \hrule \@height \arrayrulewidth
594         \skip_vertical:n { - \arrayrulewidth }
595       }
596     }
597   }
598 }

```

The following code `\@@_pre_array:` is used in `{NiceArray}` and in `{NiceMatrix}`. It contains code that will be executed *before* the construction of the array.

```

599 \cs_new_protected:Npn \__nm_pre_array:
600 {

```

If the user requires all the columns to have a width equal to the widest cell of the array, we read this length in the file `.aux` (of, course, this is possible only on the second run of LaTeX: on the first run, the dimension `\l_@@_columns_width_dim` will be set to zero — and the columns will have their natural width). Remark that, even if the environment has a name (see just below) we have to write in the `aux` file the information with the number of environment because of `auto-columns-width` of `{NiceMatrixBlock}`.

```

601 \bool_if:NT \l__nm_auto_columns_width_bool
602 {
603   \group_insert_after:N \__nm_write_max_cell_width:
604   \cs_if_free:cTF { __nm_max_cell_width_ \int_use:N \g__nm_env_int }
605     { \dim_zero:N \l__nm_columns_width_dim }
606     {
607       \dim_set:Nx \l__nm_columns_width_dim
608       { \use:c { __nm_max_cell_width_ \int_use:N \g__nm_env_int } }
609     }

```

If the environment has a name, we read the value of the maximal value of the columns from `__@_name_cell_widthname` (the value will be the correct value even if the number of the environment has changed (for example because the user has created or deleted an environment before the current one)).

```

610 \str_if_empty:NF \l__nm_name_str
611 {
612   \cs_if_free:cF { __nm_max_cell_width_ \l__nm_name_str }
613   {
614     \dim_set:Nx \l__nm_columns_width_dim
615     { \use:c { __nm_max_cell_width_ \l__nm_name_str } }
616   }
617 }
618 }

```

We don't want to patch any code and that's why some code is executed in a `\group_insert_after:N`. In particular, in this `\group_insert_after:N`, we will have to know the value of some parameters like `\l_@@_extra_nodes_bool`. That's why we transit via a global version for some variables.

```

619 \bool_gset_eq:NN \g__nm_extra_nodes_bool \l__nm_extra_nodes_bool
620 \dim_gset_eq:NN \g__nm_left_margin_dim \l__nm_left_margin_dim
621 \dim_gset_eq:NN \g__nm_right_margin_dim \l__nm_right_margin_dim
622 \dim_gset_eq:NN \g__nm_extra_right_margin_dim \l__nm_extra_right_margin_dim
623 \int_gset_eq:NN \g__nm_last_row_int \l__nm_last_row_int
624 \tl_gset_eq:NN \g__nm_name_str \l__nm_name_str
625 \int_gset_eq:NN \g__nm_first_row_int \l__nm_first_row_int
626 \int_gset_eq:NN \g__nm_first_col_int \l__nm_first_col_int
627 \bool_gset_eq:NN \g__nm_NiceArray_bool \l__nm_NiceArray_bool
628 \bool_gset_eq:NN \g__nm_last_row_without_value_bool
629 \l__nm_last_row_without_value_bool

```

The environment `{array}` uses internally the command `\ialign` and, in particular, this command `\ialign` sets `\everycr` to `{}`. However, we want to use `\everycr` in our array. The solution is to give to `\ialign` a new definition (giving to `\everycr` the value we want) that will revert automatically to its default definition after the first utilisation.²⁰

```

630 \cs_set:Npn \ialign
631 {
632   \bool_if:NTF \c__nm_colortbl_loaded_bool
633   {
634     \CT@everycr
635     {
636       \noalign { \global\let\CT@row@color\relax }
637       \__nm_everycr:
638     }
639   }
640   { \everycr { \__nm_everycr: } }

```

²⁰With this programming, we will have, in the cells of the array, a clean version of `\ialign`. That's necessary: the user will probably not employ directly `\ialign` in the array... but more likely environments that utilize `\ialign` internally (e.g.: `{substack}`)

```

641     \tabskip = \c_zero_skip
642     \cs_set:Npn \ialign
643     {
644         \everycr { }
645         \tabskip = \c_zero_skip
646         \halign
647         }
648     \halign
649 }

```

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}.

```

650 \dim_compare:nNnTF \l__nm_columns_width_dim = \c_zero_dim
651 {
652     \newcolumntype L { > \__nm_Cell: l < \__nm_end_Cell: }
653     \newcolumntype C { > \__nm_Cell: c < \__nm_end_Cell: }
654     \newcolumntype R { > \__nm_Cell: r < \__nm_end_Cell: }
655 }

```

If there is an option that specify that all the columns must have the same width, the column types L, C and R are in fact defined upon the column type w of array which is, in fact, redefined below.

```

656 {
657     \newcolumntype L { w l { \dim_use:N \l__nm_columns_width_dim } }
658     \newcolumntype C { w c { \dim_use:N \l__nm_columns_width_dim } }
659     \newcolumntype R { w r { \dim_use:N \l__nm_columns_width_dim } }
660 }
661 \cs_set_eq:NN \Ldots \__nm_Ldots
662 \cs_set_eq:NN \Cdots \__nm_Cdots
663 \cs_set_eq:NN \Vdots \__nm_Vdots
664 \cs_set_eq:NN \Ddots \__nm_Ddots
665 \cs_set_eq:NN \Iddots \__nm_Iddots
666 \cs_set_eq:NN \hdottedline \__nm_hdottedline:
667 \cs_set_eq:NN \Hspace \__nm_Hspace:
668 \cs_set_eq:NN \Hdotsfor \__nm_Hdotsfor:
669 \cs_set_eq:NN \multicolumn \__nm_multicolumn:nnn
670 \bool_if:NT \l__nm_renew_dots_bool
671 {
672     \cs_set_eq:NN \ldots \__nm_Ldots
673     \cs_set_eq:NN \cdots \__nm_Cdots
674     \cs_set_eq:NN \vdots \__nm_Vdots
675     \cs_set_eq:NN \ddots \__nm_Ddots
676     \cs_set_eq:NN \iddots \__nm_Iddots
677     \cs_set_eq:NN \dots \__nm_Ldots
678     \cs_set_eq:NN \hdotsfor \__nm_Hdotsfor:
679 }

```

The sequence `\g_@@_empty_cells_seq` will contain a list of “empty” cells (not all the empty cells of the matrix). If we want to indicate that the cell in row i and column j must be considered as empty, the token list “ $i-j$ ” will be put in this sequence.

```

680 \seq_gclear_new:N \g__nm_empty_cells_seq

```

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

```

681 \seq_gclear_new:N \g__nm_multicolumn_cells_seq
682 \seq_gclear_new:N \g__nm_multicolumn_sizes_seq

```

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

```

683 \int_gzero_new:N \g__nm_row_int
684 \int_gset:Nn \g__nm_row_int { \l__nm_first_row_int - 1 }

```

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

```
685 \int_gzero_new:N \g__nm_row_total_int
```

The counter `\g_@@_col_int` 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.

```
686 \int_gzero_new:N \g__nm_col_int
687 \int_gzero_new:N \g__nm_col_total_int
688 \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:`.

```
689 \cs_set_eq:NN \NC@find@w \relax
690 \cs_set_eq:NN \NC@find@W \relax
691 \__nm_renewcolumnntype:nn w { }
692 \__nm_renewcolumnntype:nn W { \cs_set_eq:NN \hss \hfil }
693 \dim_gzero_new:N \g__nm_dp_row_zero_dim
694 \dim_gzero_new:N \g__nm_ht_row_zero_dim
695 \dim_gzero_new:N \g__nm_ht_row_one_dim
696 \dim_gzero_new:N \g__nm_dp_ante_last_row_dim
697 \dim_gzero_new:N \g__nm_ht_last_row_dim
698 \dim_gzero_new:N \g__nm_dp_last_row_dim
```

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`.

```
699 \exp_args:Nx \newcolumnntype \l__nm_letter_for_dotted_lines_str
700 {
701   !
702   {
703     \skip_horizontal:n { 0.53 pt }
704     \bool_gset_true:N \g__nm_extra_nodes_bool
```

Consider the following code:

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

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

```
705 \int_compare:nNnT \g__nm_col_int > \g__nm_last_vdotted_col_int
706 {
707   \int_gset_eq:NN \g__nm_last_vdotted_col_int \g__nm_col_int
708   \tl_gput_right:Nx \g__nm_code_after_tl
```

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

```
709   { \__nm_vdottedline:n { \int_use:N \g__nm_col_int } }
710 }
711 }
712 }
```

```

713 \int_gzero_new:N \g__nm_last_vdotted_col_int
714 \bool_if:NT \c__nm_siunitx_loaded_bool \__nm_renew_NC@rewrite@S:
715 \int_gset:Nn \g__nm_last_vdotted_col_int { -1 }
716 \bool_gset_false:N \g__nm_last_col_found_bool
717 }

```

13.5 The environment {NiceArrayWithDelims}

```

718 \NewDocumentEnvironment { NiceArrayWithDelims } { m m O { } m ! O { } }
719 {
720   \__nm_adapt_S_column:
721   \__nm_test_if_math_mode:
722   \bool_if:NT \l__nm_in_env_bool { \__nm_fatal:n { Yet~in~env } }
723   \bool_set_true:N \l__nm_in_env_bool

```

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

```

724 \cs_if_exist:NT \tikz@library@external@loaded
725 { \tikzset { external / export = false } }

```

In {NiceArrayWithDelims}, it would have been possible to avoid the `\group_insert_after:N` and to put `\@@_after_array` in the second part of the environment {NiceArrayWithDelims}. However, it's not possible to do that in {NiceMatrix} (because of the option `renew-matrix`) and that's why we use this technique in {NiceArrayWithDelims} and in {NiceMatrix}.

```

726 \group_insert_after:N \__nm_after_array:
727 \tl_gclear_new:N \g__nm_lines_to_draw_tl

```

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

```

728 \int_gincr:N \g__nm_env_int
729 \bool_if:NF \l__nm_block_auto_columns_width_bool
730 { \dim_gzero_new:N \g__nm_max_cell_width_dim }

```

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

```

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

```

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

```

732 \bool_if:NTF \l__nm_NiceArray_bool
733 { \keys_set:nn { NiceMatrix / NiceArray } }
734 { \keys_set:nn { NiceMatrix / pNiceArray } }
735 { #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).

```

736 \int_compare:nNnT \l__nm_last_row_int = { -1 }
737 {
738   \bool_set_true:N \l__nm_last_row_without_value_bool

```

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

```

739 \str_if_empty:NTF \g__nm_name_str
740 {
741   \cs_if_exist:cT { __nm_last_row_ \int_use:N \g__nm_env_int }
742   {
743     \int_set:Nn \l__nm_last_row_int
744     { \use:c { __nm_last_row_ \int_use:N \g__nm_env_int } }
745   }
746 }
747 {
748   \cs_if_exist:cT { __nm_last_row_ \g__nm_name_str }
749   {
750     \int_set:Nn \l__nm_last_row_int
751     { \use:c { __nm_last_row_ \g__nm_name_str } }
752   }

```



```

753     }
754 }

```

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

```

755 \_nm_pre_array:

```

We compute the width of the two delimiters.

```

756 \dim_zero_new:N \g__nm_left_delim_dim
757 \dim_zero_new:N \g__nm_right_delim_dim
758 \bool_if:NTF \l__nm_NiceArray_bool
759 {
760   \dim_gset:Nn \g__nm_left_delim_dim { 2 \arraycolsep }
761   \dim_gset:Nn \g__nm_right_delim_dim { 2 \arraycolsep }
762 }
763 {
764   \group_begin:
765   \dim_set_eq:NN \nulldelimiterspace \c_zero_dim
766   \hbox_set:Nn \l_tmpa_box
767   {
768     \c_math_toggle_token
769     \left #1 \vcenter to 1 cm { } \right.
770     \c_math_toggle_token
771   }
772   \dim_gset:Nn \g__nm_left_delim_dim { \box_wd:N \l_tmpa_box }
773   \hbox_set:Nn \l_tmpa_box
774   {
775     \dim_set_eq:NN \nulldelimiterspace \c_zero_dim
776     \c_math_toggle_token
777     \left. \vcenter to 1 cm { } \right #2
778     \c_math_toggle_token
779   }
780   \dim_gset:Nn \g__nm_right_delim_dim { \box_wd:N \l_tmpa_box }
781   \group_end:
782 }
783 }

```

The array will be composed in a box (named `\l_@@_the_array_box`) because we have to do manipulations concerning the potential exterior rows (such construction in a box is not possible for `{NiceMatrix}`).

```

784 \box_clear_new:N \l__nm_the_array_box

```

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

```

785 \tl_set:Nn \l_tmpa_tl { #4 }
786 \int_compare:nNnTF \l__nm_first_col_int = \c_zero_int
787 { \tl_put_left:NV \l_tmpa_tl \c__nm_preamble_first_col_tl }
788 {
789   \bool_if:NT \l__nm_NiceArray_bool
790   {
791     \bool_if:NF \l__nm_exterior_arraycolsep_bool
792     { \tl_put_left:Nn \l_tmpa_tl { @ { } } }
793   }
794 }
795 \bool_if:NTF \l__nm_last_col_bool
796 { \tl_put_right:NV \l_tmpa_tl \c__nm_preamble_last_col_tl }
797 {
798   \bool_if:NT \l__nm_NiceArray_bool
799   {
800     \bool_if:NF \l__nm_exterior_arraycolsep_bool
801     { \tl_put_right:Nn \l_tmpa_tl { @ { } } }
802   }
803 }

```

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).

```

804 \hbox_set:Nw \l__nm_the_array_box
805 \skip_horizontal:n \l__nm_left_margin_dim
806 \skip_horizontal:n \l__nm_extra_left_margin_dim
807 \c_math_toggle_token

```

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

```

808 \exp_args:NV \__nm_array: \l_tmpa_tl
809 }

```

We begin the second part of the environment `{NiceArrayWithDelims}`.

```

810 {
811 \endarray
812 \c_math_toggle_token
813 \skip_horizontal:n \g__nm_right_margin_dim
814 \skip_horizontal:n \g__nm_extra_right_margin_dim
815 \hbox_set_end:

```

Now, we compute `\l_tmpa_dim` which is the vertical dimension of the “first row” above the array (when the key `first-row` is used).

```

816 \int_compare:nNnTF \l__nm_first_row_int = \c_zero_int
817 {
818 \dim_compare:nNnTF
819 { \g__nm_ht_row_one_dim + \g__nm_dp_row_zero_dim + \lineskiplimit }
820 > \baselineskip
821 {
822 \dim_set:Nn \l_tmpa_dim
823 {
824 \g__nm_ht_row_one_dim + \g__nm_dp_row_zero_dim + \lineskip
825 + \g__nm_ht_row_zero_dim - \g__nm_ht_row_one_dim
826 }
827 }
828 {
829 \dim_set:Nn \l_tmpa_dim
830 { \baselineskip + \g__nm_ht_row_zero_dim - \g__nm_ht_row_one_dim }
831 }
832 }
833 { \dim_zero:N \l_tmpa_dim }

```

We compute `\l_tmpb_dim` which is the vertical dimension of the “last row” below the array (when the key `last-row` is used). A value of `-2` for `\l_@@_last_row_int` means that there is no “last row”.²¹

```

834 \int_compare:nNnTF \l__nm_last_row_int > { -2 }
835 {
836 \dim_compare:nNnTF
837 {
838 \g__nm_ht_last_row_dim + \g__nm_dp_ante_last_row_dim + \lineskiplimit
839 }
840 >
841 \baselineskip
842 {
843 \dim_set:Nn \l_tmpb_dim
844 {
845 \g__nm_ht_last_row_dim + \g__nm_dp_ante_last_row_dim + \lineskip
846 + \g__nm_dp_last_row_dim - \g__nm_dp_ante_last_row_dim
847 }
848 }
849 {
850 \dim_set:Nn \l_tmpb_dim
851 {
852 \baselineskip
853 + \g__nm_dp_last_row_dim - \g__nm_dp_ante_last_row_dim

```

²¹A 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`).

```

854     }
855   }
856 }
857 { \dim_zero:N \l_tmpb_dim }

```

Now, we begin the real construction in the output flow of TeX. First, we take into account a potential “first column” (we remind that this “first column” has been constructed in an overlapping position and that we have computed its width in `\g_@@_width_first_col_dim`: see p. 44).

```

858 \int_compare:nNnT \g__nm_first_col_int = \c_zero_int
859 {
860   \skip_horizontal:n \arraycolsep
861   \skip_horizontal:n \g__nm_width_first_col_dim
862 }

```

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}`.

```

863 \bool_if:NTF \g__nm_NiceArray_bool
864 {
865   \int_compare:nNnT \g__nm_first_row_int = \c_zero_int
866   {
867     \str_if_eq:VnTF \l__nm_pos_env_str { t }
868     {
869       \box_move_up:nn
870       { \l_tmpa_dim - \g__nm_ht_row_zero_dim + \g__nm_ht_row_one_dim }
871     }
872   }
873   {
874     \bool_if:NT \g__nm_last_row_int
875     {
876       \str_if_eq:VnT \l__nm_pos_env_str { b }
877       {
878         \box_move_down:nn
879         {
880           \l_tmpb_dim
881           - \g__nm_dp_last_row_dim + \g__nm_dp_ante_last_row_dim
882         }
883       }
884     }
885   }
886   { \box_use_drop:N \l__nm_the_array_box }
887 }

```

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

```

888 {
889   \hbox_set:Nn \l_tmpa_box
890   {
891     \c_math_toggle_token
892     \left #1
893     \vcenter
894     {

```

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

```

895       \skip_vertical:n { - \l_tmpa_dim }
896       \hbox:n
897       {
898         \skip_horizontal:n { - \arraycolsep }
899         \box_use_drop:N \l__nm_the_array_box
900         \skip_horizontal:n { - \arraycolsep }
901       }

```

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

```

902       \skip_vertical:n { - \l_tmpb_dim }
903     }
904     \right #2

```

```

905         \c_math_toggle_token
906     }
907     \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }
908     \box_set_dp:Nn \l_tmpa_box { \box_dp:N \l_tmpa_box + \l_tmpb_dim }
909     \box_use_drop:N \l_tmpa_box
910 }

```

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. 45).

```

911     \bool_if:NT \g__nm_last_col_found_bool
912     {
913         \skip_horizontal:n \g__nm_width_last_col_dim
914         \skip_horizontal:n \arraycolsep
915     }
916 }

```

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

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

```

917 \tl_const:Nn \c__nm_preamble_first_col_tl
918 {
919     >
920     {
921         \__nm_begin_of_row:

```

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

```

922         \hbox_set:Nw \l_tmpa_box
923         \c_math_toggle_token
924         \l__nm_code_for_first_col_tl
925     }
926     l
927     <
928     {
929         \c_math_toggle_token
930         \hbox_set_end:
931         \__nm_actualization_for_first_and_last_row:

```

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

```

932     \dim_gset:Nn \g__nm_width_first_col_dim
933     {
934         \dim_max:nn
935         \g__nm_width_first_col_dim
936         { \box_wd:N \l_tmpa_box }
937     }

```

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

```

938     \hbox_overlap_left:n
939     {
940         \tikz
941         [
942             remember~picture ,
943             inner~sep = \c_zero_dim ,
944             minimum~width = \c_zero_dim ,
945             baseline
946         ]
947         \node
948         [
949             anchor = base ,
950             name =
951                 nm -
952                 \int_use:N \g__nm_env_int -
953                 \int_use:N \g__nm_row_int -
954                 0 ,
955             alias =

```

```

956         \str_if_empty:NF \l__nm_name_str
957         {
958             \l__nm_name_str -
959             \int_use:N \g__nm_row_int -
960             0
961         }
962     ]
963     { \box_use:N \l_tmpa_box } ;
964     \skip_horizontal:n
965     {
966         \g__nm_left_delim_dim +
967         \l__nm_left_margin_dim +
968         \l__nm_extra_left_margin_dim
969     }
970 }
971 \skip_horizontal:n { - 2 \arraycolsep }
972 }
973 }

```

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

```

974 \tl_const:Nn \c__nm_preamble_last_col_tl
975 {
976     >
977     {

```

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

```

978         \bool_gset_true:N \g__nm_last_col_found_bool
979         \int_gincr:N \g__nm_col_int
980         \int_gset:Nn \g__nm_col_total_int
981         { \int_max:nn \g__nm_col_total_int \g__nm_col_int }

```

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

```

982         \hbox_set:Nw \l_tmpa_box
983         \c_math_toggle_token
984         \l__nm_code_for_last_col_tl
985     }
986     1
987     <
988     {
989         \c_math_toggle_token
990         \hbox_set_end:
991         \__nm_actualization_for_first_and_last_row:

```

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

```

992         \dim_gset:Nn \g__nm_width_last_col_dim
993         {
994             \dim_max:nn
995             \g__nm_width_last_col_dim
996             { \box_wd:N \l_tmpa_box }
997         }
998         \skip_horizontal:n { - 2 \arraycolsep }

```

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

```

999         \hbox_overlap_right:n
1000         {
1001             \skip_horizontal:n
1002             {
1003                 \g__nm_right_delim_dim +
1004                 \l__nm_right_margin_dim +
1005                 \l__nm_extra_right_margin_dim
1006             }
1007             \tikz
1008             [
1009                 remember~picture ,
1010                 inner~sep = \c_zero_dim ,

```

```

1011         minimum-width = \c_zero_dim ,
1012         baseline
1013     ]
1014 \node
1015 [
1016     anchor = base ,
1017     name =
1018         nm -
1019         \int_use:N \g__nm_env_int -
1020         \int_use:N \g__nm_row_int -
1021         \int_use:N \g__nm_col_int ,
1022     alias =
1023         \str_if_empty:NF \l__nm_name_str
1024         {
1025             \l__nm_name_str -
1026             \int_use:N \g__nm_row_int -
1027             \int_use:N \g__nm_col_int
1028         }
1029 ]
1030 { \box_use:N \l_tmpa_box } ;
1031 }
1032 }
1033 }

```

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.

```

1034 \NewDocumentEnvironment { NiceArray } { }
1035 {
1036     \bool_set_true:N \l__nm_NiceArray_bool

```

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).

```

1037     \NiceArrayWithDelims . .
1038 }
1039 { \endNiceArrayWithDelims }

```

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

```

1040 \NewDocumentEnvironment { pNiceArray } { }
1041 {
1042     \__nm_test_if_math_mode:
1043     \NiceArrayWithDelims ( )
1044 }
1045 { \endNiceArrayWithDelims }
1046 \NewDocumentEnvironment { bNiceArray } { }
1047 {
1048     \__nm_test_if_math_mode:
1049     \NiceArrayWithDelims [ ]
1050 }
1051 { \endNiceArrayWithDelims }
1052 \NewDocumentEnvironment { BNiceArray } { }
1053 {
1054     \__nm_test_if_math_mode:
1055     \NiceArrayWithDelims \{ \}
1056 }
1057 { \endNiceArrayWithDelims }
1058 \NewDocumentEnvironment { vNiceArray } { }
1059 {
1060     \__nm_test_if_math_mode:

```

```

1061 \NiceArrayWithDelims | |
1062 }
1063 { \endNiceArrayWithDelims }

1064 \NewDocumentEnvironment { VNiceArray } { }
1065 {
1066 \__nm_test_if_math_mode:
1067 \NiceArrayWithDelims \| \|
1068 }
1069 { \endNiceArrayWithDelims }

```

13.6 The environment `{NiceMatrix}` and its variants

Our environment `{NiceMatrix}` must have the same second part as the environment `{matrix}` of `amsmath` (because of the programming of the option `renew-matrix`). Hence, this second part is the following:

```

\endarray
\skip_horizontal:n { - \arraycolsep }

```

That's why in the definition of `{NiceMatrix}`, we have to use `\array` and not `\begin{array}`. This command `\array` is in `\@@_array:` (we have written this command because of the redefinition of `\array` done in `revtex4-1` and `revtex4-2`).

In order to execute code after the array, we use a command `\group_insert_after:N`.

Here's the definition of `{NiceMatrix}`:

```

1070 \NewDocumentEnvironment { NiceMatrix } { ! 0 { } }
1071 {
1072 \__nm_test_if_math_mode:
1073 \bool_if:NT \l__nm_in_env_bool { \__nm_fatal:n { Yet-in-env } }
1074 \bool_set_true:N \l__nm_in_env_bool

```

We have to deactivate the potential externalisation of Tikz because `nicematrix` uses Tikz with `remember picture`.

```

1075 \cs_if_exist:NT \tikz@library@external@loaded
1076 { \tikzset { external / export = false } }

```

The instruction for actual drawing of the dotted lines must be in a `\group_insert_after:N` because the second part of the environment must be the same as in `{array}` (for the option `renew-matrix`).

```

1077 \group_insert_after:N \__nm_after_array:
1078 \tl_gclear_new:N \g__nm_lines_to_draw_tl
1079 \int_gincr:N \g__nm_env_int
1080 \bool_if:NF \l__nm_block_auto_columns_width_bool
1081 { \dim_gzero_new:N \g__nm_max_cell_width_dim }
1082 \keys_set:nn { NiceMatrix / NiceMatrix } { #1 }

```

The macro `\@@_pre_array:` is defined above (see p. 36). It is also used in `{NiceArrayWithDelims}`.

```

1083 \__nm_pre_array:
1084 \skip_horizontal:n { - \arraycolsep }
1085 \skip_horizontal:n \l__nm_left_margin_dim
1086 \skip_horizontal:n \l__nm_extra_left_margin_dim
1087 \str_set:Nn \l__nm_pos_env_str c
1088 \bool_set_false:N \l__nm_exterior_arraycolsep_bool
1089 \__nm_array: { * \c@MaxMatrixCols C }
1090 }
1091 {
1092 \endarray
1093 \skip_horizontal:n { - \arraycolsep }

```

The two following lines are, of course, not in the second part of `{array}`, but that doesn't matter because, when `renew-matrix` is used, `\g_@@_right_margin_dim` and `\g_@@_extra_right_margin_dim` will be equal to 0 pt.

```

1094 \skip_horizontal:n \g__nm_right_margin_dim
1095 \skip_horizontal:n \g__nm_extra_right_margin_dim
1096 }

```

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

```

1097 \NewDocumentEnvironment { pNiceMatrix } { }
1098 {
1099   \_nm_test_if_math_mode:
1100   \left( \begin{NiceMatrix}
1101   }
1102   { \end{NiceMatrix} \right) }

1103 \NewDocumentEnvironment { bNiceMatrix } { }
1104 {
1105   \_nm_test_if_math_mode:
1106   \left[ \begin{NiceMatrix}
1107   }
1108   { \end{NiceMatrix} \right] }

1109 \NewDocumentEnvironment { BNiceMatrix } { }
1110 {
1111   \_nm_test_if_math_mode:
1112   \left\{ \begin{NiceMatrix}
1113   }
1114   { \end{NiceMatrix} \right\} }

1115 \NewDocumentEnvironment { vNiceMatrix } { }
1116 {
1117   \_nm_test_if_math_mode:
1118   \left\lvert \begin{NiceMatrix}
1119   }
1120   { \end{NiceMatrix} \right\rvert }

1121 \NewDocumentEnvironment { VNiceMatrix } { }
1122 {
1123   \_nm_test_if_math_mode:
1124   \left\lvert \begin{NiceMatrix}
1125   }
1126   { \end{NiceMatrix} \right\rvert }

```

13.7 Automatic width of the cells

For the option `columns-width=auto` (or the option `auto-columns-width` of the environment `{NiceMatrixBlock}`), we want to know the maximal width of the cells of the array (except the cells of the “exterior” column of an environment of the kind of `{pNiceAccayC}`). This length can be known only after the end of the construction of the array (or at the end of the environment `{NiceMatrixBlock}`). That’s why we store this value in the main `.aux` file and it will be available in the next run. We write a dedicated command for this because it will be called in a `\group_insert_after:N`.

```

1127 \cs_new_protected:Nn \_nm_write_max_cell_width:
1128 {
1129   \bool_if:NF \l__nm_block_auto_columns_width_bool
1130   {
1131     \iow_now:Nn \@mainaux \ExplSyntaxOn
1132     \iow_now:Nx \@mainaux
1133     {
1134       \cs_gset:cpn { __nm_max_cell_width_ \int_use:N \g__nm_env_int }
1135       { \dim_use:N \g__nm_max_cell_width_dim }
1136     }

```

If the environment has a name, we also create an alias named `\@@_max_cell_width_name`.

```

1137   \str_if_empty:NF \g__nm_name_str
1138   {
1139     \iow_now:Nx \@mainaux
1140     {
1141       \cs_gset:cpn { __nm_max_cell_width_ \g__nm_name_str }
1142       { \dim_use:N \g__nm_max_cell_width_dim }

```



```

1143     }
1144   }
1145   \iow_now:Nn \@mainaux \ExplSyntaxOff
1146 }
1147 }

```

13.8 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.

```

1148 \prg_set_conditional:Npnn \_nm_if_not_empty_cell:nn #1 #2 { T , TF }

```

If the cell is an implicit cell (that is after the symbol `\\` of end of row), the cell must, of course, be considered as empty. It’s easy to check whether we are in this situation considering the correspondent Tikz node.

```

1149 {
1150   \cs_if_free:cTF
1151   { pgf@sh@ns@nm -\int_use:N \g__nm_env_int - \int_use:N #1 - \int_use:N #2 }
1152   \prg_return_false:

```

We manage a list of “empty cells” called `\g__empty_cells_seq`. In fact, this list is not a list of all the empty cells of the array but only those explicitly declared empty for some reason. It’s easy to check if the current cell is in this list.

```

1153 {
1154   \seq_if_in:NxTF \g__nm_empty_cells_seq { \int_use:N #1 - \int_use:N #2 }
1155   \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.

```

1156 {
1157   \begin { pgfpicture }

```

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

```

1158   \tl_set:Nx \l_tmpa_tl
1159   { nm - \int_use:N \g__nm_env_int - \int_use:N #1 - \int_use:N #2 }
1160   \pgfpointanchor \l_tmpa_tl { east }
1161   \dim_gset:Nn \g_tmpa_dim \pgf@x
1162   \pgfpointanchor \l_tmpa_tl { west }
1163   \dim_gset:Nn \g_tmpb_dim \pgf@x
1164   \end { pgfpicture }
1165   \dim_compare:nNnTF
1166   { \dim_abs:n { \g_tmpb_dim - \g_tmpa_dim } } < { 0.5 pt }
1167   \prg_return_false:
1168   \prg_return_true:
1169 }
1170 }
1171 }

```

13.9 After the construction of the array

The macro `\@@_after_array:` is called (via `\group_insert_after:N`) in `{NiceArrayWithDelims}` and `{NiceMatrix}`.

```

1172 \cs_new_protected:Nn \_nm_after_array:
1173 {
1174   \int_compare:nNnTF \g__nm_row_int > \c_zero_int
1175   \_nm_after_array_i:
1176   { \_nm_error:n { Zero-row } }
1177 }

```

```

1178 \_nm_msg_new:n { Zero~row }
1179 {
1180   There~is~a~problem.~Maybe~your~environment~\{\@currenvir\}~is~empty.~
1181   Maybe~you~have~used~l,~c~and~r~instead~of~L,~C~and~R~in~the~preamble~
1182   of~your~environment. \\\
1183   If~you~go~on,~the~result~may~be~incorrect.
1184 }

```

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

```

1185 \cs_new_protected:Nn \_nm_after_array_i:
1186 {
1187   \group_begin:
1188   \cs_if_exist:NT \tikz@library@external@loaded
1189     { \tikzset { external / export = false } }

```

Now, the definition of `\g_@@_col_int` and `\g_@@_col_total_int` change: `\g_@@_col_int` will be the number of columns without the “last column”; `\g_@@_col_total_int` will be the number of columns with this “last column”.²²

```

1190 \int_gset_eq:NN \g__nm_col_int \g__nm_col_total_int
1191 \bool_if:nT \g__nm_last_col_found_bool { \int_gdecr:N \g__nm_col_int }

```

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

```

1192 \int_gset_eq:NN \g__nm_row_total_int \g__nm_row_int
1193 \int_compare:nNnT \g__nm_last_row_int > { -1 }
1194   { \int_gsub:Nn \g__nm_row_int \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.

```

1195 \bool_if:NT \g__nm_last_row_without_value_bool
1196 {
1197   \iow_now:Nn \@mainaux \ExplSyntaxOn
1198   \iow_now:Nx \@mainaux
1199   {
1200     \cs_gset:cpn { __nm_last_row_ \int_use:N \g__nm_env_int }
1201     { \int_use:N \g__nm_row_total_int }
1202   }

```

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.

```

1203   \str_if_empty:NF \g__nm_name_str
1204   {
1205     \iow_now:Nx \@mainaux
1206     {
1207       \cs_gset:cpn { __nm_last_row_ \g__nm_name_str }
1208       { \int_use:N \g__nm_row_total_int }
1209     }
1210   }
1211   \iow_now:Nn \@mainaux \ExplSyntaxOff
1212 }

```

The sequence `\g_@@_yet_drawn_seq` contains a list of lines which have been drawn previously in the matrix. We maintain this sequence because we don’t want to draw two overlapping lines.

```

1213 \seq_gclear_new:N \g__nm_yet_drawn_seq

```

By default, the diagonal lines will be parallelized²³. 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.

```

1214 \bool_if:NT \l__nm_parallelize_diags_bool
1215 {
1216   \int_zero_new:N \l__nm_ddots_int
1217   \int_zero_new:N \l__nm_iddots_int

```

²²We remind that the potential “first column” has the number 0.

²³It’s possible to use the option `parallelize-diags` to disable this parallelization.

```

1218         \dim_zero_new:N \l__nm_delta_x_one_dim
1219         \dim_zero_new:N \l__nm_delta_y_one_dim
1220         \dim_zero_new:N \l__nm_delta_x_two_dim
1221         \dim_zero_new:N \l__nm_delta_y_two_dim
1222     }

```

```
1223 \bool_if:NT \g_nm_extra_nodes_bool \_nm_create_extra_nodes:
```

```

1224 \tl_if_empty:NF \g_nm_lines_to_draw_tl
1225 {
1226   \int_zero_new:N \l_nm_initial_i_int
1227   \int_zero_new:N \l_nm_initial_j_int
1228   \int_zero_new:N \l_nm_final_i_int
1229   \int_zero_new:N \l_nm_final_j_int
1230   \bool_set_false:N \l_nm_initial_open_bool
1231   \bool_set_false:N \l_nm_final_open_bool
1232   \g_nm_lines_to_draw_tl
1233 }
1234 \tl_gclear:N \g_nm_lines_to_draw_tl

```

```

1235 \tikzset
1236 {
1237     every~picture / .style =
1238     {
1239         overlay ,
1240         remember~picture ,
1241         name~prefix = nm - \int_use:N \g__nm_env_int -
1242     }
1243 }
1244 \cs_set_eq:NN \line \__nm_line:nn
1245 \g__nm_code_after_tl
1246 \tl_gclear:N \g__nm_code_after_tl
1247 \group_end:
1248 }

```

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

51

- 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.

```

1249 \cs_new_protected:Nn \__nm_find_extremities_of_line:nnnn
1250 {
1251   \int_set:Nn \l__nm_initial_i_int { #1 }
1252   \int_set:Nn \l__nm_initial_j_int { #2 }
1253   \int_set:Nn \l__nm_final_i_int { #1 }
1254   \int_set:Nn \l__nm_final_j_int { #2 }

```

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

```

1255   \bool_set_false:N \l__nm_stop_loop_bool
1256   \bool_do_until:Nn \l__nm_stop_loop_bool
1257   {
1258     \int_add:Nn \l__nm_final_i_int { #3 }
1259     \int_add:Nn \l__nm_final_j_int { #4 }

```

We test if we are still in the matrix.

```

1260     \bool_set_false:N \l__nm_final_open_bool
1261     \int_compare:nNnTF \l__nm_final_i_int > \g__nm_row_int
1262     {
1263       \int_compare:nNnT { #3 } = 1
1264       { \bool_set_true:N \l__nm_final_open_bool }
1265     }
1266     {
1267       \int_compare:nNnTF \l__nm_final_j_int < 1
1268       {
1269         \int_compare:nNnT { #4 } = { -1 }
1270         { \bool_set_true:N \l__nm_final_open_bool }
1271       }
1272       {
1273         \int_compare:nNnT \l__nm_final_j_int > \g__nm_col_int
1274         {
1275           \int_compare:nNnT { #4 } = 1
1276           { \bool_set_true:N \l__nm_final_open_bool }
1277         }
1278       }
1279     }
1280     \bool_if:NTF \l__nm_final_open_bool

```

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

```

1281   {

```

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).

```

1282     \int_sub:Nn \l__nm_final_i_int { #3 }
1283     \int_sub:Nn \l__nm_final_j_int { #4 }
1284     \bool_set_true:N \l__nm_stop_loop_bool
1285   }

```

If we are in the matrix, we test if 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`.

```

1286     {
1287         \__nm_if_not_empty_cell:nnT \l__nm_final_i_int \l__nm_final_j_int
1288         { \bool_set_true:N \l__nm_stop_loop_bool }
1289     }
1290 }

```

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

```

1291     \bool_set_false:N \l__nm_stop_loop_bool
1292     \bool_do_until:Nn \l__nm_stop_loop_bool
1293     {
1294         \int_sub:Nn \l__nm_initial_i_int { #3 }
1295         \int_sub:Nn \l__nm_initial_j_int { #4 }
1296         \bool_set_false:N \l__nm_initial_open_bool
1297         \int_compare:nNnTF \l__nm_initial_i_int < 1
1298         {
1299             \int_compare:nNnT { #3 } = 1
1300             { \bool_set_true:N \l__nm_initial_open_bool }
1301         }
1302         {
1303             \int_compare:nNnTF \l__nm_initial_j_int < 1
1304             {
1305                 \int_compare:nNnT { #4 } = 1
1306                 { \bool_set_true:N \l__nm_initial_open_bool }
1307             }
1308             {
1309                 \int_compare:nNnT \l__nm_initial_j_int > \g__nm_col_int
1310                 {
1311                     \int_compare:nNnT { #4 } = { -1 }
1312                     { \bool_set_true:N \l__nm_initial_open_bool }
1313                 }
1314             }
1315         }
1316         \bool_if:NTF \l__nm_initial_open_bool
1317         {
1318             \int_add:Nn \l__nm_initial_i_int { #3 }
1319             \int_add:Nn \l__nm_initial_j_int { #4 }
1320             \bool_set_true:N \l__nm_stop_loop_bool
1321         }
1322         {
1323             \__nm_if_not_empty_cell:nnT
1324             \l__nm_initial_i_int \l__nm_initial_j_int
1325             { \bool_set_true:N \l__nm_stop_loop_bool }
1326         }
1327     }

```

If we have at least one open extremity, we create the “medium nodes” in the matrix²⁴. We remind that, when used once, the command `\@@_create_extra_nodes:` becomes no-op in the current TeX group.

```

1328     \bool_if:nT { \l__nm_initial_open_bool || \l__nm_final_open_bool }
1329     \__nm_create_extra_nodes:
1330 }

```

If the dotted line to draw is in the list of the previously drawn lines (`\g_@@_yet_drawn_seq`), we don't draw (so, we won't have overlapping lines in the PDF). The token list `\l_tmpa_tl` is the 4-list characteristic of the line.

²⁴We 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”.

```

1331 \prg_set_conditional:Npnn \__nm_if_yet_drawn: { F }
1332 {
1333   \tl_set:Nx \l_tmpa_tl
1334   {
1335     \int_use:N \l__nm_initial_i_int -
1336     \int_use:N \l__nm_initial_j_int -
1337     \int_use:N \l__nm_final_i_int -
1338     \int_use:N \l__nm_final_j_int
1339   }
1340   \seq_if_in:NVTF \g__nm_yet_drawn_seq \l_tmpa_tl

```

If the dotted line to draw is not in the list, we add it to the list `\g__nm_yet_drawn_seq`.

```

1341   \prg_return_true:
1342   {
1343     \seq_gput_left:NV \g__nm_yet_drawn_seq \l_tmpa_tl
1344     \prg_return_false:
1345   }
1346 }

```

The command `\@@_retrieve_coords:nn` retrieves the Tikz coordinates of the two extremities of the dotted line we will have to draw ²⁵. This command has four implicit arguments which are `\l__nm_initial_i_int`, `\l__nm_initial_j_int`, `\l__nm_final_i_int` and `\l__nm_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__nm_x_initial_dim`, `\g__nm_y_initial_dim`, `\g__nm_x_final_dim`, `\g__nm_y_final_dim`. These variables are global for technical reasons: we have to do an affectation in an environment `{tikzpicture}`.

```

1347 \cs_new_protected:Nn \__nm_retrieve_coords:nn
1348 {
1349   \dim_gzero_new:N \g__nm_x_initial_dim
1350   \dim_gzero_new:N \g__nm_y_initial_dim
1351   \dim_gzero_new:N \g__nm_x_final_dim
1352   \dim_gzero_new:N \g__nm_y_final_dim
1353   \begin { tikzpicture } [ remember-picture ]
1354     \tikz@parse@node \pgfutil@firstofone
1355       ( nm - \int_use:N \g__nm_env_int -
1356         \int_use:N \l__nm_initial_i_int -
1357         \int_use:N \l__nm_initial_j_int #1 )
1358     \dim_gset:Nn \g__nm_x_initial_dim \pgf@x
1359     \dim_gset:Nn \g__nm_y_initial_dim \pgf@y
1360     \tikz@parse@node \pgfutil@firstofone
1361       ( nm - \int_use:N \g__nm_env_int -
1362         \int_use:N \l__nm_final_i_int -
1363         \int_use:N \l__nm_final_j_int #2 )
1364     \dim_gset:Nn \g__nm_x_final_dim \pgf@x
1365     \dim_gset:Nn \g__nm_y_final_dim \pgf@y
1366   \end { tikzpicture }
1367 }
1368 \cs_generate_variant:Nn \__nm_retrieve_coords:nn { x x }

```

```

1369 \cs_new_protected:Nn \__nm_draw_Ldots:nn
1370 {
1371   \__nm_find_extremities_of_line:nnnn { #1 } { #2 } \c_zero_int \c_one_int
1372   \__nm_if_yet_drawn:F \__nm_actually_draw_Ldots:
1373 }

```

The command `\@@_actually_draw_Ldots:` draws the `Ldots` line using `\l__nm_initial_i_int`, `\l__nm_initial_j_int`, `\l__nm_initial_open_bool`, `\l__nm_final_i_int`, `\l__nm_final_j_int` and `\l__nm_final_open_bool`. We have a dedicated command because it is used also by `\Hdotsfor`.

²⁵In fact, with diagonal lines, or vertical lines in columns of type L or R, an adjustment of one of the coordinates may be done.

```

1374 \cs_new_protected:Nn \__nm_actually_draw_Ldots:
1375 {
1376     \__nm_retrieve_coords:xx
1377     {
1378         \bool_if:NTF \l__nm_initial_open_bool
1379         {

```

If a `w` node exists (created when the key `columns-width` is used), we use the `w` node for the extremity.

```

1380         \cs_if_exist:cTF
1381         {
1382             pgf@sh@ns@nm
1383             - \int_use:N \g__nm_env_int
1384             - \int_use:N \l__nm_initial_i_int
1385             - \int_use:N \l__nm_initial_j_int - w
1386         }
1387         { - w.base~west }
1388         { - medium.base~west }
1389     }
1390     { .base~east }
1391 }
1392 {
1393     \bool_if:NTF \l__nm_final_open_bool
1394     {
1395         \cs_if_exist:cTF
1396         {
1397             pgf@sh@ns@nm
1398             - \int_use:N \g__nm_env_int
1399             - \int_use:N \l__nm_final_i_int
1400             - \int_use:N \l__nm_final_j_int - w
1401         }
1402         { - w.base~east }
1403         { - medium.base~east }
1404     }
1405     { .base~west }
1406 }
1407 \bool_if:NT \l__nm_initial_open_bool
1408 { \dim_gset_eq:NN \g__nm_y_initial_dim \g__nm_y_final_dim }
1409 \bool_if:NT \l__nm_final_open_bool
1410 { \dim_gset_eq:NN \g__nm_y_final_dim \g__nm_y_initial_dim }

```

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

```

1411 \dim_gadd:Nn \g__nm_y_initial_dim { 0.53 pt }
1412 \dim_gadd:Nn \g__nm_y_final_dim { 0.53 pt }
1413 \__nm_draw_tikz_line:
1414 }

```

```

1415 \cs_new_protected:Nn \__nm_draw_Cdots:nn
1416 {
1417     \__nm_find_extremities_of_line:nnnn { #1 } { #2 } \c_zero_int \c_one_int
1418     \__nm_if_yet_drawn:F
1419     {
1420         \__nm_retrieve_coords:xx
1421         {
1422             \bool_if:NTF \l__nm_initial_open_bool
1423             {
1424                 \cs_if_exist:cTF
1425                 {
1426                     pgf@sh@ns@nm
1427                     - \int_use:N \g__nm_env_int
1428                     - \int_use:N \l__nm_initial_i_int
1429                     - \int_use:N \l__nm_initial_j_int - w
1430                 }
1431                 { - w.mid~west }

```

```

1432         { - medium.mid-west }
1433     }
1434     { .mid-east }
1435 }
1436 {
1437     \bool_if:NTF \l__nm_final_open_bool
1438     {
1439         \cs_if_exist:cTF
1440         {
1441             pgf@sh@ns@nm
1442             - \int_use:N \g__nm_env_int
1443             - \int_use:N \l__nm_final_i_int
1444             - \int_use:N \l__nm_final_j_int - w
1445         }
1446         { - w.mid-east }
1447         { - medium.mid-east }
1448     }
1449     { .mid-west }
1450 }
1451 \bool_if:NT \l__nm_initial_open_bool
1452 { \dim_gset_eq:NN \g__nm_y_initial_dim \g__nm_y_final_dim }
1453 \bool_if:NT \l__nm_final_open_bool
1454 { \dim_gset_eq:NN \g__nm_y_final_dim \g__nm_y_initial_dim }
1455 \__nm_draw_tikz_line:
1456 }
1457 }

```

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?).

```

1458 \cs_new_protected:Nn \__nm_draw_Vdots:nn
1459 {
1460     \__nm_find_extremities_of_line:nnnn { #1 } { #2 } \c_one_int \c_zero_int
1461     \__nm_if_yet_drawn:F
1462     { \__nm_retrieve_coords:xx
1463         {
1464             \bool_if:NTF \l__nm_initial_open_bool
1465             { - medium.north-west }
1466             { .south-west }
1467         }
1468         {
1469             \bool_if:NTF \l__nm_final_open_bool
1470             { - medium.south-west }
1471             { .north-west }
1472         }
1473     }
1474 }

```

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

```

1473 \bool_set:Nn \l_tmpa_bool
1474 { \dim_compare_p:nNn \g__nm_x_initial_dim = \g__nm_x_final_dim }
1475 \__nm_retrieve_coords:xx
1476 {
1477     \bool_if:NTF \l__nm_initial_open_bool
1478     { - medium.north }
1479     { .south }
1480 }
1481 {
1482     \bool_if:NTF \l__nm_final_open_bool
1483     { - medium.south }
1484     { .north }
1485 }

```

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


```

1486 \bool_set:Nn \l_tmpb_bool
1487 { \dim_compare_p:nNn \g__nm_x_initial_dim = \g__nm_x_final_dim }
1488 \bool_if:NF \l_tmpb_bool
1489 {
1490   \dim_gset:Nn \g__nm_x_initial_dim
1491   {
1492     \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
1493     \g__nm_x_initial_dim \g__nm_x_final_dim
1494   }
1495   \dim_gset_eq:NN \g__nm_x_final_dim \g__nm_x_initial_dim
1496 }
1497 \__nm_draw_tikz_line:
1498 }
1499 }

```

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.

```

1500 \cs_new_protected:Nn \__nm_draw_Ddots:nn
1501 {
1502   \__nm_find_extremities_of_line:nnnn { #1 } { #2 } \c_one_int \c_one_int
1503   \__nm_if_yet_drawn:F
1504   {
1505     \__nm_retrieve_coords:xx
1506     {
1507       \bool_if:NTF \l__nm_initial_open_bool
1508       { - medium.north-west }
1509       { .south-east }
1510     }
1511     {
1512       \bool_if:NTF \l__nm_final_open_bool
1513       { - medium.south-east }
1514       { .north-west }
1515     }
1516   }
1517 }

```

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.

```

1516 \bool_if:NT \l__nm_parallelize_diags_bool
1517 {
1518   \int_incr:N \l__nm_ddots_int

```

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

```

1519 \int_compare:nNnTF \l__nm_ddots_int = \c_one_int

```

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

```

1520 {
1521   \dim_set:Nn \l__nm_delta_x_one_dim
1522   { \g__nm_x_final_dim - \g__nm_x_initial_dim }
1523   \dim_set:Nn \l__nm_delta_y_one_dim
1524   { \g__nm_y_final_dim - \g__nm_y_initial_dim }
1525 }

```

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`.

```

1526 {
1527   \dim_gset:Nn \g__nm_y_final_dim
1528   {
1529     \g__nm_y_initial_dim +
1530     ( \g__nm_x_final_dim - \g__nm_x_initial_dim ) *
1531     \dim_ratio:nn \l__nm_delta_y_one_dim \l__nm_delta_x_one_dim
1532   }

```

```

1533     }
1534 }

```

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

```

1535     \__nm_draw_tikz_line:
1536 }
1537 }

```

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

```

1538 \cs_new_protected:Nn \__nm_draw_Iddots:nn
1539 {
1540   \__nm_find_extremities_of_line:nnnn { #1 } { #2 } 1 { -1 }
1541   \__nm_if_yet_drawn:F
1542   { \__nm_retrieve_coords:xx
1543     {
1544       \bool_if:NTF \l__nm_initial_open_bool
1545       { - medium.north~east }
1546       { .south~west }
1547     }
1548     {
1549       \bool_if:NTF \l__nm_final_open_bool
1550       { - medium.south~west }
1551       { .north~east }
1552     }
1553     \bool_if:NT \l__nm_parallelize_diags_bool
1554     {
1555       \int_incr:N \l__nm_iddots_int
1556       \int_compare:nNnTF \l__nm_iddots_int = \c_one_int
1557       {
1558         \dim_set:Nn \l__nm_delta_x_two_dim
1559         { \g__nm_x_final_dim - \g__nm_x_initial_dim }
1560         \dim_set:Nn \l__nm_delta_y_two_dim
1561         { \g__nm_y_final_dim - \g__nm_y_initial_dim }
1562       }
1563       {
1564         \dim_gset:Nn \g__nm_y_final_dim
1565         {
1566           \g__nm_y_initial_dim +
1567           ( \g__nm_x_final_dim - \g__nm_x_initial_dim ) *
1568           \dim_ratio:nn \l__nm_delta_y_two_dim \l__nm_delta_x_two_dim
1569         }
1570       }
1571     }
1572     \__nm_draw_tikz_line:
1573   }
1574 }

```

13.10 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`.

```

1575 \cs_new_protected:Nn \__nm_draw_tikz_line:
1576 {

```

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.

```

1577   \dim_zero_new:N \l__nm_l_dim
1578   \dim_set:Nn \l__nm_l_dim
1579   {
1580     \fp_to_dim:n

```

```

1581 {
1582     sqrt
1583     (
1584         ( \dim_use:N \g__nm_x_final_dim
1585           - \dim_use:N \g__nm_x_initial_dim
1586         ) ^ 2
1587         +
1588         ( \dim_use:N \g__nm_y_final_dim
1589           - \dim_use:N \g__nm_y_initial_dim
1590         ) ^ 2
1591     )
1592 }
1593 }

```

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

```

1594 \dim_compare:nNnF \l__nm_l_dim = \c_zero_dim

```

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

```

1595 {
1596     \bool_if:NTF \l__nm_initial_open_bool
1597     {
1598         \bool_if:NTF \l__nm_final_open_bool
1599         {
1600             \int_set:Nn \l_tmpa_int
1601             { \dim_ratio:nn \l__nm_l_dim { 0.45 em } }
1602         }
1603         {
1604             \int_set:Nn \l_tmpa_int
1605             { \dim_ratio:nn { \l__nm_l_dim - 0.3 em } { 0.45 em } }
1606         }
1607     }
1608     {
1609         \bool_if:NTF \l__nm_final_open_bool
1610         {
1611             \int_set:Nn \l_tmpa_int
1612             { \dim_ratio:nn { \l__nm_l_dim - 0.3 em } { 0.45 em } }
1613         }
1614         {
1615             \int_set:Nn \l_tmpa_int
1616             { \dim_ratio:nn { \l__nm_l_dim - 0.6 em } { 0.45 em } }
1617         }
1618     }

```

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

```

1619 \dim_set:Nn \l_tmpa_dim
1620 {
1621     ( \g__nm_x_final_dim - \g__nm_x_initial_dim ) *
1622     \dim_ratio:nn { 0.45 em } \l__nm_l_dim
1623 }
1624 \dim_set:Nn \l_tmpb_dim
1625 {
1626     ( \g__nm_y_final_dim - \g__nm_y_initial_dim ) *
1627     \dim_ratio:nn { 0.45 em } \l__nm_l_dim
1628 }

```

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`.

```

1629 \int_set:Nn \l_tmpb_int
1630 {
1631     \bool_if:NTF \l__nm_initial_open_bool
1632     { \bool_if:NTF \l__nm_final_open_bool 1 0 }

```

```

1633         { \bool_if:NTF \l__nm_final_open_bool 2 1 }
1634     }

```

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.

```

1635     \dim_gadd:Nn \g__nm_x_initial_dim
1636     {
1637         ( \g__nm_x_final_dim - \g__nm_x_initial_dim ) *
1638         \dim_ratio:nn
1639         { \l__nm_l_dim - 0.45 em * \l_tmpa_int } { \l__nm_l_dim * 2 } *
1640         \l_tmpb_int
1641     }

```

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

```

1642     \dim_gadd:Nn \g__nm_y_initial_dim
1643     {
1644         ( \g__nm_y_final_dim - \g__nm_y_initial_dim ) *
1645         \dim_ratio:nn
1646         { \l__nm_l_dim - 0.45 em * \l_tmpa_int }
1647         { \l__nm_l_dim * 2 } *
1648         \l_tmpb_int
1649     }
1650     \begin { tikzpicture } [ overlay ]
1651         \int_step_inline:nnnn 0 1 \l_tmpa_int
1652         {
1653             \pgfpathcircle
1654             { \pgfpoint { \g__nm_x_initial_dim } { \g__nm_y_initial_dim } }
1655             { 0.53 pt }
1656             \pgfusepath { fill }
1657             \dim_gadd:Nn \g__nm_x_initial_dim \l_tmpa_dim
1658             \dim_gadd:Nn \g__nm_y_initial_dim \l_tmpb_dim
1659         }
1660     \end { tikzpicture }
1661 }
1662 }

```

13.11 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).

```

1663 \cs_set_eq:NN \__nm_ldots \ldots
1664 \cs_set_eq:NN \__nm_cdots \cdots
1665 \cs_set_eq:NN \__nm_vdots \vdots
1666 \cs_set_eq:NN \__nm_ddots \ddots
1667 \cs_set_eq:NN \__nm_iddots \iddots

```

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

```

1668 \cs_new_protected:Nn \__nm_add_to_empty_cells:
1669 {
1670     \seq_gput_right:Nx \g__nm_empty_cells_seq
1671     { \int_use:N \g__nm_row_int - \int_use:N \g__nm_col_int }
1672 }

```

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}`).

```

1673 \NewDocumentCommand \__nm_Ldots { s }

```

```

1674 {
1675   \bool_if:nF { #1 } { \__nm_instruction_of_type:n { Ldots } }
1676   \bool_if:NF \l__nm_nullify_dots_bool { \phantom \__nm_ldots }
1677   \__nm_add_to_empty_cells:
1678 }

1679 \NewDocumentCommand \__nm_Cdots { s }
1680 {
1681   \bool_if:nF { #1 } { \__nm_instruction_of_type:n { Cdots } }
1682   \bool_if:NF \l__nm_nullify_dots_bool { \phantom \__nm_cdots }
1683   \__nm_add_to_empty_cells:
1684 }

1685 \NewDocumentCommand \__nm_Vdots { s }
1686 {
1687   \bool_if:nF { #1 } { \__nm_instruction_of_type:n { Vdots } }
1688   \bool_if:NF \l__nm_nullify_dots_bool { \phantom \__nm_vdots }
1689   \__nm_add_to_empty_cells:
1690 }

1691 \NewDocumentCommand \__nm_Ddots { s }
1692 {
1693   \bool_if:nF { #1 } { \__nm_instruction_of_type:n { Ddots } }
1694   \bool_if:NF \l__nm_nullify_dots_bool { \phantom \__nm_ddots }
1695   \__nm_add_to_empty_cells:
1696 }

1697 \NewDocumentCommand \__nm_Iddots { s }
1698 {
1699   \bool_if:nF { #1 } { \__nm_instruction_of_type:n { Iddots } }
1700   \bool_if:NF \l__nm_nullify_dots_bool { \phantom \__nm_iddots }
1701   \__nm_add_to_empty_cells:
1702 }

```

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

```

1703 \cs_new_protected:Nn \__nm_Hspace:
1704 {
1705   \__nm_add_to_empty_cells:
1706   \hspace
1707 }

```

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

```

1708 \cs_set_eq:NN \__nm_old_multicolumn \multicolumn
1709 \cs_new:Npn \__nm_multicolumn:nnn #1 #2 #3
1710 {
1711   \__nm_old_multicolumn { #1 } { #2 } { #3 }
1712   \int_compare:nNnT #1 > 1
1713   {
1714     \seq_gput_left:Nx \g__nm_multicolumn_cells_seq
1715     { \int_eval:n \g__nm_row_int - \int_use:N \g__nm_col_int }
1716     \seq_gput_left:Nn \g__nm_multicolumn_sizes_seq { #1 }
1717   }
1718   \int_gadd:Nn \g__nm_col_int { #1 - 1 }
1719 }

```

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`.

```

1720 \cs_new:Npn \__nm_Hdotsfor:
1721 {
1722   \multicolumn { 1 } { C } { }
1723   \__nm_Hdotsfor_i
1724 }

```

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

```

1725 \bool_if:NTF \c_nm_draft_bool
1726 {
1727   \NewDocumentCommand \__nm_Hdotsfor_i { 0 { } m }
1728   {
1729     \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { C } { } }
1730   }
1731 }
1732 {
1733   \NewDocumentCommand \__nm_Hdotsfor_i { 0 { } m }
1734   {
1735     \tl_gput_right:Nx \g__nm_lines_to_draw_tl
1736     {
1737       \__nm_draw_Hdotsfor:nnn
1738       { \int_use:N \g__nm_row_int }
1739       { \int_use:N \g__nm_col_int }
1740       { #2 }
1741     }
1742     \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { C } { } }
1743   }
1744 }

1745 \cs_new_protected:Nn \__nm_draw_Hdotsfor:nnn
1746 {
1747   \bool_set_false:N \l_nm_initial_open_bool
1748   \bool_set_false:N \l_nm_final_open_bool

```

For the row, it's easy.

```

1749 \int_set:Nn \l_nm_initial_i_int { #1 }
1750 \int_set:Nn \l_nm_final_i_int { #1 }

```

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

```

1751 \int_compare:nNnTF #2 = 1
1752 {
1753   \int_set:Nn \l_nm_initial_j_int 1
1754   \bool_set_true:N \l_nm_initial_open_bool
1755 }
1756 {
1757   \int_set:Nn \l_tmpa_int { #2 - 1 }
1758   \__nm_if_not_empty_cell:nnTF \l_nm_initial_i_int \l_tmpa_int
1759   { \int_set:Nn \l_nm_initial_j_int { #2 - 1 } }
1760   {
1761     \int_set:Nn \l_nm_initial_j_int {#2}
1762     \bool_set_true:N \l_nm_initial_open_bool
1763   }
1764 }
1765 \int_compare:nNnTF { #2 + #3 - 1 } = \g__nm_col_int
1766 {
1767   \int_set:Nn \l_nm_final_j_int { #2 + #3 - 1 }
1768   \bool_set_true:N \l_nm_final_open_bool

```

```

1769     }
1770     {
1771         \int_set:Nn \l_tmpa_int { #2 + #3 }
1772         \__nm_if_not_empty_cell:nnTF \l__nm_final_i_int \l_tmpa_int
1773         { \int_set:Nn \l__nm_final_j_int { #2 + #3 } }
1774         {
1775             \int_set:Nn \l__nm_final_j_int { #2 + #3 - 1 }
1776             \bool_set_true:N \l__nm_final_open_bool
1777         }
1778     }
1779     \bool_if:nT { \l__nm_initial_open_bool || \l__nm_final_open_bool }
1780     \__nm_create_extra_nodes:
1781     \__nm_actually_draw_ldots:
1782 }

```

13.12 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.

```

1783 \cs_new_protected:Nn \__nm_line:nn
1784 {
1785     \dim_zero_new:N \g__nm_x_initial_dim
1786     \dim_zero_new:N \g__nm_y_initial_dim
1787     \dim_zero_new:N \g__nm_x_final_dim
1788     \dim_zero_new:N \g__nm_y_final_dim
1789     \bool_set_false:N \l__nm_initial_open_bool
1790     \bool_set_false:N \l__nm_final_open_bool
1791     \begin { tikzpicture }
1792         \path~( #1 ) ~---~( #2 ) ~node[at~start]~( i ) ~{}~node[at~end]~( f ) ~{} ;
1793         \tikz@parse@node \pgfutil@firstofone ( i )
1794         \dim_gset:Nn \g__nm_x_initial_dim \pgf@x
1795         \dim_gset:Nn \g__nm_y_initial_dim \pgf@y
1796         \tikz@parse@node \pgfutil@firstofone ( f )
1797         \dim_gset:Nn \g__nm_x_final_dim \pgf@x
1798         \dim_gset:Nn \g__nm_y_final_dim \pgf@y
1799     \end { tikzpicture }
1800     \__nm_draw_tikz_line:
1801 }

```

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).

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

The command `\hdottedline` draws an horizontal dotted line to separate two rows. Similarly, the letter “.” in the preamble draws a vertical dotted line (the letter can be changed with the option `letter-for-dotted-lines`). Both mechanisms write instructions in the `code-after`. The actual instructions in the `code-after` use the commands `\@@_hdottedline:n` and `\@@_vdottedline:n`.

We want the horizontal lines at the same position²⁶ as the line created by `\hline` (or `\hdashline` of `arydshln`). To this end, we construct a “false row” and, in this row, we create a Tikz node (`\coordinate`) that will be used to have the *y*-value of the line.

```

1802 \cs_generate_variant:Nn \dim_set:Nn { N v }

```

²⁶In fact, almost the same position because of the width of the line: the width of a dotted line is not the same as the width of a line created by `\hline`.

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

```

1803 \bool_if:NTF \c_nm_draft_bool
1804 { \cs_set_eq:NN \__nm_dotfill: \prg_do_nothing: }
1805 {
1806   \cs_set:Npn \__nm_dotfill:
1807   {
1808     \cleaders \hbox_to_wd:nn { .44 em } { \hss . \hss } \hfill
1809     \skip_horizontal:n \c_zero_dim
1810   }
1811 }

```

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

```

1812 \cs_new:Npn \__nm_hdottedline:
1813 {
1814   \noalign
1815   {
1816     \bool_gset_true:N \g_nm_extra_nodes_bool
1817     \cs_if_exist:cTF { __nm_width_ \int_use:N \g_nm_env_int }
1818     { \dim_set:Nv \l_tmpa_dim { __nm_width_ \int_use:N \g_nm_env_int } }
1819     { \dim_set:Nn \l_tmpa_dim { 5 mm } }
1820     \hbox_overlap_right:n
1821     {
1822       \hbox_to_wd:nn
1823       {
1824         \l_tmpa_dim + 2 \arraycolsep
1825         - \l_nm_left_margin_dim - \g_nm_right_margin_dim
1826       }
1827       \__nm_dotfill:
1828     }
1829   }
1830 }

```

```

1831 \cs_new_protected:Nn \__nm_vdottedline:n
1832 {

```

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

```

1833 \int_compare:nNnTF #1 = \c_zero_int
1834 { \__nm_error:n { Use-of~::~in-first-position } }
1835 {
1836   \__nm_create_extra_nodes:
1837   \bool_if:NF \c_nm_draft_bool
1838   {
1839     \dim_zero_new:N \g_nm_x_initial_dim
1840     \dim_zero_new:N \g_nm_y_initial_dim
1841     \dim_zero_new:N \g_nm_x_final_dim
1842     \dim_zero_new:N \g_nm_y_final_dim
1843     \bool_set_true:N \l_nm_initial_open_bool
1844     \bool_set_true:N \l_nm_final_open_bool

```

In order to have the coordinates of the line to draw, we use the “large nodes”.

```

1845 \begin { tikzpicture } [ remember~picture ]
1846   \tikz@parse@node\pgfutil@firstofone
1847   ( 1 - #1 - large .north-east )
1848   \dim_gset:Nn \g_nm_x_initial_dim \pgf@x
1849   \dim_gset:Nn \g_nm_y_initial_dim \pgf@y
1850   \tikz@parse@node\pgfutil@firstofone
1851   ( \int_use:N \g_nm_row_int - #1 - large .south-east )
1852   \dim_gset:Nn \g_nm_x_final_dim \pgf@x
1853   \dim_gset:Nn \g_nm_y_final_dim \pgf@y
1854 \end { tikzpicture }

```


However, if the `w`-nodes are created in the previous column (that is if the previous column was constructed explicitly or implicitly²⁷ with a letter `w`), we use the `w`-nodes to change the x -value of the nodes in order to have the dotted lines perfectly aligned when we use the environment `{NiceMatrixBlock}` with the option `auto-columns-width`.

```

1855         \cs_if_exist:cT
1856         { pgf@sh@ns@nm -\int_use:N \g__nm_env_int - 1 - #1 - w }
1857         {
1858             \begin { tikzpicture } [ remember=picture ]
1859             \tikz@parse@node\pgfutil@firstofone
1860             ( 1 - #1 - w .north-east )
1861             \dim_gset:Nn \g__nm_x_initial_dim \pgf@x
1862             \tikz@parse@node\pgfutil@firstofone
1863             ( \int_use:N \g__nm_row_int - #1 - w .south-east )
1864             \dim_gset:Nn \g__nm_x_final_dim \pgf@x
1865             \end { tikzpicture }
1866             \dim_gadd:Nn \g__nm_x_initial_dim \arraycolsep
1867             \dim_gadd:Nn \g__nm_x_final_dim \arraycolsep
1868         }
1869         \__nm_draw_tikz_line:
1870     }
1871 }
1872 }
1873 \__nm_msg_new:nn { Use~of~::~in~first~position }
1874 {
1875     You~can't~use~the~column~specifier~"\l__nm_letter_for_dotted_lines_str"~in~the~
1876     first~position~of~the~preamble~of~the~environment~\{\@currenvir\}. \\\
1877     If~you~go~on,~this~dotted~line~will~be~ignored.
1878 }

```

13.14 The vertical rules

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

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

```

\newcolumntype { | }
{ ! { \int_compare:nNf \g_@@_row_int = \c_zero_int \vline } }

```

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

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

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

```

1879 \cs_new_protected:Npn \__nm_vline:
1880 {
1881     \int_compare:nNf \l__nm_first_col_int = \c_zero_int
1882     {
1883         \int_compare:nNf \g__nm_col_int = \c_zero_int
1884         {
1885             \int_compare:nNf \l__nm_first_row_int = \c_zero_int
1886             {
1887                 \int_compare:nNf \g__nm_row_int = \c_zero_int
1888                 {
1889                     \int_compare:nNf \g__nm_row_int = \l__nm_last_row_int \vline
1890                 }

```

²⁷A column is constructed implicitly with the letter `w` if the option `columns-width` is used or if the environment `{NiceMatrixBlock}` is used with the option `auto-columns-width`.

```

1891     }
1892     {
1893         \int_compare:nNnF \g__nm_row_int = \c_zero_int
1894         {
1895             \int_compare:nNnF \g__nm_row_int = \l__nm_last_row_int \vline
1896         }
1897     }
1898 }
1899 {
1900     \int_compare:nNnF \g__nm_row_int = \c_zero_int
1901     { \int_compare:nNnF \g__nm_row_int = \l__nm_last_row_int \vline }
1902 }
1903 }
1904 {
1905     \int_compare:nNnTF \g__nm_col_int = \c_zero_int
1906     {
1907         \int_compare:nNnF \g__nm_row_int = { -1 }
1908         {
1909             \int_compare:nNnF \g__nm_row_int = { \l__nm_last_row_int - 1 }
1910             \vline
1911         }
1912     }
1913     {
1914         \int_compare:nNnF \g__nm_row_int = \c_zero_int
1915         { \int_compare:nNnF \g__nm_row_int = \l__nm_last_row_int \vline }
1916     }
1917 }
1918 }

```

13.15 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.

```

1919 \bool_new:N \l__nm_block_auto_columns_width_bool

```

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

```

1920 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
1921 {
1922     auto-columns-width .code:n =
1923     {
1924         \bool_set_true:N \l__nm_block_auto_columns_width_bool
1925         \dim_gzero_new:N \g__nm_max_cell_width_dim
1926         \bool_set_true:N \l__nm_auto_columns_width_bool
1927     }
1928 }

1929 \NewDocumentEnvironment { NiceMatrixBlock } { ! 0 { } }
1930 {
1931     \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
1932     \int_zero_new:N \l__nm_first_env_block_int
1933     \int_set:Nn \l__nm_first_env_block_int { \g__nm_env_int + 1 }
1934 }

```

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__nm_first_env_block_int).

```

1935 {
1936     \bool_if:NT \l__nm_block_auto_columns_width_bool
1937     {
1938         \iow_now:Nn \@mainaux \ExplSyntaxOn

```

```

1939 \int_step_inline:nnnn \l__nm_first_env_block_int 1 \g__nm_env_int
1940 {
1941     \iow_now:Nx \@mainaux
1942     {
1943         \cs_gset:cpn { __nm_max_cell_width_ ##1 }
1944         { \dim_use:N \g__nm_max_cell_width_dim }
1945     }
1946 }
1947 \iow_now:Nn \@mainaux \ExplSyntaxOff
1948 }
1949 }

```

13.16 The extra nodes

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

```

1950 \cs_generate_variant:Nn \dim_min:nn { v n }
1951 \cs_generate_variant:Nn \dim_max:nn { v n }

```

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`.

```

1952 \cs_new_protected:Nn \__nm_create_extra_nodes:
1953 {
1954     \begin { tikzpicture } [ remember-picture , overlay ]
1955     \int_step_variable:nnNn \g__nm_first_row_int \g__nm_row_total_int \__nm_i:
1956     {
1957         \dim_zero_new:c { l__nm_row\___nm_i: _min_dim }
1958         \dim_set_eq:cN { l__nm_row\___nm_i: _min_dim } \c_max_dim
1959         \dim_zero_new:c { l__nm_row\___nm_i: _max_dim }
1960         \dim_set:cn { l__nm_row\___nm_i: _max_dim } { - \c_max_dim }
1961     }
1962     \int_step_variable:nnNn \g__nm_first_col_int \g__nm_col_total_int \__nm_j:
1963     {
1964         \dim_zero_new:c { l__nm_column\___nm_j: _min_dim }
1965         \dim_set_eq:cN { l__nm_column\___nm_j: _min_dim } \c_max_dim
1966         \dim_zero_new:c { l__nm_column\___nm_j: _max_dim }
1967         \dim_set:cn { l__nm_column\___nm_j: _max_dim } { - \c_max_dim }
1968     }

```

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

```

1969 \int_step_variable:nnNn \g__nm_first_row_int \g__nm_row_total_int \__nm_i:
1970 {
1971     \int_step_variable:nnNn
1972     \g__nm_first_col_int \g__nm_col_total_int \__nm_j:

```

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

```

1973 { \cs_if_exist:cT
1974     { pgf@sh@ns@nm - \int_use:N \g__nm_env_int - \__nm_i: - \__nm_j: }

```

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

```

1975 {
1976     \tikz@parse@node \pgfutil@firstofone
1977     ( nm - \int_use:N \g__nm_env_int
1978     - \__nm_i: - \__nm_j: .south-west )

```

```

1979 \dim_set:cn { l__nm_row _ \__nm_i: _min_dim}
1980 { \dim_min:vn { l__nm_row _ \__nm_i: _min_dim } \pgf@y }
1981 \seq_if_in:NxF \g__nm_multicolumn_cells_seq { \__nm_i: - \__nm_j: }
1982 {
1983 \dim_set:cn { l__nm_column _ \__nm_j: _min_dim}
1984 { \dim_min:vn { l__nm_column _ \__nm_j: _min_dim } \pgf@x }
1985 }

```

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`.

```

1986 \tikz@parse@node \pgfutil@firstofone
1987 ( nm - \int_use:N \g__nm_env_int - \__nm_i: - \__nm_j: .north-east )
1988 \dim_set:cn { l__nm_row _ \__nm_i: _ max_dim }
1989 { \dim_max:vn { l__nm_row _ \__nm_i: _ max_dim } \pgf@y }
1990 \seq_if_in:NxF \g__nm_multicolumn_cells_seq { \__nm_i: - \__nm_j: }
1991 {
1992 \dim_set:cn { l__nm_column _ \__nm_j: _ max_dim }
1993 { \dim_max:vn { l__nm_column _ \__nm_j: _ max_dim } \pgf@x }
1994 }
1995 }
1996 }
1997 }

```

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`).

```

1998 \tikzset { name-suffix = -medium }
1999 \__nm_create_nodes:

```

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

```

2000 \int_set:Nn \g__nm_first_row_int 1
2001 \int_set:Nn \g__nm_first_col_int 1

```

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

```

2002 \int_step_variable:nNn { \g__nm_row_int - 1 } \__nm_i:
2003 {
2004 \dim_set:cn { l__nm_row _ \__nm_i: _ min _ dim }
2005 {
2006 (
2007 \dim_use:c { l__nm_row _ \__nm_i: _ min _ dim } +
2008 \dim_use:c { l__nm_row _ \int_eval:n { \__nm_i: + 1 } _ max _ dim }
2009 )
2010 / 2
2011 }
2012 \dim_set_eq:cc { l__nm_row _ \int_eval:n { \__nm_i: + 1 } _ max _ dim }
2013 { l__nm_row _ \__nm_i: _ min_dim }
2014 }
2015 \int_step_variable:nNn { \g__nm_col_int - 1 } \__nm_j:
2016 {
2017 \dim_set:cn { l__nm_column _ \__nm_j: _ max _ dim }
2018 {
2019 (
2020 \dim_use:c
2021 { l__nm_column _ \__nm_j: _ max _ dim } +
2022 \dim_use:c
2023 { l__nm_column _ \int_eval:n { \__nm_j: + 1 } _ min _ dim }
2024 )
2025 / 2
2026 }
2027 \dim_set_eq:cc { l__nm_column _ \int_eval:n { \__nm_j: + 1 } _ min _ dim }
2028 { l__nm_column _ \__nm_j: _ max _ dim }
2029 }

```

```

2030 \dim_sub:cn
2031 { l__nm_column _ 1 _ min _ dim }
2032 \g__nm_left_margin_dim
2033 \dim_add:cn
2034 { l__nm_column _ \int_use:N \g__nm_col_int _ max _ dim }
2035 \g__nm_right_margin_dim

```

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

```

2036 \tikzset { name~suffix = -large }
2037 \__nm_create_nodes:
2038 \end{tikzpicture}

```

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

```

2039 \cs_set:Npn \__nm_create_extra_nodes: { }

```

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

```

2040 \begin { tikzpicture } [ remember~picture , overlay ]
2041 \tikz@parse@node \pgfutil@firstofone
2042 ( nm - \int_use:N \g__nm_env_int - 1 - 1 - large .north-west )
2043 \dim_gset:Nn \g_tmpa_dim \pgf@x
2044 \tikz@parse@node \pgfutil@firstofone
2045 ( nm - \int_use:N \g__nm_env_int - 1 -
2046 \int_use:N \g__nm_col_int - large .north-east )
2047 \dim_gset:Nn \g_tmpb_dim \pgf@x
2048 \end { tikzpicture }
2049 \iow_now:Nn \@mainaux \ExplSyntaxOn
2050 \iow_now:Nx \@mainaux
2051 {
2052 \cs_gset:cpn { __nm_width_ \int_use:N \g__nm_env_int }
2053 { \dim_eval:n { \g_tmpb_dim - \g_tmpa_dim } }
2054 }
2055 \iow_now:Nn \@mainaux \ExplSyntaxOff
2056 }

```

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.

```

2057 \cs_new_protected:Nn \__nm_create_nodes:
2058 {
2059 \int_step_variable:nnNn \g__nm_first_row_int \g__nm_row_total_int \__nm_i:
2060 {
2061 \int_step_variable:nnNn \g__nm_first_col_int \g__nm_col_total_int \__nm_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.

```

2062 {
2063 \coordinate ( __nm~south~west )
2064 at ( \dim_use:c { l__nm_column_ \__nm_j: _min_dim } ,
2065 \dim_use:c { l__nm_row_ \__nm_i: _min_dim } ) ;
2066 \coordinate ( __nm~north~east )
2067 at ( \dim_use:c { l__nm_column_ \__nm_j: _max_dim } ,
2068 \dim_use:c { l__nm_row_ \__nm_i: _max_dim } ) ;

```

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

```

2069 \node
2070 [

```

```

2071         node~contents = { } ,
2072         fit = ( __nm~south~west ) ( __nm~north~east ) ,
2073         inner~sep = \c_zero_dim ,
2074         name = nm - \int_use:N \g__nm_env_int - \__nm_i: - \__nm_j: ,
2075         alias =
2076             \str_if_empty:NF \g__nm_name_str
2077             { \g__nm_name_str - \__nm_i: - \__nm_j: }
2078     ]
2079 ;
2080 }
2081 }

```

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 .

```

2082     \__nm_seq_mapthread_function:NNN
2083     \g__nm_multicolumn_cells_seq
2084     \g__nm_multicolumn_sizes_seq
2085     \__nm_node_for_multicolumn:nn
2086 }
2087 \cs_new_protected:Npn \__nm_extract_coords: #1 - #2 \q_stop
2088 {
2089     \cs_set:Npn \__nm_i: { #1 }
2090     \cs_set:Npn \__nm_j: { #2 }
2091 }

```

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”).

```

2092 \cs_new_protected:Nn \__nm_node_for_multicolumn:nn
2093 {
2094     \__nm_extract_coords: #1 \q_stop
2095     \coordinate ( __nm~south~west ) at
2096     (
2097         \dim_use:c { l__nm_column _ \__nm_j: _ min _ dim } ,
2098         \dim_use:c { l__nm_row _ \__nm_i: _ min _ dim }
2099     ) ;
2100     \coordinate ( __nm~north~east ) at
2101     (
2102         \dim_use:c { l__nm_column _ \int_eval:n { \__nm_j: + #2 - 1 } _ max _ dim } ,
2103         \dim_use:c { l__nm_row _ \__nm_i: _ max _ dim }
2104     ) ;
2105     \node
2106     [
2107         node~contents = { } ,
2108         fit = ( __nm~south~west ) ( __nm~north~east ) ,
2109         inner~sep = \c_zero_dim ,
2110         name = nm - \int_use:N \g__nm_env_int - \__nm_i: - \__nm_j: ,
2111         alias =
2112             \str_if_empty:NF \g__nm_name_str { \g__nm_name_str - \__nm_i: - \__nm_j: }
2113     ]
2114     ;
2115 }

```

13.17 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.

```

2116 \ProcessKeysOptions { NiceMatrix }

```

13.18 Code for \seq_mapthread_function:NNN

In \@@_create_nodes: (used twice in \@@_create_extra_nodes: to create the “medium nodes” and “large nodes”), we want to use \seq_mapthread_function:NNN which is in l3candidates). For security, we define a function \@@_seq_mapthread_function:NNN. We will delete the following code when \seq_mapthread_function:NNN will be in l3seq.

```
2117 \cs_new:Npn \__nm_seq_mapthread_function:NNN #1 #2 #3
2118 {
2119   \group_begin:
```

In the group, we can use \seq_pop:NN safely.

```
2120   \int_step_inline:nn { \seq_count:N #1 }
2121   {
2122     \seq_pop:NN #1 \l_tmpa_tl
2123     \seq_pop:NN #2 \l_tmpb_tl
2124     \exp_args:NVV #3 \l_tmpa_tl \l_tmpb_tl
2125   }
2126   \group_end:
2127 }
```

```
2128 \cs_set_protected:Npn \__nm_renew_matrix:
2129 {
2130   \RenewDocumentEnvironment { pmatrix } { } {
2131     { \pNiceMatrix }
2132     { \endpNiceMatrix }
2133   }
2134   \RenewDocumentEnvironment { vmatrix } { } {
2135     { \vNiceMatrix }
2136     { \endvNiceMatrix }
2137   }
2138   \RenewDocumentEnvironment { Vmatrix } { } {
2139     { \VNiceMatrix }
2140     { \endVNiceMatrix }
2141   }
2142   \RenewDocumentEnvironment { bmatrix } { } {
2143     { \bNiceMatrix }
2144     { \endbNiceMatrix }
2145   }
2146   \RenewDocumentEnvironment { Bmatrix } { } {
2147     { \BNiceMatrix }
2148     { \endBNiceMatrix }
2149   }
2150 }
```

13.19 Obsolete environments

```
2146 \NewDocumentEnvironment { pNiceArrayC } { } {
2147   {
2148     \bool_set_true:N \l__nm_last_col_bool
2149     \pNiceArray
2150   }
2151   { \endpNiceArray }
2152 \NewDocumentEnvironment { bNiceArrayC } { } {
2153   {
2154     \bool_set_true:N \l__nm_last_col_bool
2155     \bNiceArray
2156   }
2157   { \endbNiceArray }
2158 \NewDocumentEnvironment { BNiceArrayC } { } {
2159   {
2160     \bool_set_true:N \l__nm_last_col_bool
2161     \BNiceArray
2162   }
2163   { \endBNiceArray }
2164 \NewDocumentEnvironment { vNiceArrayC } { } {
2165   {
2166     \bool_set_true:N \l__nm_last_col_bool
```

```

2167     \vNiceArray
2168   }
2169   { \endvNiceArray }

2170 \NewDocumentEnvironment { VNiceArrayC } { }
2171 {
2172   \bool_set_true:N \l__nm_last_col_bool
2173   \VNiceArray
2174 }
2175 { \endVNiceArray }

2176 \NewDocumentEnvironment { pNiceArrayRC } { }
2177 {
2178   \bool_set_true:N \l__nm_last_col_bool
2179   \int_set:Nn \l__nm_first_row_int \c_zero_int
2180   \pNiceArray
2181 }
2182 { \endpNiceArray }

2183 \NewDocumentEnvironment { bNiceArrayRC } { }
2184 {
2185   \bool_set_true:N \l__nm_last_col_bool
2186   \int_set:Nn \l__nm_first_row_int \c_zero_int
2187   \bNiceArray
2188 }
2189 { \endbNiceArray }

2190 \NewDocumentEnvironment { BNiceArrayRC } { }
2191 {
2192   \bool_set_true:N \l__nm_last_col_bool
2193   \int_set:Nn \l__nm_first_row_int \c_zero_int
2194   \BNiceArray
2195 }
2196 { \endBNiceArray }

2197 \NewDocumentEnvironment { vNiceArrayRC } { }
2198 {
2199   \bool_set_true:N \l__nm_last_col_bool
2200   \int_set:Nn \l__nm_first_row_int \c_zero_int
2201   \vNiceArray
2202 }
2203 { \endvNiceArray }

2204 \NewDocumentEnvironment { VNiceArrayRC } { }
2205 {
2206   \bool_set_true:N \l__nm_last_col_bool
2207   \int_set:Nn \l__nm_first_row_int \c_zero_int
2208   \VNiceArray
2209 }
2210 { \endVNiceArray }

2211 \NewDocumentEnvironment { NiceArrayCwithDelims } { }
2212 {
2213   \bool_set_true:N \l__nm_last_col_bool
2214   \NiceArrayWithDelims
2215 }
2216 { \endNiceArrayWithDelims }

2217 \NewDocumentEnvironment { NiceArrayRCwithDelims } { }
2218 {
2219   \bool_set_true:N \l__nm_last_col_bool
2220   \int_set:Nn \l__nm_first_row_int \c_zero_int
2221   \NiceArrayWithDelims
2222 }
2223 { \endNiceArrayWithDelims }

```


14 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”. New names are in lowercase and hyphens (but backward compatibility is kept).

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 StackExchange²⁸, Tikz externalization is now deactivated in the environments of the extension `nicematrix`.²⁹

Changes between version 2.1 and 2.1.2

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

²⁸cf. tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package

²⁹Before this version, there was an error when using `nicematrix` with Tikz externalization. In any case, it's not possible to externalize the Tikz elements constructed by `nicematrix` because they use the options `overlay` and `remember picture`.

Changes between version 2.1.2 and 2.1.3

When searching the end of a dotted line from a command like `\Cdots` issued in the “main matrix” (not in the column `C`), the cells in the column `C` 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 <https://www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end>

Changes between version 2.1.4 and 2.1.5

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

Option `allow-duplicate-names`.

Changes between version 2.1.5 and 2.2

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

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

Changes between version 2.2 and 2.2.1

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

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

Changes between version 2.2.1 and 2.3

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

Option `hlines`.

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

Changes between version 2.2.1 and 2.3

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

Changes between version 2.3 and 3.0

Composition of exterior rows and columns of the four sides of the matrix (and not only on two sides) with the options `first-row`, `last-row`, `first-col` and `last-col`.

Index

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

Symbols	
<code>\</code>	64, 70, 134, 139, 270, 276, 277, 310, 311, 334, 335, 358, 359, 396, 397, 1182, 1876
<code>\{</code>	63, 69, 310, 358, 396, 1055, 1112, 1180, 1876
<code>\}</code>	63, 69, 310, 358, 396, 1055, 1114, 1180, 1876
<code> </code>	1067
A	
<code>\array</code>	579
<code>\arraycolsep</code>	175, 177, 179, 760, 761, 860, 898, 900, 914, 971, 998, 1084, 1093, 1824, 1866, 1867
<code>\arrayrulewidth</code>	593, 594
<code>\AtBeginDocument</code>	74, 94
B	
<code>\baselineskip</code>	820, 830, 841, 852
<code>\begin</code>	1100, 1106, 1112, 1118, 1124, 1157, 1353, 1650, 1791, 1845, 1858, 1954, 2040
<code>\bgroup</code>	496
<code>\BNiceArray</code>	2161, 2194
<code>\bNiceArray</code>	2155, 2187
<code>\BNiceMatrix</code>	2143
<code>\bNiceMatrix</code>	2140
bool commands:	
<code>\bool_do_until:Nn</code>	1256, 1292
<code>\bool_gset_eq:NN</code>	619, 627, 628
<code>\bool_gset_false:N</code>	716
<code>\bool_gset_true:N</code>	704, 978, 1816
<code>\bool_if:NTF</code>	35, 102, 556, 571, 587, 601, 632, 670, 714, 722, 729, 732, 758, 789, 791, 795, 798, 800, 863, 874, 911, 1073, 1080, 1129, 1195, 1214, 1223, 1280, 1316, 1378, 1393, 1407, 1409, 1422, 1437, 1451, 1453, 1464, 1469, 1477, 1482, 1488, 1492, 1507, 1512, 1516, 1544, 1549, 1553, 1596, 1598, 1609, 1631, 1632, 1633, 1676, 1682, 1688, 1694, 1700, 1725, 1803, 1837, 1936
<code>\bool_if:nTF</code>	1191, 1328, 1675, 1681, 1687, 1693, 1699, 1779
<code>\bool_new:N</code>	11, 26, 52, 53, 54, 73, 89, 90, 91, 92, 93, 144, 145, 147, 148, 149, 152, 153, 1919
<code>\bool_set:Nn</code>	1473, 1486
<code>\bool_set_false:N</code>	1088, 1230, 1231, 1255, 1260, 1291, 1296, 1747, 1748, 1789, 1790
<code>\bool_set_true:N</code>	12, 28, 31, 77, 97, 146, 200, 249, 723, 738, 1036, 1074, 1264, 1270, 1276, 1284, 1288, 1300, 1306, 1312, 1320, 1325, 1754, 1762, 1768, 1776, 1843, 1844, 1924, 1926, 2148, 2154, 2160, 2166, 2172, 2178, 2185, 2192, 2199, 2206, 2213, 2219
<code>\l_tmpa_bool</code>	1473, 1492
<code>\l_tmpb_bool</code>	1486, 1488
box commands:	
<code>\box_clear_new:N</code>	784
<code>\box_dp:N</code>	454, 466, 516, 908
<code>\box_ht:N</code>	456, 461, 464, 907
<code>\box_move_down:n</code>	517, 878
<code>\box_move_up:n</code>	531, 869
<code>\box_set_dp:Nn</code>	908
<code>\box_set_ht:Nn</code>	907
<code>\box_use:N</code>	497, 515, 531, 963, 1030
<code>\box_use_drop:N</code>	886, 899, 909
<code>\box_wd:N</code>	473, 521, 773, 781, 936, 996
<code>\l_tmpa_box</code>	433, 454, 456, 461, 464, 466, 473, 497, 506, 515, 766, 773, 774, 781, 889, 907, 908, 909, 922, 936, 963, 982, 996, 1030
<code>\l_tmpb_box</code>	514, 516, 521, 531
C	
<code>\Cdots</code>	662
<code>\cdots</code>	673, 1664
Cdots internal commands:	
<code>_nm_Cdots</code>	662, 673, 1679
cdots internal commands:	
<code>_nm_cdots</code>	1664, 1682
<code>\cleaders</code>	1808
<code>\coordinate</code>	525, 530, 2063, 2066, 2095, 2100
cs commands:	
<code>\cs_generate_variant:Nn</code>	500, 1368, 1802, 1950, 1951
<code>\cs_gset:Npn</code>	1134, 1141, 1200, 1207, 1943, 2052
<code>\cs_gset_eq:NN</code>	115
<code>\cs_if_exist:NTF</code>	724, 741, 748, 1075, 1188, 1380, 1395, 1424, 1439, 1817, 1855, 1973
<code>\cs_if_free:NTF</code>	604, 612, 1150
<code>\cs_new:Npn</code>	582, 1709, 1720, 1812, 2117
<code>\cs_new_protected:Nn</code>	423, 442, 468, 501, 1127, 1172, 1185, 1249, 1347, 1369, 1374, 1415, 1458, 1500, 1538, 1575, 1668, 1703, 1745, 1783, 1831, 1952, 2057, 2092
<code>\cs_new_protected:Npn</code>	18, 19, 20, 21, 22, 23, 24, 55, 118, 449, 559, 569, 584, 599, 1879, 2087
<code>\cs_set:Npn</code>	576, 630, 642, 1806, 2039, 2089, 2090
<code>\cs_set_eq:NN</code>	106, 573, 574, 575, 661, 662, 663, 664, 665, 666, 667, 668, 669, 672, 673, 674, 675, 676, 677, 678, 688, 689, 690, 692, 1244, 1663, 1664, 1665, 1666, 1667, 1708, 1804
<code>\cs_set_protected:Npn</code>	100, 557, 731, 2128
D	
<code>\Ddots</code>	664
<code>\ddots</code>	675, 1666
Ddots internal commands:	
<code>_nm_Ddots</code>	664, 675, 1691
ddots internal commands:	
<code>_nm_ddots</code>	1666, 1694
<code>\DeclareOption</code>	12, 13
dim commands:	
<code>\dim_abs:n</code>	1166
<code>\dim_add:Nn</code>	2033
<code>\dim_compare:nNnTF</code>	650, 818, 836, 1165, 1594
<code>\dim_compare_p:nNn</code>	1474, 1487
<code>\dim_eval:n</code>	2053

<code>\dim_gadd:Nn</code>	1411, 1412, 1635, 1642, 1657, 1658, 1866, 1867
<code>\dim_gset:Nn</code>	453, 455, 460, 463, 465, 472, 760, 761, 773, 781, 932, 992, 1161, 1163, 1358, 1359, 1364, 1365, 1490, 1527, 1564, 1794, 1795, 1797, 1798, 1848, 1849, 1852, 1853, 1861, 1864, 2043, 2047
<code>\dim_gset_eq:NN</code>	445, 620, 621, 622, 1408, 1410, 1452, 1454, 1495
<code>\dim_gzero:N</code>	446, 447
<code>\dim_gzero_new:N</code>	693, 694, 695, 696, 697, 698, 730, 1081, 1349, 1350, 1351, 1352, 1925
<code>\dim_max:nn</code>	454, 456, 461, 464, 466, 473, 934, 994, 1492, 1951, 1989, 1993
<code>\dim_min:nn</code>	1492, 1950, 1980, 1984
<code>\dim_new:N</code>	50, 154, 155, 156, 157, 158, 159, 160, 161, 162
<code>\dim_ratio:nn</code>	1531, 1568, 1601, 1605, 1612, 1616, 1622, 1627, 1638, 1645
<code>\dim_set:Nn</code>	201, 255, 500, 516, 607, 614, 822, 829, 843, 850, 1521, 1523, 1558, 1560, 1578, 1619, 1624, 1802, 1818, 1819, 1960, 1967, 1979, 1983, 1988, 1992, 2004, 2017
<code>\dim_set_eq:NN</code>	765, 776, 1958, 1965, 2012, 2027
<code>\dim_sub:Nn</code>	2030
<code>\dim_use:N</code>	657, 658, 659, 1135, 1142, 1584, 1585, 1588, 1589, 1944, 2007, 2008, 2020, 2022, 2064, 2065, 2067, 2068, 2097, 2098, 2102, 2103
<code>\dim_zero:N</code>	605, 833, 857
<code>\dim_zero_new:N</code>	756, 757, 1218, 1219, 1220, 1221, 1577, 1785, 1786, 1787, 1788, 1839, 1840, 1841, 1842, 1957, 1959, 1964, 1966
<code>\c_max_dim</code>	1958, 1960, 1965, 1967
<code>\g_tmpa_dim</code>	1161, 1166, 2043, 2053
<code>\l_tmpa_dim</code>	516, 517, 531, 822, 829, 833, 870, 895, 907, 1619, 1657, 1818, 1819, 1824
<code>\g_tmpb_dim</code>	1163, 1166, 2047, 2053
<code>\l_tmpb_dim</code>	843, 850, 857, 880, 902, 908, 1624, 1658
<code>\c_zero_dim</code>	478, 479, 549, 650, 765, 776, 943, 944, 1010, 1011, 1594, 1809, 2073, 2109
<code>\dots</code>	677
E	
<code>\egroup</code>	498
else commands:	
<code>\else:</code>	57
<code>\end</code>	1102, 1108, 1114, 1120, 1126, 1164, 1366, 1660, 1799, 1854, 1865, 2038, 2048
<code>\endarray</code>	811, 1092
<code>\endBNiceArray</code>	2163, 2196
<code>\endbNiceArray</code>	2157, 2189
<code>\endBNiceMatrix</code>	2144
<code>\endbNiceMatrix</code>	2141
<code>\endNiceArrayWithDelims</code>	1039, 1045, 1051, 1057, 1063, 1069, 2216, 2223
<code>\endpNiceArray</code>	2151, 2182
<code>\endpNiceMatrix</code>	2132
<code>\endVNiceArray</code>	2175, 2210
<code>\endvNiceArray</code>	2169, 2203
<code>\endVNiceMatrix</code>	2138
<code>\endvNiceMatrix</code>	2135
<code>\everycr</code>	640, 644
exp commands:	
<code>\exp_after:wN</code>	122
<code>\exp_args:NV</code>	808
<code>\exp_args:NVV</code>	2124
<code>\exp_args:Nx</code>	699
<code>\ExplSyntaxOff</code>	1145, 1211, 1947, 2055
<code>\ExplSyntaxOn</code>	1131, 1197, 1938, 2049
F	
fi commands:	
<code>\fi:</code>	59
fp commands:	
<code>\fp_to_dim:n</code>	1580
G	
<code>\global</code>	636
group commands:	
<code>\group_begin:</code>	104, 764, 1187, 2119
<code>\group_end:</code>	109, 782, 1247, 2126
<code>\group_insert_after:N</code>	603, 726, 1077
H	
<code>\halign</code>	646, 648
hbox commands:	
<code>\hbox:n</code>	42, 44, 46, 527, 896
<code>\hbox_overlap_left:n</code>	938
<code>\hbox_overlap_right:n</code>	999, 1820
<code>\hbox_set:Nn</code>	514, 766, 774, 889
<code>\hbox_set:Nw</code>	433, 506, 804, 922, 982
<code>\hbox_set_end:</code>	471, 512, 815, 930, 990
<code>\hbox_to_wd:nn</code>	521, 1808, 1822
<code>\Hdotsfor</code>	668
<code>\hdotsfor</code>	678
<code>\hdottedline</code>	666
<code>\hfil</code>	523, 692
<code>\hfill</code>	1808
<code>\hrule</code>	593
<code>\Hspace</code>	667
<code>\hspace</code>	1706
<code>\hss</code>	692, 1808
I	
<code>\ialign</code>	630, 642
<code>\iddots</code>	665
<code>\iddots</code>	37, 676, 1667
lddots internal commands:	
<code>_nm_Iddots</code>	665, 676, 1697
iddots internal commands:	
<code>_nm_iddots</code>	1667, 1700
if commands:	
<code>\if_mode_math:</code>	57
int commands:	
<code>\int_add:Nn</code>	1258, 1259, 1318, 1319
<code>\int_compare:nNnTF</code>	426, 428, 435, 438, 451, 458, 589, 591, 705, 736, 786, 816, 834, 858, 865, 1174, 1193, 1261, 1263, 1267, 1269, 1273, 1275, 1297, 1299, 1303, 1305, 1309, 1311, 1519, 1556, 1712, 1751, 1765, 1833, 1881, 1883, 1885, 1887, 1889, 1893, 1895, 1900, 1901, 1905, 1907, 1909, 1914, 1915
<code>\int_compare:nTF</code>	261
<code>\int_eval:n</code>	1715, 2008, 2012, 2023, 2027, 2102
<code>\int_gadd:Nn</code>	1718
<code>\int_gdecr:N</code>	1191

<code>\int_gincr:N</code>	425, 444, 728, 979, 1079
<code>\int_gset:Nn</code>	431, 684, 715, 980
<code>\int_gset_eq:NN</code>	623, 625, 626, 707, 1190, 1192
<code>\int_gsub:Nn</code>	1194
<code>\int_gzero:N</code>	586
<code>\int_gzero_new:N</code>	683, 685, 686, 687, 713
<code>\int_incr:N</code>	1518, 1555
<code>\int_max:nn</code>	432, 981
<code>\int_new:N</code>	49, 80, 82, 83, 85, 86, 88
<code>\int_set:Nn</code>	81, 84, 87, 743, 750, 1251, 1252, 1253, 1254, 1600, 1604, 1611, 1615, 1629, 1749, 1750, 1753, 1757, 1759, 1761, 1767, 1771, 1773, 1775, 1933, 2000, 2001, 2179, 2186, 2193, 2200, 2207, 2220
<code>\int_step_inline:nn</code>	2120
<code>\int_step_inline:nnnn</code>	1651, 1939
<code>\int_step_variable:nNn</code>	2002, 2015
<code>\int_step_variable:nnNn</code>	1955, 1962, 1969, 1971, 2059, 2061
<code>\int_sub:Nn</code>	1282, 1283, 1294, 1295
<code>\int_use:N</code>	485, 486, 487, 492, 493, 539, 540, 541, 546, 547, 564, 565, 604, 608, 709, 741, 744, 952, 953, 959, 1019, 1020, 1021, 1026, 1027, 1134, 1151, 1154, 1159, 1200, 1201, 1208, 1241, 1335, 1336, 1337, 1338, 1355, 1356, 1357, 1361, 1362, 1363, 1383, 1384, 1385, 1398, 1399, 1400, 1427, 1428, 1429, 1442, 1443, 1444, 1671, 1715, 1738, 1739, 1817, 1818, 1851, 1856, 1863, 1974, 1977, 1987, 2034, 2042, 2045, 2046, 2052, 2074, 2110
<code>\int_zero:N</code>	348, 350, 386, 388
<code>\int_zero_new:N</code>	1216, 1217, 1226, 1227, 1228, 1229, 1932
<code>\c_one_int</code>	261, 426, 428, 458, 1194, 1371, 1417, 1460, 1502, 1519, 1556
<code>\l_tmpa_int</code>	1600, 1604, 1611, 1615, 1639, 1646, 1651, 1757, 1758, 1771, 1772
<code>\l_tmpb_int</code>	1629, 1640, 1648
<code>\c_zero_int</code>	435, 451, 786, 816, 858, 865, 1174, 1371, 1417, 1460, 1833, 1881, 1883, 1885, 1887, 1893, 1900, 1905, 1914, 2179, 2186, 2193, 2200, 2207, 2220
iow commands:	
<code>\iow_now:Nn</code>	1131, 1132, 1139, 1145, 1197, 1198, 1205, 1211, 1938, 1941, 1947, 2049, 2050, 2055
K	
<code>\kern</code>	46
keys commands:	
<code>\keys_define:nn</code>	163, 185, 196, 214, 239, 305, 343, 384, 1920
<code>\l_keys_key_tl</code>	270, 275, 309, 357, 395
<code>\keys_set:nn</code>	304, 733, 734, 1082, 1931
<code>\l_keys_value_tl</code>	331
L	
<code>\Ldots</code>	661
<code>\ldots</code>	672, 1663
Ldots internal commands:	
<code>__nm_Ldots</code>	661, 672, 677, 1673
ldots internal commands:	
<code>__nm_ldots</code>	1663, 1676
<code>\left</code>	769, 778, 892, 1100, 1106, 1112, 1118, 1124
<code>\let</code>	636
<code>\line</code>	1244
<code>\lineskip</code>	824, 845
<code>\lineskiplimit</code>	819, 838
<code>\lVert</code>	1124
<code>\lvert</code>	1118
M	
<code>\makebox</code>	515
math commands:	
<code>\c_math_toggle_token</code>	434, 470, 768, 770, 777, 779, 807, 812, 891, 905, 923, 929, 983, 989
<code>\mathinner</code>	39
<code>\mkern</code>	41, 43, 45, 46
msg commands:	
<code>\msg_error:nn</code>	18, 19
<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>	669, 1708, 1722, 1729, 1742
<code>\myfiledate</code>	8
<code>\myfileversion</code>	9
N	
<code>\newcolumntype</code>	503, 652, 653, 654, 657, 658, 659, 699
<code>\NewDocumentCommand</code>	303, 1673, 1679, 1685, 1691, 1697, 1727, 1733
<code>\NewDocumentEnvironment</code>	718, 1034, 1040, 1046, 1052, 1058, 1064, 1070, 1097, 1103, 1109, 1115, 1121, 1929, 2146, 2152, 2158, 2164, 2170, 2176, 2183, 2190, 2197, 2204, 2211, 2217
<code>\NiceArrayWithDelims</code>	1037, 1043, 1049, 1055, 1061, 1067, 2214, 2221
<code>\NiceMatrixOptions</code>	276, 303
nm internal commands:	
<code>__nm_actualization_for_first_and_- last_row:</code>	449, 474, 931, 991
<code>__nm_actually_draw_Ldots:</code>	1372, 1374, 1781
<code>__nm_adapt_S_column:</code>	100, 115, 720
<code>__nm_add_to_empty_cells:</code>	1668, 1677, 1683, 1689, 1695, 1701, 1705
<code>__nm_after_array:</code>	726, 1077, 1172
<code>__nm_after_array_i:</code>	1175, 1185
<code>__nm_array:</code>	569, 808, 1089
<code>\l_nm_auto_columns_width_bool</code>	149, 200, 601, 1926
<code>__nm_begin_of_row:</code>	429, 442, 921
<code>\l_nm_block_auto_columns_width_bool</code>	729, 1080, 1129, 1919, 1924, 1936
<code>__nm_Cell:</code>	125, 423, 507, 652, 653, 654
<code>\g_nm_code_after_tl</code>	210, 708, 1245, 1246
<code>\l_nm_code_for_first_col_tl</code>	187, 924
<code>\l_nm_code_for_first_row_tl</code>	191, 436
<code>\l_nm_code_for_last_col_tl</code>	189, 984
<code>\l_nm_code_for_last_row_tl</code>	193, 439
<code>\g_nm_col_int</code>	425, 426, 432, 487, 493, 541, 547, 565, 586, 686, 705, 707, 709, 979, 981, 1021, 1027, 1190, 1191, 1273, 1309, 1671, 1715, 1718, 1739, 1765, 1883, 1905, 2015, 2034, 2046

<code>\g_nm_col_total_int</code>	431,	1453, 1469, 1482, 1512, 1549, 1598, 1609,
	432, 687, 980, 981, 1190, 1962, 1972, 2061	1632, 1633, 1748, 1768, 1776, 1779, 1790, 1844
<code>\c_nm_colortbl_loaded_bool</code> ...	73, 77, 632	
<code>\l_nm_columns_width_dim</code>	50,	<code>_nm_find_extremities_of_line:nnnn</code> ..
	201, 255, 605, 607, 614, 650, 657, 658, 659 1249, 1371, 1417, 1460, 1502, 1540
<code>_nm_create_extra_nodes:</code>		<code>\g_nm_first_col_int</code>
 1223, 1329, 1780, 1836, 1952, 2039 85, 626, 858, 1962, 1972, 2001, 2061
<code>_nm_create_nodes:</code>	1999, 2037, 2057	<code>\l_nm_first_col_int</code>
<code>\l_nm_ddots_int</code>	1216, 1518, 1519 83, 84, 348, 386, 428, 626, 786, 1881
<code>\l_nm_delta_x_one_dim</code> ...	1218, 1521, 1531	<code>\l_nm_first_env_block_int</code> 1932, 1933, 1939
<code>\l_nm_delta_x_two_dim</code> ...	1220, 1558, 1568	<code>\g_nm_first_row_int</code>
<code>\l_nm_delta_y_one_dim</code> ...	1219, 1523, 1531 82, 625, 865, 1955, 1969, 2000, 2059
<code>\l_nm_delta_y_two_dim</code> ...	1221, 1560, 1568	<code>\l_nm_first_row_int</code>
<code>_nm_dotfill:</code>	1804, 1806, 1827 80, 81, 350, 388, 625, 684,
<code>\g_nm_dp_ante_last_row_dim</code>		816, 1885, 2179, 2186, 2193, 2200, 2207, 2220
 445, 696, 838, 845, 846, 853, 881	<code>_nm_Hdotsfor:</code>
<code>\g_nm_dp_last_row_dim</code>		668, 678, 1720
 445, 446, 465, 466, 698, 846, 853, 881	<code>_nm_Hdotsfor_i</code>
<code>\g_nm_dp_row_zero_dim</code> 453, 454, 693, 819, 824		1723, 1727, 1733
<code>\c_nm_draft_bool</code>		<code>_nm_hdottedline:</code>
 11, 12, 35, 556, 1725, 1803, 1837	666, 1812
<code>_nm_draw_Cdots:nn</code>	1415	<code>\l_nm_hlines_bool</code>
<code>_nm_draw_Ddots:nn</code>	1500	147, 165, 587
<code>_nm_draw_Hdotsfor:nnn</code>	1737, 1745	<code>_nm_Hspace:</code>
<code>_nm_draw_Iddots:nn</code>	1538	667, 1703
<code>_nm_draw_Ldots:nn</code>	1369	<code>\g_nm_ht_last_row_dim</code>
<code>_nm_draw_tikz_line:</code> 447, 463, 464, 697, 838, 845
1413, 1455, 1497, 1535, 1572, 1575, 1800, 1869		<code>\g_nm_ht_row_one_dim</code>
<code>_nm_draw_Vdots:nn</code>	1458 460, 461, 695, 819, 824, 825, 830, 870
<code>\g_nm_empty_cells_seq</code>	680, 1154, 1670	<code>\g_nm_ht_row_zero_dim</code>
<code>_nm_end_Cell:</code> . 127, 468, 511, 652, 653, 654	 455, 456, 694, 825, 830, 870
<code>\g_nm_env_int</code> ...	49, 485, 539, 604, 608,	<code>_nm_i:</code>
	728, 741, 744, 952, 1019, 1079, 1134, 1151,	1955, 1957,
	1159, 1200, 1241, 1355, 1361, 1383, 1398,	1958, 1959, 1960, 1969, 1974, 1978, 1979,
	1427, 1442, 1817, 1818, 1856, 1933, 1939,	1980, 1981, 1987, 1988, 1989, 1990, 2002,
	1974, 1977, 1987, 2042, 2045, 2052, 2074, 2110	2004, 2007, 2008, 2012, 2013, 2059, 2065,
<code>_nm_error:n</code>	18, 243,	2068, 2074, 2077, 2089, 2098, 2103, 2110, 2112
	247, 254, 263, 267, 306, 353, 391, 1176, 1834	<code>\l_nm_iddots_int</code>
<code>_nm_error:nn</code>	19, 206	1217, 1555, 1556
<code>_nm_everycr:</code>	582, 637, 640	<code>_nm_if_not_empty_cell:nn</code>
<code>_nm_everycr_i:</code>	583, 584	1148
<code>\l_nm_exterior_arraycolsep_bool</code>		<code>_nm_if_not_empty_cell:nnTF</code>
 144, 250, 791, 800, 1088 1287, 1323, 1758, 1772
<code>\l_nm_extra_left_margin_dim</code>		<code>_nm_if_yet_drawn:</code>
 160, 180, 806, 968, 1086	1331
<code>\g_nm_extra_nodes_bool</code>		<code>_nm_if_yet_drawn:TF</code>
 153, 619, 704, 1223, 1816 1372, 1418, 1461, 1503, 1541
<code>\l_nm_extra_nodes_bool</code>	152, 172, 619	<code>\l_nm_in_env_bool</code> . 52, 722, 723, 1073, 1074
<code>\g_nm_extra_right_margin_dim</code>		<code>\l_nm_initial_i_int</code>
 162, 622, 814, 1095 1226, 1251, 1294, 1297,
<code>\l_nm_extra_right_margin_dim</code>		1318, 1324, 1335, 1356, 1384, 1428, 1749, 1758
 161, 181, 622, 1005	<code>\l_nm_initial_j_int</code>
<code>_nm_extract_coords:</code>	2087, 2094 1227, 1252, 1295, 1303, 1309, 1319,
<code>_nm_fatal:n</code>	20, 58, 722, 1073	1324, 1336, 1357, 1385, 1429, 1753, 1759, 1761
<code>_nm_fatal:nn</code>	21	<code>\l_nm_initial_open_bool</code>
<code>\l_nm_final_i_int</code> 1228, 1253, 1258, 1261,		1230,
1282, 1287, 1337, 1362, 1399, 1443, 1750, 1772		1296, 1300, 1306, 1312, 1316, 1328, 1378,
<code>\l_nm_final_j_int</code>		1407, 1422, 1451, 1464, 1477, 1507, 1544,
..... 1229, 1254, 1259, 1267, 1273, 1283,		1596, 1631, 1747, 1754, 1762, 1779, 1789, 1843
1287, 1338, 1363, 1400, 1444, 1767, 1773, 1775		<code>_nm_instruction_of_type:n</code>
<code>\l_nm_final_open_bool</code> 1231, 1260, 1264,	 557, 559, 1675, 1681, 1687, 1693, 1699
1270, 1276, 1280, 1328, 1393, 1409, 1437,		<code>_nm_j:</code>
		1962, 1964,
		1965, 1966, 1967, 1972, 1974, 1978, 1981,
		1983, 1984, 1987, 1990, 1992, 1993, 2015,
		2017, 2021, 2023, 2027, 2028, 2061, 2064,
		2067, 2074, 2077, 2090, 2097, 2102, 2110, 2112
		<code>\l_nm_l_dim</code>
		1577, 1578, 1594, 1601,
		1605, 1612, 1616, 1622, 1627, 1639, 1646, 1647
		<code>\l_nm_last_col_bool</code>
		91,
		349, 387, 795, 2148, 2154, 2160, 2166,
		2172, 2178, 2185, 2192, 2199, 2206, 2213, 2219
		<code>\g_nm_last_col_found_bool</code>
	 92, 716, 911, 978, 1191
		<code>\g_nm_last_row_int</code> .. 88, 591, 623, 874, 1193

<code>\l_nm_last_row_int</code>	86, 87, 351, 389, 438, 623, 736, 743, 750, 834, 1889, 1895, 1901, 1909, 1915
<code>\g_nm_last_row_without_value_bool</code> ..	90, 628, 1195
<code>\l_nm_last_row_without_value_bool</code> ..	89, 629, 738
<code>\g_nm_last_vdotted_col_int</code>	705, 707, 713, 715
<code>\g_nm_left_delim_dim</code> ...	756, 760, 773, 966
<code>\g_nm_left_margin_dim</code>	156, 620, 2032
<code>\l_nm_left_margin_dim</code>	154, 174, 620, 805, 967, 1085, 1825
<code>\l_nm_letter_for_dotted_lines_str</code> ..	262, 699, 1875
<code>_nm_line:nn</code>	1244, 1783
<code>\g_nm_lines_to_draw_tl</code>	561, 727, 1078, 1224, 1232, 1234, 1735
<code>\g_nm_max_cell_width_dim</code>	472, 473, 730, 1081, 1135, 1142, 1925, 1944
<code>_nm_msg_new:nn</code>	22, 33, 61, 67, 132, 137, 268, 298, 1178, 1873
<code>_nm_msg_new:nnn</code> 23, 273, 307, 329, 355, 393	
<code>_nm_msg_redirect_name:nn</code>	24, 257
<code>_nm_multicolumn:nnn</code>	669, 1709
<code>\g_nm_multicolumn_cells_seq</code>	681, 1714, 1981, 1990, 2083
<code>\g_nm_multicolumn_sizes_seq</code> 682, 1716, 2084	
<code>\g_nm_name_str</code>	151, 624, 739, 748, 751, 1137, 1141, 1203, 1207, 2076, 2077, 2112
<code>\l_nm_name_str</code> .	150, 208, 489, 491, 543, 545, 610, 612, 615, 624, 956, 958, 1023, 1025
<code>\g_nm_names_seq</code>	51, 205, 207, 341
<code>\g_nm_NiceArray_bool</code>	54, 627, 863
<code>\l_nm_NiceArray_bool</code>	53, 627, 732, 758, 789, 798, 1036
<code>_nm_node_for_multicolumn:nn</code> ..	2085, 2092
<code>\l_nm_nullify_dots_bool</code>	148, 170, 1676, 1682, 1688, 1694, 1700
<code>_nm_old_multicolumn</code>	1708, 1711
<code>\l_nm_parallelize_diags_bool</code>	145, 146, 166, 1214, 1516, 1553
<code>\l_nm_pos_env_str</code>	142, 143, 345, 346, 347, 580, 867, 876, 1087
<code>_nm_pre_array:</code>	599, 755, 1083
<code>\c_nm_preamble_first_col_tl</code>	787, 917
<code>\c_nm_preamble_last_col_tl</code>	796, 974
<code>\l_nm_renew_dots_bool</code>	168, 249, 670
<code>_nm_renew_matrix:</code>	241, 244, 248, 2128
<code>_nm_renew_NC@rewrite@S:</code>	118, 714
<code>_nm_renewcolumnntype:nn</code>	501, 691, 692
<code>_nm_retrieve_coords:nn</code>	1347, 1368, 1376, 1420, 1462, 1475, 1505, 1542
<code>\c_nm_revtex_bool</code>	26, 28, 31, 571
<code>\g_nm_right_delim_dim</code> .	757, 761, 781, 1003
<code>\g_nm_right_margin_dim</code>	157, 621, 813, 1094, 1825, 2035
<code>\l_nm_right_margin_dim</code> 155, 176, 621, 1004	
<code>\g_nm_row_int</code>	435, 438, 444, 451, 458, 486, 492, 540, 546, 564, 589, 591, 683, 684, 953, 959, 1020, 1026, 1174, 1192, 1194, 1261, 1671, 1715, 1738, 1851, 1863, 1887, 1889, 1893, 1895, 1900, 1901, 1907, 1909, 1914, 1915, 2002
<code>\g_nm_row_total_int</code>	685, 1192, 1201, 1208, 1955, 1969, 2059
<code>_nm_seq_mapthread_function:NNN</code> 2082, 2117	
<code>\c_nm_siunitx_loaded_bool</code> 93, 97, 102, 714	
<code>\l_nm_stop_loop_bool</code>	1255, 1256, 1284, 1288, 1291, 1292, 1320, 1325
<code>\c_nm_table_collect_begin_tl</code> 110, 112, 125	
<code>\c_nm_table_print_tl</code>	113, 114, 127
<code>_nm_test_if_math_mode:</code>	55, 721, 1042, 1048, 1054, 1060, 1066, 1072, 1099, 1105, 1111, 1117, 1123
<code>\l_nm_the_array_box</code>	784, 804, 886, 899
<code>_nm_vdottedline:n</code>	709, 1831
<code>_nm_vline:</code>	731, 1879
<code>\g_nm_width_first_col_dim</code> 159, 861, 932, 935	
<code>\g_nm_width_last_col_dim</code> 158, 913, 992, 995	
<code>_nm_write_max_cell_width:</code>	603, 1127
<code>\g_nm_x_final_dim</code> 1351, 1364, 1474, 1487, 1493, 1495, 1522, 1530, 1559, 1567, 1584, 1621, 1637, 1787, 1797, 1841, 1852, 1864, 1867	
<code>\g_nm_x_initial_dim</code>	1349, 1358, 1474, 1487, 1490, 1493, 1495, 1522, 1530, 1559, 1567, 1585, 1621, 1635, 1637, 1654, 1657, 1785, 1794, 1839, 1848, 1861, 1866
<code>\g_nm_y_final_dim</code> ...	1352, 1365, 1408, 1410, 1412, 1452, 1454, 1524, 1527, 1561, 1564, 1588, 1626, 1644, 1788, 1798, 1842, 1853
<code>\g_nm_y_initial_dim</code>	1350, 1359, 1408, 1410, 1411, 1452, 1454, 1524, 1529, 1561, 1566, 1589, 1626, 1642, 1644, 1654, 1658, 1786, 1795, 1840, 1849
<code>\g_nm_yet_drawn_seq</code>	1213, 1340, 1343
<code>\noalign</code>	583, 636, 1814
<code>\node</code>	482, 536, 947, 1014, 2069, 2105
<code>\nulldelimiterspace</code>	765, 776
P	
<code>\path</code>	1792
<code>\pgfpathcircle</code>	1653
<code>\pgfpoint</code>	1654
<code>\pgfpointanchor</code>	1160, 1162
<code>\pgfusepath</code>	1656
<code>\phantom</code>	1676, 1682, 1688, 1694, 1700
<code>\pNiceArray</code>	2149, 2180
<code>\pNiceMatrix</code>	2131
prg commands:	
<code>\prg_do_nothing:</code>	106, 115, 1804
<code>\prg_replicate:nn</code>	1729, 1742
<code>\prg_return_false:</code> ..	1152, 1155, 1167, 1344
<code>\prg_return_true:</code>	1168, 1341
<code>\prg_set_conditional:Npnn</code>	1148, 1331
<code>\ProcessKeysOptions</code>	2116
<code>\ProcessOptions</code>	14
<code>\ProvideDocumentCommand</code>	37
<code>\ProvidesExplPackage</code>	6
Q	
quark commands:	
<code>\q_stop</code>	2087, 2094
R	
<code>\raise</code>	42, 44, 46

`\relax` 14, 636, 689, 690
`\renewcommand` 120
`\RenewDocumentEnvironment`
 2130, 2133, 2136, 2139, 2142
`\RequirePackage` 2, 4, 5, 15, 16, 17
`\right` 769, 778, 904, 1102, 1108, 1114, 1120, 1126
`\rVert` 1126
`\rvert` 1120

S

seq commands:

`\seq_count:N` 2120
`\seq_gclear_new:N` 680, 681, 682, 1213
`\seq_gput_left:Nn` 207, 1343, 1714, 1716
`\seq_gput_right:Nn` 1670
`\seq_if_in:NnTF` . 205, 1154, 1340, 1981, 1990
`\seq_new:N` 51
`\seq_pop:NN` 2122, 2123
`\seq_use:Nnnn` 341

skip commands:

`\skip_horizontal:n` 703, 805, 806, 813, 814,
 860, 861, 898, 900, 913, 914, 964, 971, 998,
 1001, 1084, 1085, 1086, 1093, 1094, 1095, 1809
`\skip_vertical:n` 594, 895, 902
`\c_zero_skip` 641, 645

str commands:

`\c_colon_str` 266
`\str_if_empty:NnTF` 489, 543,
 610, 739, 956, 1023, 1137, 1203, 2076, 2112
`\str_if_eq:nnTF` 199, 253, 867, 876
`\str_new:N` 142, 150, 151
`\str_set:Nn` 143, 204, 345, 346, 347, 1087
`\str_set_eq:NN` 208
`\l_tmpa_str` 204, 205, 207, 208

T

`\tabskip` 641, 645

\TeX and $\LaTeX_2\epsilon$ commands:

`\@acol` 575
`\@acoll` 573
`\@acolr` 574
`\@addtopreamble` 731
`\@array@array` 577
`\@arrayacol` 573, 574, 575
`\@arrayrule` 731
`\@currenvir` 69, 396, 1180, 1876
`\@halignto` 576
`\@height` 593
`\@ifclassloaded` 27, 30
`\@ifnextchar` 688
`\@ifpackageloaded` 76, 96
`\@mainaux`
 1131, 1132, 1139, 1145, 1197, 1198,
 1205, 1211, 1938, 1941, 1947, 2049, 2050, 2055
`\@temptokena` 105, 108, 122, 124
`\c@MaxMatrixCols` 1089
`\CT@everycr` 634
`\CT@row@color` 636
`\NC@find` 106, 129
`\NC@find@W` 690
`\NC@find@w` 689
`\NC@rewrite@S` 107, 120

`\new@ifnextchar` 688
`\p@` 42, 44, 46
`\pgf@x` . 1161, 1163, 1358, 1364, 1794, 1797,
 1848, 1852, 1861, 1864, 1984, 1993, 2043, 2047
`\pgf@y`
 1359, 1365, 1795, 1798, 1849, 1853, 1980, 1989
`\pgfutil@firstofone`
 1354, 1360, 1793, 1796,
 1846, 1850, 1859, 1862, 1976, 1986, 2041, 2044
`\tikz@library@external@loaded`
 724, 1075, 1188
`\tikz@parse@node` 1354, 1360, 1793, 1796,
 1846, 1850, 1859, 1862, 1976, 1986, 2041, 2044

tex commands:

`\tex_the:D` 124
`\tikz` 475, 524, 529, 535, 940, 1007
`\tikzset` 725, 1076, 1189, 1235, 1998, 2036

tl commands:

`\c_empty_tl` 211
`\tl_const:Nn` 917, 974
`\tl_count:n` 261
`\tl_gclear:N` 1234, 1246
`\tl_gclear_new:N` 727, 1078
`\tl_gput_right:Nn` 561, 708, 1735
`\tl_gset:Nn` 108, 112, 114
`\tl_gset_eq:NN` 624
`\tl_if_empty:NnTF` 1224
`\tl_item:Nn` 111, 112, 114
`\tl_new:N` 110, 113
`\tl_put_left:Nn` 787, 792
`\tl_put_right:Nn` 796, 801
`\tl_set:Nn` 111, 262, 785, 1158, 1333
`\tl_use:N` 270, 275, 309, 357, 395
`\g_tmpa_tl` 108, 111, 114
`\l_tmpa_tl`
 .. 111, 112, 785, 787, 792, 796, 801, 808,
 1158, 1160, 1162, 1333, 1340, 1343, 2122, 2124
`\l_tmpb_tl` 2123, 2124

token commands:

`\token_to_str:N` 70, 276

U

use commands:

`\use:N` 563, 608, 615, 744, 751
`\usetikzlibrary` 3

V

`\vbox` 46

vbox commands:

`\vbox:n` 519
`\vcenter` 70, 769, 778, 893
`\Vdots` 663
`\vdots` 674, 1665

Vdots internal commands:

`_nm_Vdots` 663, 674, 1685

vdots internal commands:

`_nm_vdots` 1665, 1688
`\vline` 1889, 1895, 1901, 1910, 1915
`\VNiceArray` 2173, 2208
`\vNiceArray` 2167, 2201
`\VNiceMatrix` 2137
`\vNiceMatrix` 2134

Contents

1	Presentation	1
2	The environments of this extension	2
3	The continuous dotted lines	2
3.1	The option nullify-dots	4
3.2	The command <code>\Hdotsfor</code>	4
3.3	How to generate the continuous dotted lines transparently	5
4	The Tikz nodes created by nicematrix	5
5	The code-after	7
6	The environment <code>{NiceArray}</code>	7
7	The dotted lines to separate rows or columns	9
8	The width of the columns	10
9	The option <code>hlines</code>	11
10	Utilisation of the column type <code>S</code> of <code>siunitx</code>	11
11	Technical remarks	12
11.1	Diagonal lines	12
11.2	The “empty” cells	12
11.3	The option <code>exterior-arraycolsep</code>	13
11.4	The class option <code>draft</code>	13
11.5	A technical problem with the argument of <code>\\</code>	13
11.6	Obsolete environments	14
12	Examples	14
12.1	Dotted lines	14
12.2	Width of the columns	16
12.3	How to highlight cells of the matrix	16
12.4	Direct utilisation of the Tikz nodes	19
12.5	Block matrices	20
13	Implementation	21
13.1	Declaration of the package and extensions loaded	21
13.2	Technical definitions	22
13.2.1	Variables for the exterior rows and columns	23
13.2.2	The column <code>S</code> of <code>siunitx</code>	24
13.3	The options	26
13.4	Code common to <code>{NiceArrayWithDelims}</code> and <code>{NiceMatrix}</code>	32
13.5	The environment <code>{NiceArrayWithDelims}</code>	40
13.6	The environment <code>{NiceMatrix}</code> and its variants	47
13.7	Automatic width of the cells	48
13.8	How to know whether a cell is “empty”	49
13.9	After the construction of the array	49
13.10	The actual instructions for drawing the dotted line with Tikz	58
13.11	User commands available in the new environments	60
13.12	The command <code>\line</code> accessible in code-after	63
13.13	The commands to draw dotted lines to separate columns and rows	63
13.14	The vertical rules	65
13.15	The environment <code>{NiceMatrixBlock}</code>	66
13.16	The extra nodes	67

13.17	We process the options	70
13.18	Code for <code>\seq_mapthread_function:NNN</code>	71
13.19	Obsolete environments	71
14	History	73
	Index	75