

The package `nicematrix`*

F. Pantigny
 fpantigny@wanadoo.fr

September 4, 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 3.1 of `nicematrix`, at the date of 2019/09/04.

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, `|`, `>{...}`, `<{...}`, `@{...}`, `!{...}` 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}
```

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

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

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

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

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

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

⁴The precise definition of a “non-empty cell” is given below (cf. p. 13).

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

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

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

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

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

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

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

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

3.1 The option `nullify-dots`

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

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

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

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

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

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

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

⁶It's also possible to fix the width of a column by using the environment `{NiceArray}` (or one of its variants) with a column of type `w` or `W`: see p. 9

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

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

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

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

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

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

3.2 The command `\Hdotsfor`

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

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

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

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

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

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

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

3.3 How to generate the continuous dotted lines transparently

The package `nicematrix` provides an option called `transparent` for using existing code transparently in the environments `{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.

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

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. 17).

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}
```

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

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

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` (by default: 5 pt). There is also an option `margin` to fix both `left-margin` and `right-margin` to the same value.

6 The environment `{NiceArray}`

The environment `{NiceArray}` is similar to the environment `{array}`. As for `{array}`, the mandatory argument is the preamble of the array. However, for technical reasons, in this preamble, the user must use the letters L, C and R⁹ instead of l, c and r. It's possible to use the constructions `w{...}{...}`, `W{...}{...}`, `l`, `>{...}`, `<{...}`, `@{...}`, `!{...}` and `*{n}{...}` but the letters p, m and b should not be used.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).

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

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

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).

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

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

⁹The column types L, C and R are defined locally inside `{NiceArray}` with `\newcolumnntype` 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.

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

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

Remarks

- As shown in the previous example, an horizontal rule (drawn by `\hline`) doesn't extend in the exterior columns and a vertical rule (specified by a `|` in the preamble of the array) doesn't extend in the exterior rows.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. 9) 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.

```
$\begin{NiceArrayWithDelims}
    {\downarrow}{\downarrow}{CCC}[last-col]
1 & 2 & 3 & L_1 \\
4 & 5 & 6 & L_2 \\
7 & 8 & 9 & L_3 \\
\end{NiceArrayWithDelims}$
```

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`. On another note, if one really wants a vertical rule running in the first and in the last row, he should use `!\vline` instead of `|` in the preamble of the array.

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

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

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

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

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

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

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

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

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

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

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

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)
```

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

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 \\
 $\end{pNiceMatrix}$ 

```

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

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

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 \\
 $\end{pNiceMatrix}$ 

```

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

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 Block matrices

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

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

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

```

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

```

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

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

```

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

```

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

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

10 The option hlines

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

```

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

```

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

11 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 the package `dcolum` are not supported by `nicematrix`.

```

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

```

$$\left(\begin{array}{c|ccc} C_1 & \cdots & C_n & \\ \hline 2.3 & 0 & \cdots & 0 \\ 12.4 & \vdots & & \vdots \\ 1.45 & \vdots & & \vdots \\ 7.2 & 0 & \cdots & 0 \end{array} \right)$$

12 Technical remarks

12.1 Intersections of dotted lines

Since the version 3.1 of `nicematrix`, the dotted lines created by `\Cdots`, `\Ldots`, `\Vdots`, etc. can't intersect.¹³

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

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

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

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

12.2 Diagonal lines

By default, all the diagonal lines¹⁴ of a same array are “parallelized”. That means that the first diagonal line is drawn and, then, the other lines are drawn parallel to the first one (by rotation around the left-most extremity of the line). That's why the position of the instructions `\Ddots` in the array can have a marked effect on the final result.

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

Example with parallelization (default):

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

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

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

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

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

The same example without parallelization:

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

¹³Of the contrary, dotted lines created by `\hdottedline`, the letter “:” in the preamble of the array and the command `\line` in the `code-after` can have intersections with other dotted lines.

¹⁴We speak of the lines created by `\Ddots` and not the lines created by a command `\line` in `code-after`.

12.3 The “empty” cells

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

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

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

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

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

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

12.4 The option `exterior-arraycolsep`

The environment `{array}` inserts an horizontal space equal to `\arraycolsep` before and after each column. In particular, there is a space equal to `\arraycolsep` before and after the array. This feature of the environment `{array}` was probably not a good idea.¹⁵

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`.

12.5 The class option `draft`

The package `nicematrix` is rather slow when drawing the dotted lines (generated by `\Cdots`, `\Ldots`, `\Ddots`, etc. but also by `\hdottedline` or the specifier `:`).¹⁶

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

¹⁵In 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@{}}`.

¹⁶The main reason is that we want dotted lines with round dots (and not square dots) with the same space on both extremities of the lines. To achieve this goal, we have to construct our own system of dotted lines.

12.6 A technical problem with the argument of `\`

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

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

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

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

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

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

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

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

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

12.7 Obsolete environments

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

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

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

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

13 Examples

13.1 Dotted lines

A tridiagonal matrix:

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

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

A permutation matrix:

```

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

```

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

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

```

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

An example with `\Hdotsfor`:

```

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

```

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

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

The figure consists of two separate graphs. The left graph, representing G_0 , has vertices labeled a_0, a_1, \dots, a_p arranged in a vertical column on the left. Edges connect a_i to a_{i+1} for $i=0, \dots, p-1$, and each a_i is also connected to a_p . The right graph, representing G , has vertices labeled b_0, b_1, \dots, b_q arranged in a vertical column on the right. Edges connect b_i to b_{i+1} for $i=0, \dots, q-1$, and each b_i is also connected to b_q . The two graphs are separated by a vertical dotted line.

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

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

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

```

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

```

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

13.3 How to highlight cells of the matrix

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

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

```

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

```

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

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

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

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

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

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

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

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

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

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

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

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

```

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

```

We obtain the following matrix.

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

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

```

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

```

We obtain the following matrix.

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

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

```

\begin{pNiceArray}{>{\strut}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}

```

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

13.4 Direct utilisation of the Tikz nodes

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

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

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

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

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

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

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

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

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

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

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

14 Implementation

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

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

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

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

14.1 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.¹⁷

```

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

```

¹⁷cf. tex.stackexchange.com/questions/57424/using-of-usetikzlibrary-in-an-expl3-package-fails

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

We load `array` and `amsmath`.

```

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

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

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

```

14.2 Technical definitions

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

```

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

34 \bool_if:NT \c__nm_draft_bool
35   { \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 `mathtdots`), we don't define it again.

```

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

```

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.

```

48 \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.

```

49 \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`.

```
50 \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.

```
51 \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`).

```
52 \bool_new:N \l__nm_NiceArray_bool
```

```
53 \bool_new:N \g__nm_NiceArray_bool
```

```
54 \cs_new_protected:Npn \__nm_test_if_math_mode:
```

```
55 {
```

```
56   \if_mode_math: \else:
```

```
57     \__nm_fatal:n { Outside-math-mode }
```

```
58   \fi:
```

```
59 }
```

Consider the following code:

```
$\begin{pNiceMatrix}
```

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

```
d & e & \Vdots \\\
```

```
f & \Cdots \\\
```

```
g & h & i \\\
```

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

First, the dotted line created by the `\Vdots` will be drawn. The implicit cell in position 2-3 will be considered as “dotted”. Then, we will have to draw the dotted line specified by the `\Cdots`; the final extremity of that line will be exactly in position 2-3 and, for that new second line, it should be considered as a *closed* extremity (since it is dotted). However, we don't have the (normal) Tikz node of that node (since it's an implicit cell): we can't draw such a line. That's why that dotted line will be said *impossible* and an error will be raised.¹⁸

```
60 \bool_new:N \l__nm_impossible_line_bool
```

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

```
61 \bool_new:N \c__nm_colortbl_loaded_bool
```

```
62 \AtBeginDocument
```

```
63 {
```

```
64   \@ifpackageloaded { colortbl }
```

```
65   {
```

```
66     \bool_set_true:N \c__nm_colortbl_loaded_bool
```

```
67     \cs_set_protected:Npn \__nm_vline_i: { { \CT@arc@ \vline } }
```

```
68   }
```

```
69   { }
```

```
70 }
```

¹⁸Of course, the user should solve the problem by adding the lacking ampersands.

14.2.1 Variables for the exterior rows and columns

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

- **First row**

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

```
71 \int_new:N \l__nm_first_row_int
72 \int_set:Nn \l__nm_first_row_int 1
73 \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.

```
74 \int_new:N \l__nm_first_col_int
75 \int_set:Nn \l__nm_first_col_int 1
76 \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`).

```
77 \int_new:N \l__nm_last_row_int
78 \int_set:Nn \l__nm_last_row_int { -2 }
79 \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”.¹⁹

```
80 \bool_new:N \l__nm_last_row_without_value_bool
81 \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.

```
82 \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.

```
83 \bool_new:N \g__nm_last_col_found_bool
```

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

¹⁹We 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.

14.2.2 The column S of siunitx

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

```

84 \bool_new:N \c__nm_siunitx_loaded_bool
85 \AtBeginDocument
86 {
87   \ifpackageloaded { siunitx }
88     { \bool_set_true:N \c__nm_siunitx_loaded_bool }
89     { }
90 }
```

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

```

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

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

```

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

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

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

```

91 \cs_set_protected:Npn \__nm_adapt_S_column:
92 {
```

In the preamble of the LaTeX document, the boolean `\c__nm_siunitx_loaded_bool` won't be known. That's why we test the existence of `\c__nm_siunitx_loaded_bool` and not its value.²⁰

```

93   \bool_if:NT \c__nm_siunitx_loaded_bool
94   {
95     \group_begin:
96     \@temptokena = { }
```

²⁰Indeed, `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`).

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

```

97      \cs_set_eq:NN \NC@find \prg_do_nothing:
98      \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`).

```

99      \tl_gset:NV \g_tmpa_tl \@temptokena
100     \group_end:
101     \tl_new:N \c__nm_table_collect_begin_tl
102     \tl_set:Nx \l_tmpa_tl { \tl_item:Nn \g_tmpa_tl 2 }
103     \tl_gset:Nx \c__nm_table_collect_begin_tl { \tl_item:Nn \l_tmpa_tl 1 }
104     \tl_new:N \c__nm_table_print_tl
105     \tl_gset:Nx \c__nm_table_print_tl { \tl_item:Nn \g_tmpa_tl { -1 } }

```

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

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

```

106     \cs_gset_eq:NN \__nm_adapt_S_column: \prg_do_nothing:
107   }
108 }

```

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).

```

109 \cs_new_protected:Npn \__nm_renew_NC@rewrite@S:
110 {
111   \renewcommand*{\NC@rewrite@S}[1] []
112   {
113     \@temptokena \exp_after:wN
114     {
115       \tex_the:D \@temptokena
116       > { \__nm_Cell: \c__nm_table_collect_begin_tl S {##1} }
117       c
118       < { \c__nm_table_print_tl \__nm_end_Cell: }
119     }
120     \NC@find
121   }
122 }

```

14.3 The options

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

```

123 \str_new:N \l__nm_pos_env_str
124 \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`).

```

125 \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`.

```

126 \bool_new:N \l__nm_parallelize_diags_bool
127 \bool_set_true:N \l__nm_parallelize_diags_bool

```

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

```
128 \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.).

```
129 \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).

```
130 \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.

```
131 \str_new:N \l__nm_name_str
```

```
132 \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.

```
133 \bool_new:N \l__nm_extra_nodes_bool
```

```
134 \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`.

```
135 \dim_new:N \l__nm_left_margin_dim
```

```
136 \dim_new:N \l__nm_right_margin_dim
```

```
137 \dim_new:N \g__nm_left_margin_dim
```

```
138 \dim_new:N \g__nm_right_margin_dim
```

```
139 \dim_new:N \g__nm_width_last_col_dim
```

```
140 \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`.

```
141 \dim_new:N \l__nm_extra_left_margin_dim
```

```
142 \dim_new:N \l__nm_extra_right_margin_dim
```

```
143 \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.

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

```
145 {
```

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

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

```
148   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`.

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

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

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

```
152   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.²¹

```

153   create-extra-nodes .bool_set:N = \l__nm_extra_nodes_bool ,
154   create-extra-nodes .default:n = true,
155   left-margin .dim_set:N = \l__nm_left_margin_dim ,
156   left-margin .default:n = \arraycolsep ,
157   right-margin .dim_set:N = \l__nm_right_margin_dim ,
158   right-margin .default:n = \arraycolsep ,
159   margin .meta:n = { left-margin = #1 , right-margin = #1 } ,
160   margin .default:n = \arraycolsep ,
161   extra-left-margin .dim_set:N = \l__nm_extra_left_margin_dim ,
162   extra-right-margin .dim_set:N = \l__nm_extra_right_margin_dim ,
163   extra-margin .meta:n =
164     { extra-left-margin = #1 , extra-right-margin = #1 } ,
165 }

```

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

```

166 \keys_define:nn { NiceMatrix / Code }
167 {
168   code-for-first-col .tl_set:N = \l__nm_code_for_first_col_tl ,
169   code-for-first-col .value_required:n = true ,
170   code-for-last-col .tl_set:N = \l__nm_code_for_last_col_tl ,
171   code-for-last-col .value_required:n = true ,
172   code-for-first-row .tl_set:N = \l__nm_code_for_first_row_tl ,
173   code-for-first-row .value_required:n = true ,
174   code-for-last-row .tl_set:N = \l__nm_code_for_last_row_tl ,
175   code-for-last-row .value_required:n = true ,
176 }

```

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

```

177 \keys_define:nn { NiceMatrix / Env }
178 {
179   columns-width .code:n =
180     \str_if_eq:nnTF { #1 } { auto }
181     { \bool_set_true:N \l__nm_auto_columns_width_bool }
182     { \dim_set:Nn \l__nm_columns_width_dim { #1 } } ,
183   columns-width .value_required:n = true ,
184   name .code:n =
185     \str_set:Nn \l_tmpa_str { #1 }
186     \seq_if_in:NVTF \g__nm_names_seq \l_tmpa_str
187     { \__nm_error:nn { Duplicate-name } { #1 } }
188     { \seq_gput_left:NV \g__nm_names_seq \l_tmpa_str }
189     \str_set_eq:NN \l__nm_name_str \l_tmpa_str ,
190   name .value_required:n = true ,
191   code-after .tl_gset:N = \g__nm_code_after_tl ,
192   code-after .initial:n = \c_empty_tl ,
193   code-after .value_required:n = true ,
194 }

```

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

```

195 \keys_define:nn { NiceMatrix }
196 {
197   NiceMatrixOptions .inherit:n =
198     {
199     NiceMatrix / Global ,

```

²¹In fact, we should not because, if there is a `w` node, the `w` node is used instead of the “medium” node.

```

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

```

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

```

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

```

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

```

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

```

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

```

231   exterior-arraycolsep .bool_set:N = \l__nm_exterior_arraycolsep_bool ,
232   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.

```

233   columns-width .code:n =
234     \str_if_eq:nnTF { #1 } { auto }
235     { \__nm_error:n { Option-auto-for-columns-width } }
236     { \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.

```

237   allow-duplicate-names .code:n =
238     \__nm_msg_redirect_name:nn { Duplicate-name } { none } ,
239   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`).

```

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

248   unknown .code:n = \__nm_error:n { Unknown~key~for~NiceMatrixOptions } }

```

`\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.

```

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

```

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

```

251 \keys_define:nn { NiceMatrix / NiceMatrix }
252 { unknown .code:n = \__nm_error:n { Unknown~option~for~NiceMatrix } }

```

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

```

253 \keys_define:nn { NiceMatrix / NiceArray }
254 {

```

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

```

255   c .code:n = \str_set:Nn \l__nm_pos_env_str c ,
256   t .code:n = \str_set:Nn \l__nm_pos_env_str t ,
257   b .code:n = \str_set:Nn \l__nm_pos_env_str b ,
258   first-col .code:n = \int_zero:N \l__nm_first_col_int ,
259   last-col .bool_set:N = \l__nm_last_col_bool ,
260   first-row .code:n = \int_zero:N \l__nm_first_row_int ,
261   last-row .int_set:N = \l__nm_last_row_int ,
262   last-row .default:n = -1 ,
263   unknown .code:n = \__nm_error:n { Unknown~option~for~NiceArray }
264 }

265 \keys_define:nn { NiceMatrix / pNiceArray }
266 {
267   first-col .code:n = \int_zero:N \l__nm_first_col_int ,
268   last-col .bool_set:N = \l__nm_last_col_bool ,
269   first-row .code:n = \int_zero:N \l__nm_first_row_int ,
270   last-row .int_set:N = \l__nm_last_row_int ,
271   last-row .default:n = -1 ,
272   unknown .code:n = \__nm_error:n { Unknown~option~for~pNiceArray }
273 }

```

14.4 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}`).

```
274 \cs_new_protected:Nn \g__nm_Cell:
275 {
```

We increment `\g__nm_col_int`, which is the counter of the columns.

```
276 \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.

```
277 \int_compare:nNnT \g__nm_col_int = \c_one_int
278 {
279   \int_compare:nNnT \l__nm_first_col_int = \c_one_int
280     \__nm_begin_of_row:
281 }
282 \int_gset:Nn \g__nm_col_total_int
283 { \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).

```
284 \hbox_set:Nw \l_tmpa_box
285 \c_math_toggle_token
286 \int_compare:nNnTF \g__nm_row_int = \c_zero_int
287   \l__nm_code_for_first_row_tl
288   {
289     \int_compare:nNnT \g__nm_row_int = \l__nm_last_row_int
290       \l__nm_code_for_last_row_tl
291   }
292 }
```

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.

```
293 \cs_new_protected:Nn \__nm_begin_of_row:
294 {
295   \int_gincr:N \g__nm_row_int
296   \dim_gset_eq:NN \g__nm_dp_ante_last_row_dim \g__nm_dp_last_row_dim
297   \dim_gzero:N \g__nm_dp_last_row_dim
298   \dim_gzero:N \g__nm_ht_last_row_dim
299 }
```

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.

```
300 \cs_new_protected:Npn \__nm_actualization_for_first_and_last_row:
301 {
302   \int_compare:nNnT \g__nm_row_int = \c_zero_int
303   {
304     \dim_gset:Nn \g__nm_dp_row_zero_dim
305       { \dim_max:nn \g__nm_dp_row_zero_dim { \box_dp:N \l_tmpa_box } }
306     \dim_gset:Nn \g__nm_ht_row_zero_dim
307       { \dim_max:nn \g__nm_ht_row_zero_dim { \box_ht:N \l_tmpa_box } }
308   }
309   \int_compare:nNnT \g__nm_row_int = \c_one_int
310   {
311     \dim_gset:Nn \g__nm_ht_row_one_dim
312       { \dim_max:nn \g__nm_ht_row_one_dim { \box_ht:N \l_tmpa_box } }
313   }
```

```

314 \dim_gset:Nn \g__nm_ht_last_row_dim
315 { \dim_max:nn \g__nm_ht_last_row_dim { \box_ht:N \l_tmpa_box } }
316 \dim_gset:Nn \g__nm_dp_last_row_dim
317 { \dim_max:nn \g__nm_dp_last_row_dim { \box_dp:N \l_tmpa_box } }
318 }
319 \cs_new_protected:Nn \__nm_end_Cell:
320 {
321 \c_math_toggle_token
322 \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”).

```

323 \dim_gset:Nn \g__nm_max_cell_width_dim
324 { \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”.

```

325 \__nm_actualization_for_first_and_last_row:

```

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

```

326 \tikz
327 [
328 remember~picture ,
329 inner~sep = \c_zero_dim ,
330 minimum~width = \c_zero_dim ,
331 baseline
332 ]
333 \node
334 [
335 anchor = base ,
336 name = nm - \int_use:N \g__nm_env_int -
337 \int_use:N \g__nm_row_int -
338 \int_use:N \g__nm_col_int ,
339 alias =
340 \str_if_empty:NF \l__nm_name_str
341 {
342 \l__nm_name_str -
343 \int_use:N \g__nm_row_int -
344 \int_use:N \g__nm_col_int
345 }
346 ]
347 \bgroup
348 \box_use:N \l_tmpa_box
349 \egroup ;
350 }
351 \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.

```

352 \cs_new_protected:Nn \__nm_renewcolumnntype:nn
353 {
354 \newcolumnntype #1 [ 2 ]
355 {
356 > {
357 \hbox_set:Nw \l_tmpa_box
358 \__nm_Cell:
359 }
360 c
361 < {
362 \__nm_end_Cell:

```

```

363 \hbox_set_end:
364 #2
365 \hbox_set:Nn \l_tmpb_box
366 { \makebox [ ##2 ] [ ##1 ] { \box_use:N \l_tmpa_box } }
367 \dim_set:Nn \l_tmpa_dim { \box_dp:N \l_tmpb_box }
368 \box_move_down:nn \l_tmpa_dim
369 {
370   \vbox:n
371   {
372     \hbox_to_wd:nn { \box_wd:N \l_tmpb_box }
373     {
374       \hfil
375       \tikz [ remember~picture , overlay ]
376       \coordinate ( __nm~north~east ) ;
377     }
378     \hbox:n
379     {
380       \tikz [ remember~picture , overlay ]
381       \coordinate ( __nm~south~west ) ;
382       \box_move_up:nn \l_tmpa_dim { \box_use:N \l_tmpb_box }
383     }
384   }
385 }

```

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

```

386 \tikz [ remember~picture , overlay ]
387 \node
388 [
389   node~contents = { } ,
390   name = nm - \int_use:N \g__nm_env_int -
391             \int_use:N \g__nm_row_int -
392             \int_use:N \g__nm_col_int - w,
393   alias =
394     \str_if_empty:NF \l__nm_name_str
395     {
396       \l__nm_name_str -
397       \int_use:N \g__nm_row_int -
398       \int_use:N \g__nm_col_int - w
399     } ,
400   inner~sep = \c_zero_dim ,
401   fit = ( __nm~south~west ) ( __nm~north~east )
402 ]
403 ;
404 }
405 }
406 }

```

The argument of the following command `\@@_instruction_of_type:n` defined below is the type of the instruction (`Cdots`, `Vdots`, `Ddots`, etc.). This command writes in the correspondint `\g_@@_type_lines_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 & \Cdots & & \\
\end{pNiceMatrix}

```

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

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

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

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

```
407 \bool_if:NTF \c_nm_draft_bool
408 { \cs_set_protected:Npn \__nm_instruction_of_type:n #1 { } }
409 {
410   \cs_new_protected:Npn \__nm_instruction_of_type:n #1
411   {
```

It's important to use a `\tl_gput_right:cx` and not a `\tl_gput_left:cx` because we want the `\Ddots` lines to be drawn in the order of appearance in the array (for parallelisation).

```
412     \tl_gput_right:cx
413     { g_nm_ #1 _ lines _ tl }
414     {
415       \use:c { __nm _ draw _ #1 : nn }
416       { \int_use:N \g_nm_row_int }
417       { \int_use:N \g_nm_col_int }
418     }
419   }
420 }
```

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`.

```
421 \cs_new_protected:Npn \__nm_array:
422 {
423   \bool_if:NTF \c_nm_revtex_bool
424   {
425     \cs_set_eq:NN \@acoll \@arrayacol
426     \cs_set_eq:NN \@acolr \@arrayacol
427     \cs_set_eq:NN \@acol \@arrayacol
428     \cs_set:Npn \@halignto { }
429     \@array@array
430   }
431   \array
```

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

```
432 [ \l_nm_pos_env_str ]
433 }
```

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

```
434 \cs_new:Npn \__nm_everycr:
435 { \noalign { \__nm_everycr_i: } }
436 \cs_new_protected:Npn \__nm_everycr_i:
437 {
438   \int_gzero:N \g_nm_col_int
439   \bool_if:NT \l_nm_hlines_bool
440   {
```

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.

```
441     \int_compare:nNnT \g_nm_row_int > { -1 }
442     {
443       \int_compare:nNnF \g_nm_row_int = \g_nm_last_row_int
444       {
445         \hrule \@height \arrayrulewidth
446         \skip_vertical:n { - \arrayrulewidth }
447       }
448     }
449   }
450 }
```

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.

```
451 \cs_new_protected:Npn \__nm_pre_array:
452 {
```

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}`.

```
453   \bool_if:NT \l__nm_auto_columns_width_bool
454   {
455     \group_insert_after:N \__nm_write_max_cell_width:
456     \cs_if_free:cTF { __nm_max_cell_width_ \int_use:N \g__nm_env_int }
457     { \dim_zero:N \l__nm_columns_width_dim }
458     {
459       \dim_set:Nx \l__nm_columns_width_dim
460       { \use:c { __nm_max_cell_width_ \int_use:N \g__nm_env_int } }
461     }
462   }
```

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)).

```
462   \str_if_empty:NF \l__nm_name_str
463   {
464     \cs_if_free:cF { __nm_max_cell_width_ \l__nm_name_str }
465     {
466       \dim_set:Nx \l__nm_columns_width_dim
467       { \use:c { __nm_max_cell_width_ \l__nm_name_str } }
468     }
469   }
470 }
```

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.

```
471   \bool_gset_eq:NN \g__nm_extra_nodes_bool \l__nm_extra_nodes_bool
472   \dim_gset_eq:NN \g__nm_left_margin_dim \l__nm_left_margin_dim
473   \dim_gset_eq:NN \g__nm_right_margin_dim \l__nm_right_margin_dim
474   \dim_gset_eq:NN \g__nm_extra_right_margin_dim \l__nm_extra_right_margin_dim
475   \int_gset_eq:NN \g__nm_last_row_int \l__nm_last_row_int
476   \tl_gset_eq:NN \g__nm_name_str \l__nm_name_str
477   \int_gset_eq:NN \g__nm_first_row_int \l__nm_first_row_int
478   \int_gset_eq:NN \g__nm_first_col_int \l__nm_first_col_int
479   \bool_gset_eq:NN \g__nm_NiceArray_bool \l__nm_NiceArray_bool
480   \bool_gset_eq:NN \g__nm_last_row_without_value_bool
481   \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.²²

```
482   \cs_set:Npn \ialign
483   {
484     \bool_if:NTF \c__nm_colortbl_loaded_bool
485     {
486       \CT@everycr
487     }
```

²²With 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}`)

```

488         \noalign { \cs_gset_eq:NN \CT@row@color \prg_do_nothing: }
489         \__nm_everycr:
490     }
491 }
492 { \everycr { \__nm_everycr: } }
493 \tabskip = \c_zero_skip
494 \cs_set:Npn \ialign
495     {
496         \everycr { }
497         \tabskip = \c_zero_skip
498         \halign
499     }
500 \halign
501 }

```

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}.

```

502 \dim_compare:nNnTF \l__nm_columns_width_dim = \c_zero_dim
503 {
504     \newcolumnntype L { > \__nm_Cell: l < \__nm_end_Cell: }
505     \newcolumnntype C { > \__nm_Cell: c < \__nm_end_Cell: }
506     \newcolumnntype R { > \__nm_Cell: r < \__nm_end_Cell: }
507 }

```

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.

```

508 {
509     \newcolumnntype L { w l { \dim_use:N \l__nm_columns_width_dim } }
510     \newcolumnntype C { w c { \dim_use:N \l__nm_columns_width_dim } }
511     \newcolumnntype R { w r { \dim_use:N \l__nm_columns_width_dim } }
512 }
513 \cs_set_eq:NN \Ldots \__nm_Ldots
514 \cs_set_eq:NN \Cdots \__nm_Cdots
515 \cs_set_eq:NN \Vdots \__nm_Vdots
516 \cs_set_eq:NN \Ddots \__nm_Ddots
517 \cs_set_eq:NN \Iddots \__nm_Iddots
518 \cs_set_eq:NN \hdottedline \__nm_hdottedline:
519 \cs_set_eq:NN \Hspace \__nm_Hspace:
520 \cs_set_eq:NN \Hdotsfor \__nm_Hdotsfor:
521 \cs_set_eq:NN \multicolumn \__nm_multicolumn:nnn
522 \cs_set_eq:NN \Block \__nm_Block:
523 \bool_if:NT \l__nm_renew_dots_bool
524 {
525     \cs_set_eq:NN \ldots \__nm_Ldots
526     \cs_set_eq:NN \cdots \__nm_Cdots
527     \cs_set_eq:NN \vdots \__nm_Vdots
528     \cs_set_eq:NN \ddots \__nm_Ddots
529     \cs_set_eq:NN \iddots \__nm_Iddots
530     \cs_set_eq:NN \dots \__nm_Ldots
531     \cs_set_eq:NN \hdotsfor \__nm_Hdotsfor:
532 }

```

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

```

533 \seq_gclear_new:N \g__nm_multicolumn_cells_seq
534 \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).

```

535 \int_gzero_new:N \g__nm_row_int
536 \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 of rows and `\g_@@_row_total_int` will be the number of rows excepted the last row (if `\l_@@_last_row_bool` has been raised with the option `last-row`).

```
537 \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.

```
538 \int_gzero_new:N \g__nm_col_int
539 \int_gzero_new:N \g__nm_col_total_int
540 \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:`.

```
541 \cs_set_eq:NN \NC@find@w \relax
542 \cs_set_eq:NN \NC@find@W \relax
543 \__nm_renewcolumnntype:nn w { }
544 \__nm_renewcolumnntype:nn W { \cs_set_eq:NN \hss \hfil }
545 \dim_gzero_new:N \g__nm_dp_row_zero_dim
546 \dim_gzero_new:N \g__nm_ht_row_zero_dim
547 \dim_gzero_new:N \g__nm_ht_row_one_dim
548 \dim_gzero_new:N \g__nm_dp_ante_last_row_dim
549 \dim_gzero_new:N \g__nm_ht_last_row_dim
550 \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`. We rescan this string (which is always of length 1) in particular for the case where `pdflatex` is used with `french-babel` (the colon is activated by `french-babel` at the beginning of the document).

```
551 \tl_set_rescan:Nno
552 \l__nm_letter_for_dotted_lines_str { } \l__nm_letter_for_dotted_lines_str
553 \exp_args:NV \newcolumnntype \l__nm_letter_for_dotted_lines_str
554 {
555   !
556   {
557     \skip_horizontal:n { 0.53 pt }
558     \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.

```
559 \int_compare:nNnT \g__nm_col_int > \g__nm_last_vdotted_col_int
560 {
561   \int_gset_eq:NN \g__nm_last_vdotted_col_int \g__nm_col_int
562   \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`.

```

563         { \__nm_vdottedline:n { \int_use:N \g__nm_col_int } }
564     }
565 }
566 }
567 \int_gzero_new:N \g__nm_last_vdotted_col_int
568 \bool_if:NT \c__nm_siunitx_loaded_bool \__nm_renew_NC@rewrite@S:
569 \int_gset:Nn \g__nm_last_vdotted_col_int { -1 }
570 \bool_gset_false:N \g__nm_last_col_found_bool

```

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

```

571 \tl_gclear_new:N \g__nm_Cdots_lines_tl
572 \tl_gclear_new:N \g__nm_Ldots_lines_tl
573 \tl_gclear_new:N \g__nm_Vdots_lines_tl
574 \tl_gclear_new:N \g__nm_Ddots_lines_tl
575 \tl_gclear_new:N \g__nm_Iddots_lines_tl
576 \tl_gclear_new:N \g__nm_Hdotsfor_lines_tl
577 }

```

14.5 The environment `{NiceArrayWithDelims}`

```

578 \NewDocumentEnvironment { NiceArrayWithDelims } { m m O { } m ! O { } }
579 {
580     \__nm_adapt_S_column:
581     \__nm_test_if_math_mode:
582     \bool_if:NT \l__nm_in_env_bool { \__nm_fatal:n { Yet-in-env } }
583     \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).

```

584     \cs_if_exist:NT \tikz@library@external@loaded
585     { \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}`.

```

586     \group_insert_after:N \__nm_after_array:

```

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

```

587     \int_gincr:N \g__nm_env_int
588     \bool_if:NF \l__nm_block_auto_columns_width_bool
589     { \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.

```

590     \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`.

```

591     \bool_if:NFT \l__nm_NiceArray_bool
592     { \keys_set:nn { NiceMatrix / NiceArray } }
593     { \keys_set:nn { NiceMatrix / pNiceArray } }
594     { #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).

```

595     \int_compare:nNnT \l__nm_last_row_int = { -1 }
596     {
597         \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.

```

598 \str_if_empty:NTF \g__nm_name_str
599 {
600     \cs_if_exist:cT { __nm_last_row_ \int_use:N \g__nm_env_int }
601     {
602         \int_set:Nn \l__nm_last_row_int
603         { \use:c { __nm_last_row_ \int_use:N \g__nm_env_int } }
604     }
605 }
606 {
607     \cs_if_exist:cT { __nm_last_row_ \g__nm_name_str }
608     {
609         \int_set:Nn \l__nm_last_row_int
610         { \use:c { __nm_last_row_ \g__nm_name_str } }
611     }
612 }
613 }

```

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

```

614 \__nm_pre_array:

```

We compute the width of the two delimiters.

```

615 \dim_zero_new:N \g__nm_left_delim_dim
616 \dim_zero_new:N \g__nm_right_delim_dim
617 \bool_if:NTF \l__nm_NiceArray_bool
618 {
619     \dim_gset:Nn \g__nm_left_delim_dim { 2 \arraycolsep }
620     \dim_gset:Nn \g__nm_right_delim_dim { 2 \arraycolsep }
621 }
622 {
623     \group_begin:
624     \dim_set_eq:NN \nulldelimiterspace \c_zero_dim
625     \hbox_set:Nn \l_tmpa_box
626     {
627         \c_math_toggle_token
628         \left #1 \vcenter to 3 cm { } \right.
629         \c_math_toggle_token
630     }
631     \dim_gset:Nn \g__nm_left_delim_dim { \box_wd:N \l_tmpa_box }
632     \hbox_set:Nn \l_tmpa_box
633     {
634         \dim_set_eq:NN \nulldelimiterspace \c_zero_dim
635         \c_math_toggle_token
636         \left. \vcenter to 3 cm { } \right #2
637         \c_math_toggle_token
638     }
639     \dim_gset:Nn \g__nm_right_delim_dim { \box_wd:N \l_tmpa_box }
640     \group_end:
641 }
642 }

```

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}).

```

643 \box_clear_new:N \l__nm_the_array_box

```

We construct the preamble of the array in \l_tmpa_tl.

```

644 \tl_set:Nn \l_tmpa_tl { #4 }
645 \int_compare:nNnTF \l__nm_first_col_int = \c_zero_int
646 { \tl_put_left:NV \l_tmpa_tl \c__nm_preamble_first_col_tl }
647 {
648     \bool_if:NT \l__nm_NiceArray_bool
649     {
650         \bool_if:NF \l__nm_exterior_arraycolsep_bool

```

```

651         { \tl_put_left:Nn \l_tmpa_tl { @ { } } }
652     }
653 }
654 \bool_if:NTF \l__nm_last_col_bool
655 { \tl_put_right:NV \l_tmpa_tl \c__nm_preamble_last_col_tl }
656 {
657     \bool_if:NT \l__nm_NiceArray_bool
658     {
659         \bool_if:NF \l__nm_exterior_arraycolsep_bool
660         { \tl_put_right:Nn \l_tmpa_tl { @ { } } }
661     }
662 }

```

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).

```

663 \hbox_set:Nw \l__nm_the_array_box
664 \skip_horizontal:n \l__nm_left_margin_dim
665 \skip_horizontal:n \l__nm_extra_left_margin_dim
666 \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`).

```

667 \exp_args:NV \__nm_array: \l_tmpa_tl
668 }

```

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

```

669 {
670     \endarray
671     \c_math_toggle_token
672     \skip_horizontal:n \g__nm_right_margin_dim
673     \skip_horizontal:n \g__nm_extra_right_margin_dim
674     \hbox_set_end:

675     \int_compare:nNnT \g__nm_last_row_int > { -2 }
676     {
677         \bool_if:NF \g__nm_last_row_without_value_bool
678         {
679             \int_compare:nNnF \g__nm_last_row_int = \g__nm_row_int
680             {
681                 \__nm_error:n { Wrong-last-row }
682                 \int_gset_eq:NN \g__nm_last_row_int \g__nm_row_int
683             }
684         }
685     }

```

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).

```

686     \int_compare:nNnTF \l__nm_first_row_int = \c_zero_int
687     {
688         \dim_compare:nNnTF
689         { \g__nm_ht_row_one_dim + \g__nm_dp_row_zero_dim + \lineskiplimit }
690         > \baselineskip
691         {
692             \dim_set:Nn \l_tmpa_dim
693             {
694                 \g__nm_ht_row_one_dim + \g__nm_dp_row_zero_dim + \lineskip
695                 + \g__nm_ht_row_zero_dim - \g__nm_ht_row_one_dim
696             }
697         }
698         {
699             \dim_set:Nn \l_tmpa_dim
700             { \baselineskip + \g__nm_ht_row_zero_dim - \g__nm_ht_row_one_dim }
701         }

```

```

702     }
703     { \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”.²³

```

704     \int_compare:nNnTF \l__nm_last_row_int > { -2 }
705     {
706         \dim_compare:nNnTF
707         {
708             \g__nm_ht_last_row_dim + \g__nm_dp_ante_last_row_dim + \lineskiplimit
709         }
710         >
711         \baselineskip
712         {
713             \dim_set:Nn \l_tmpb_dim
714             {
715                 \g__nm_ht_last_row_dim + \g__nm_dp_ante_last_row_dim + \lineskip
716                 + \g__nm_dp_last_row_dim - \g__nm_dp_ante_last_row_dim
717             }
718         }
719         {
720             \dim_set:Nn \l_tmpb_dim
721             {
722                 \baselineskip
723                 + \g__nm_dp_last_row_dim - \g__nm_dp_ante_last_row_dim
724             }
725         }
726     }
727     { \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. 43).

```

728     \int_compare:nNnT \g__nm_first_col_int = \c_zero_int
729     {
730         \skip_horizontal:n \arraycolsep
731         \skip_horizontal:n \g__nm_width_first_col_dim
732     }

```

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}`.

```

733     \bool_if:NTF \g__nm_NiceArray_bool
734     {
735         \int_compare:nNnT \g__nm_first_row_int = \c_zero_int
736         {
737             \str_if_eq:VnTF \l__nm_pos_env_str { t }
738             {
739                 \box_move_up:nn
740                 { \l_tmpa_dim - \g__nm_ht_row_zero_dim + \g__nm_ht_row_one_dim }
741             }
742         }
743         {
744             \bool_if:NT \g__nm_last_row_int
745             {
746                 \str_if_eq:VnT \l__nm_pos_env_str { b }
747                 {
748                     \box_move_down:nn
749                     {
750                         \l_tmpb_dim

```

²³A 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`).

```

751             - \g__nm_dp_last_row_dim + \g__nm_dp_ante_last_row_dim
752         }
753     }
754 }
755 }
756 { \box_use_drop:N \l__nm_the_array_box }
757 }

```

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

```

758 {
759     \hbox_set:Nn \l_tmpa_box
760     {
761         \c_math_toggle_token
762         \left #1
763         \vcenter
764         {

```

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

```

765         \skip_vertical:n { - \l_tmpa_dim }
766         \hbox:n
767         {
768             \skip_horizontal:n { - \arraycolsep }
769             \box_use_drop:N \l__nm_the_array_box
770             \skip_horizontal:n { - \arraycolsep }
771         }

```

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

```

772         \skip_vertical:n { - \l_tmpb_dim }
773     }
774     \right #2
775     \c_math_toggle_token
776 }
777 \box_set_ht:Nn \l_tmpa_box { \box_ht:N \l_tmpa_box + \l_tmpa_dim }
778 \box_set_dp:Nn \l_tmpa_box { \box_dp:N \l_tmpa_box + \l_tmpb_dim }
779 \box_use_drop:N \l_tmpa_box
780 }

```

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. 44).

```

781 \bool_if:NT \g__nm_last_col_found_bool
782 {
783     \skip_horizontal:n \g__nm_width_last_col_dim
784     \skip_horizontal:n \arraycolsep
785 }
786 }

```

This is the end of the environment {NiceArrayWithDelims}.

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

```

787 \tl_const:Nn \c__nm_preamble_first_col_tl
788 {
789     >
790     {
791         \__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.

```

792         \hbox_set:Nw \l_tmpa_box
793         \c_math_toggle_token
794         \l__nm_code_for_first_col_tl
795     }
796     1
797     <
798     {
799         \c_math_toggle_token
800         \hbox_set_end:

```

```
801 \__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.

```
802 \dim_gset:Nn \g__nm_width_first_col_dim
803 {
804   \dim_max:nn
805   \g__nm_width_first_col_dim
806   { \box_wd:N \l_tmpa_box }
807 }
```

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

```
808 \hbox_overlap_left:n
809 {
810   \tikz
811   [
812     remember~picture ,
813     inner~sep = \c_zero_dim ,
814     minimum~width = \c_zero_dim ,
815     baseline
816   ]
817   \node
818   [
819     anchor = base ,
820     name =
821     nm -
822     \int_use:N \g__nm_env_int -
823     \int_use:N \g__nm_row_int -
824     0 ,
825     alias =
826     \str_if_empty:NF \l__nm_name_str
827     {
828       \l__nm_name_str -
829       \int_use:N \g__nm_row_int -
830       0
831     }
832   ]
833   { \box_use:N \l_tmpa_box } ;
834   \skip_horizontal:n
835   {
836     \g__nm_left_delim_dim +
837     \l__nm_left_margin_dim +
838     \l__nm_extra_left_margin_dim
839   }
840 }
841 \skip_horizontal:n { - 2 \arraycolsep }
842 }
843 }
```

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

```
844 \tl_const:Nn \c__nm_preamble_last_col_tl
845 {
846   >
847   {
```

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

```
848 \bool_gset_true:N \g__nm_last_col_found_bool
849 \int_gincr:N \g__nm_col_int
850 \int_gset:Nn \g__nm_col_total_int
851 { \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.

```
852 \hbox_set:Nw \l_tmpa_box
853 \c_math_toggle_token
854 \l__nm_code_for_last_col_tl
855 }
```

```

856     1
857     <
858     {
859         \c_math_toggle_token
860         \hbox_set_end:
861         \__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.

```

862         \dim_gset:Nn \g__nm_width_last_col_dim
863         {
864             \dim_max:nn
865             \g__nm_width_last_col_dim
866             { \box_wd:N \l_tmpa_box }
867         }
868         \skip_horizontal:n { - 2 \arraycolsep }

```

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

```

869         \hbox_overlap_right:n
870         {
871             \skip_horizontal:n
872             {
873                 \g__nm_right_delim_dim +
874                 \l__nm_right_margin_dim +
875                 \l__nm_extra_right_margin_dim
876             }
877             \tikz
878             [
879                 remember-picture ,
880                 inner~sep = \c_zero_dim ,
881                 minimum-width = \c_zero_dim ,
882                 baseline
883             ]
884             \node
885             [
886                 anchor = base ,
887                 name =
888                 nm -
889                 \int_use:N \g__nm_env_int -
890                 \int_use:N \g__nm_row_int -
891                 \int_use:N \g__nm_col_int ,
892                 alias =
893                 \str_if_empty:NF \l__nm_name_str
894                 {
895                     \l__nm_name_str -
896                     \int_use:N \g__nm_row_int -
897                     \int_use:N \g__nm_col_int
898                 }
899             ]
900             { \box_use:N \l_tmpa_box } ;
901         }
902     }
903 }

```

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.

```

904 \NewDocumentEnvironment { NiceArray } { }
905 {
906     \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).

```

907 \NiceArrayWithDelims . .

```

```

908 }
909 { \endNiceArrayWithDelims }

```

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

```

910 \NewDocumentEnvironment { pNiceArray } { }
911 {
912   \__nm_test_if_math_mode:
913   \NiceArrayWithDelims ( )
914 }
915 { \endNiceArrayWithDelims }
916 \NewDocumentEnvironment { bNiceArray } { }
917 {
918   \__nm_test_if_math_mode:
919   \NiceArrayWithDelims [ ]
920 }
921 { \endNiceArrayWithDelims }
922 \NewDocumentEnvironment { BNiceArray } { }
923 {
924   \__nm_test_if_math_mode:
925   \NiceArrayWithDelims \{ \}
926 }
927 { \endNiceArrayWithDelims }
928 \NewDocumentEnvironment { vNiceArray } { }
929 {
930   \__nm_test_if_math_mode:
931   \NiceArrayWithDelims | |
932 }
933 { \endNiceArrayWithDelims }
934 \NewDocumentEnvironment { VNiceArray } { }
935 {
936   \__nm_test_if_math_mode:
937   \NiceArrayWithDelims \| \|
938 }
939 { \endNiceArrayWithDelims }

```

14.6 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}`:

```

940 \NewDocumentEnvironment { NiceMatrix } { ! 0 { } }
941 {
942   \__nm_test_if_math_mode:
943   \bool_if:NT \l__nm_in_env_bool { \__nm_fatal:n { Yet~in~env } }
944   \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`.

```

945   \cs_if_exist:NT \tikz@library@external@loaded
946   { \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`).

```

947 \group_insert_after:N \__nm_after_array:
948 \int_gincr:N \g__nm_env_int
949 \bool_if:NF \l__nm_block_auto_columns_width_bool
950 { \dim_gzero_new:N \g__nm_max_cell_width_dim }
951 \keys_set:nn { NiceMatrix / NiceMatrix } { #1 }

```

The macro `\@@_pre_array:` is defined above (see p. 35). It is also used in `{NiceArrayWithDelims}`.

```

952 \__nm_pre_array:
953 \skip_horizontal:n { - \arraycolsep }
954 \skip_horizontal:n \l__nm_left_margin_dim
955 \skip_horizontal:n \l__nm_extra_left_margin_dim
956 \str_set:Nn \l__nm_pos_env_str c
957 \bool_set_false:N \l__nm_exterior_arraycolsep_bool
958 \__nm_array: { * \c@MaxMatrixCols C }
959 }
960 {
961 \endarray
962 \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.

```

963 \skip_horizontal:n \g__nm_right_margin_dim
964 \skip_horizontal:n \g__nm_extra_right_margin_dim
965 }

```

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

```

966 \NewDocumentEnvironment { pNiceMatrix } { }
967 {
968 \__nm_test_if_math_mode:
969 \left( \begin{NiceMatrix}
970 }
971 { \end{NiceMatrix} \right) }
972 \NewDocumentEnvironment { bNiceMatrix } { }
973 {
974 \__nm_test_if_math_mode:
975 \left[ \begin{NiceMatrix}
976 }
977 { \end{NiceMatrix} \right] }
978 \NewDocumentEnvironment { BNiceMatrix } { }
979 {
980 \__nm_test_if_math_mode:
981 \left\{ \begin{NiceMatrix}
982 }
983 { \end{NiceMatrix} \right\} }
984 \NewDocumentEnvironment { vNiceMatrix } { }
985 {
986 \__nm_test_if_math_mode:
987 \left\lvert \begin{NiceMatrix}
988 }
989 { \end{NiceMatrix} \right\rvert }
990 \NewDocumentEnvironment { VNiceMatrix } { }
991 {
992 \__nm_test_if_math_mode:
993 \left\lVert \begin{NiceMatrix}
994 }
995 { \end{NiceMatrix} \right\rVert }

```

14.7 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`.

```

996 \cs_new_protected:Nn \__nm_write_max_cell_width:
997 {
998   \bool_if:NF \l__nm_block_auto_columns_width_bool
999   {
1000     \iow_now:Nn \@mainaux \ExplSyntaxOn
1001     \iow_now:Nx \@mainaux
1002     {
1003       \cs_gset:cpn { __nm_max_cell_width_ \int_use:N \g__nm_env_int }
1004       { \dim_use:N \g__nm_max_cell_width_dim }
1005     }

```

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

```

1006   \str_if_empty:NF \g__nm_name_str
1007   {
1008     \iow_now:Nx \@mainaux
1009     {
1010       \cs_gset:cpn { __nm_max_cell_width_ \g__nm_name_str }
1011       { \dim_use:N \g__nm_max_cell_width_dim }
1012     }
1013   }
1014   \iow_now:Nn \@mainaux \ExplSyntaxOff
1015 }
1016 }

```

14.8 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.

```

1017 \prg_set_conditional:Npnn \__nm_if_not_empty_cell:nn #1 #2 { T , TF }
1018 {

```

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

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

```

1019   \bool_set_false:N \l_tmpa_bool
1020   \cs_if_exist:cTF
1021   { __nm _dotted _ \int_use:N #1 - \int_use:N #2 }
1022   \prg_return_true:
1023   {

```

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

```

1024   \cs_if_free:cTF

```

```

1025     {
1026         pgf@sh@ns@nm -
1027         \int_use:N \g__nm_env_int -
1028         \int_use:N #1 -
1029         \int_use:N #2
1030     }
1031     { \prg_return_false: }
1032     {

```

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

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

```

1033         \bool_set_false:N \l_tmpa_bool
1034         \cs_if_exist:cT
1035         { __nm _ empty _ \int_use:N #1 - \int_use:N #2 }
1036         {
1037             \int_compare:nNnT
1038             { \use:c { __nm _ empty _ \int_use:N #1 - \int_use:N #2 } }
1039             =
1040             \g__nm_env_int
1041             { \bool_set_true:N \l_tmpa_bool }
1042         }
1043         \bool_if:NTF \l_tmpa_bool
1044         \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.

```

1045         {
1046             \begin { pgfpicture }

```

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

```

1047         \tl_set:Nx \l_tmpa_tl
1048         {
1049             nm -
1050             \int_use:N \g__nm_env_int -
1051             \int_use:N #1 -
1052             \int_use:N #2
1053         }
1054         \pgfpointanchor \l_tmpa_tl { east }
1055         \dim_gset:Nn \g_tmpa_dim \pgf@x
1056         \pgfpointanchor \l_tmpa_tl { west }
1057         \dim_gset:Nn \g_tmpb_dim \pgf@x
1058         \end { pgfpicture }
1059         \dim_compare:nNnTF
1060         { \dim_abs:n { \g_tmpb_dim - \g_tmpa_dim } } < { 0.5 pt }
1061         \prg_return_false:
1062         \prg_return_true:
1063     }
1064 }
1065 }
1066 }

```

14.9 After the construction of the array

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

```

1067 \cs_new_protected:Nn \__nm_after_array:
1068 {
1069   \int_compare:nNnTF \g__nm_row_int > \c_zero_int
1070     \__nm_after_array_i:
1071     { \__nm_error:n { Zero-row } }
1072 }

```

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

```

1073 \cs_new_protected:Nn \__nm_after_array_i:
1074 {
1075   \group_begin:
1076   \cs_if_exist:NT \tikz@library@external@loaded
1077     { \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”.²⁴

```

1078   \int_gset_eq:NN \g__nm_col_int \g__nm_col_total_int
1079   \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.

```

1080   \int_gset_eq:NN \g__nm_row_total_int \g__nm_row_int
1081   \int_compare:nNnT \g__nm_last_row_int > { -1 }
1082     { \int_gsub:Nn \g__nm_row_int \c_one_int }

```

If 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.

```

1083   \bool_if:NT \g__nm_last_row_without_value_bool
1084     {
1085       \iow_now:Nn \@mainaux \ExplSyntaxOn
1086       \iow_now:Nx \@mainaux
1087       {
1088         \cs_gset:cpn { __nm_last_row_ \int_use:N \g__nm_env_int }
1089         { \int_use:N \g__nm_row_total_int }
1090       }

```

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.

```

1091       \str_if_empty:NF \g__nm_name_str
1092       {
1093         \iow_now:Nx \@mainaux
1094         {
1095           \cs_gset:cpn { __nm_last_row_ \g__nm_name_str }
1096           { \int_use:N \g__nm_row_total_int }
1097         }
1098       }
1099       \iow_now:Nn \@mainaux \ExplSyntaxOff
1100     }

```

By default, the diagonal lines will be parallelized²⁵. 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.

```

1101   \bool_if:NT \l__nm_parallelize_diags_bool
1102   {
1103     \int_zero_new:N \l__nm_ddots_int
1104     \int_zero_new:N \l__nm_iddots_int

```

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

²⁴We remind that the potential “first column” has the number 0.

²⁵It’s possible to use the option `parallelize-diags` to disable this parallelization.

```

1105     \dim_zero_new:N \l__nm_delta_x_one_dim
1106     \dim_zero_new:N \l__nm_delta_y_one_dim
1107     \dim_zero_new:N \l__nm_delta_x_two_dim
1108     \dim_zero_new:N \l__nm_delta_y_two_dim
1109 }

```

If the user has used the option `create-extra-nodes`, the “medium nodes” and “large nodes” are created. We recall that the command `\@@_create_extra_nodes:`, when used once, becomes no-op (in the current TeX group).

```

1110     \bool_if:NT \g__nm_extra_nodes_bool \__nm_create_extra_nodes:
1111     \int_zero_new:N \l__nm_initial_i_int
1112     \int_zero_new:N \l__nm_initial_j_int
1113     \int_zero_new:N \l__nm_final_i_int
1114     \int_zero_new:N \l__nm_final_j_int
1115     \bool_set_false:N \l__nm_initial_open_bool
1116     \bool_set_false:N \l__nm_final_open_bool

```

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

```

1117     \g__nm_Hdotsfor_lines_tl
1118     \g__nm_Vdots_lines_tl
1119     \g__nm_Ddots_lines_tl
1120     \g__nm_Iddots_lines_tl
1121     \g__nm_Cdots_lines_tl
1122     \g__nm_Ldots_lines_tl

```

Now, the code-after.

```

1123     \tikzset
1124     {
1125         every-picture / .style =
1126         {
1127             overlay ,
1128             remember-picture ,
1129             name-prefix = nm - \int_use:N \g__nm_env_int -
1130         }
1131     }
1132     \cs_set_eq:NN \line \__nm_line:nn
1133     \g__nm_code_after_tl
1134     \tl_gclear:N \g__nm_code_after_tl
1135     \group_end:
1136 }

```

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

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

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

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

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

- the first argument is the row of the cell where the command was issued;

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

This command computes:

- `\l_@@_initial_i_int` and `\l_@@_initial_j_int` which are the coordinates of one extremity of the line;
- `\l_@@_final_i_int` and `\l_@@_final_j_int` which are the coordinates of the other extremity of the line;
- `\l_@@_initial_open_bool` and `\l_@@_final_open_bool` to indicate whether the extremities are open or not.

```
1137 \cs_new_protected:Nn \__nm_find_extremities_of_line:nnnn
1138 {
```

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

```
1139 \cs_set:cpn { __nm _ dotted _ #1 - #2 } { }
```

Initialisation of variables.

```
1140 \int_set:Nn \l__nm_initial_i_int { #1 }
1141 \int_set:Nn \l__nm_initial_j_int { #2 }
1142 \int_set:Nn \l__nm_final_i_int { #1 }
1143 \int_set:Nn \l__nm_final_j_int { #2 }
```

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

```
1144 \bool_set_false:N \l__nm_stop_loop_bool
1145 \bool_do_until:Nn \l__nm_stop_loop_bool
1146 {
1147     \int_add:Nn \l__nm_final_i_int { #3 }
1148     \int_add:Nn \l__nm_final_j_int { #4 }
```

We test if we are still in the matrix.

```
1149     \bool_set_false:N \l__nm_final_open_bool
1150     \int_compare:nNnTF \l__nm_final_i_int > \g__nm_row_int
1151     {
1152         \int_compare:nNnT { #3 } = 1
1153         { \bool_set_true:N \l__nm_final_open_bool }
1154     }
1155     {
1156         \int_compare:nNnTF \l__nm_final_j_int < 1
1157         {
1158             \int_compare:nNnT { #4 } = { -1 }
1159             { \bool_set_true:N \l__nm_final_open_bool }
1160         }
1161         {
1162             \int_compare:nNnT \l__nm_final_j_int > \g__nm_col_int
1163             {
1164                 \int_compare:nNnT { #4 } = 1
1165                 { \bool_set_true:N \l__nm_final_open_bool }
1166             }
1167         }
1168     }
1169     \bool_if:NNTF \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.

```
1170 {
```

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).

```

1171         \int_sub:Nn \l__nm_final_i_int { #3 }
1172         \int_sub:Nn \l__nm_final_j_int { #4 }
1173         \bool_set_true:N \l__nm_stop_loop_bool
1174     }

```

If we are in the matrix, we test whether the cell is empty. If it’s not the case, we stop the loop because we have found the correct values for `\l_@@_final_i_int` and `\l_@@_final_j_int`.

```

1175     {
1176         \__nm_if_not_empty_cell:nnTF \l__nm_final_i_int \l__nm_final_j_int
1177         { \bool_set_true:N \l__nm_stop_loop_bool }

```

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

```

1178         {
1179             \cs_set:cpn
1180             {
1181                 __nm _ dotted _
1182                 \int_use:N \l__nm_final_i_int -
1183                 \int_use:N \l__nm_final_j_int
1184             }
1185             { }
1186         }
1187     }
1188 }

```

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

```

1189     \cs_if_free:cT
1190     {
1191         pgf@sh@ns@nm -
1192         \int_use:N \g__nm_env_int -
1193         \int_use:N \l__nm_final_i_int -
1194         \int_use:N \l__nm_final_j_int
1195     }
1196     {
1197         \bool_if:NF \l__nm_final_open_bool
1198         {
1199             \msg_error:nnx { nicematrix } { Impossible~line }
1200             { \int_use:N \l__nm_final_i_int }
1201             \bool_set_true:N \l__nm_impossible_line_bool
1202         }
1203     }

```

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

```

1204     \bool_set_false:N \l__nm_stop_loop_bool
1205     \bool_do_until:Nn \l__nm_stop_loop_bool
1206     {
1207         \int_sub:Nn \l__nm_initial_i_int { #3 }
1208         \int_sub:Nn \l__nm_initial_j_int { #4 }
1209         \bool_set_false:N \l__nm_initial_open_bool
1210         \int_compare:nNnTF \l__nm_initial_i_int < 1
1211         {
1212             \int_compare:nNnT { #3 } = 1
1213             { \bool_set_true:N \l__nm_initial_open_bool }
1214         }
1215     }

```

```

1216 \int_compare:nNnTF \l__nm_initial_j_int < 1
1217 {
1218     \int_compare:nNnT { #4 } = 1
1219     { \bool_set_true:N \l__nm_initial_open_bool }
1220 }
1221 {
1222     \int_compare:nNnT \l__nm_initial_j_int > \g__nm_col_int
1223     {
1224         \int_compare:nNnT { #4 } = { -1 }
1225         { \bool_set_true:N \l__nm_initial_open_bool }
1226     }
1227 }
1228 }
1229 \bool_if:NTF \l__nm_initial_open_bool
1230 {
1231     \int_add:Nn \l__nm_initial_i_int { #3 }
1232     \int_add:Nn \l__nm_initial_j_int { #4 }
1233     \bool_set_true:N \l__nm_stop_loop_bool
1234 }
1235 {
1236     \__nm_if_not_empty_cell:nNnTF
1237     \l__nm_initial_i_int \l__nm_initial_j_int
1238     { \bool_set_true:N \l__nm_stop_loop_bool }
1239     {
1240         \cs_set:cpn
1241         {
1242             __nm_dotted_
1243             \int_use:N \l__nm_initial_i_int -
1244             \int_use:N \l__nm_initial_j_int
1245         }
1246         { }
1247     }
1248 }
1249 }

```

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

```

1250 \cs_if_free:cT
1251 {
1252     pgf@sh@ns@nm -
1253     \int_use:N \g__nm_env_int -
1254     \int_use:N \l__nm_initial_i_int -
1255     \int_use:N \l__nm_initial_j_int
1256 }
1257 {
1258     \bool_if:NF \l__nm_initial_open_bool
1259     {
1260         \msg_error:nxx { nicematrix } { Impossible~line }
1261         { \int_use:N \l__nm_initial_i_int }
1262         \bool_set_true:N \l__nm_impossible_line_bool
1263     }
1264 }

```

If we have at least one open extremity, we create the “medium nodes” in the matrix²⁶. We remind that, when used once, the command `\@@_create_extra_nodes:` becomes no-op in the current TeX group.

```

1265 \bool_if:nT \l__nm_initial_open_bool \__nm_create_extra_nodes:
1266 \bool_if:NT \l__nm_final_open_bool \__nm_create_extra_nodes:
1267 }

```

²⁶We 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”.

The command `\@@_retrieve_coords:nn` retrieves the Tikz coordinates of the two extremities of the dotted line we will have to draw ²⁷. This command has four implicit arguments which are `\l_@@_initial_i_int`, `\l_@@_initial_j_int`, `\l_@@_final_i_int` and `\l_@@_final_j_int`. The two arguments of the command `\@@_retrieve_coords:nn` are the suffix and the anchor that must be used for the two nodes.

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

```

1268 \cs_new_protected:Nn \_nm_retrieve_coords:nn
1269 {
1270   \dim_gzero_new:N \g__nm_x_initial_dim
1271   \dim_gzero_new:N \g__nm_y_initial_dim
1272   \dim_gzero_new:N \g__nm_x_final_dim
1273   \dim_gzero_new:N \g__nm_y_final_dim
1274   \begin { tikzpicture } [ remember-picture ]
1275     \tikz@parse@node \pgfutil@firstofone
1276       ( nm - \int_use:N \g__nm_env_int -
1277         \int_use:N \l__nm_initial_i_int -
1278         \int_use:N \l__nm_initial_j_int #1 )
1279     \dim_gset:Nn \g__nm_x_initial_dim \pgf@x
1280     \dim_gset:Nn \g__nm_y_initial_dim \pgf@y
1281     \tikz@parse@node \pgfutil@firstofone
1282       ( nm - \int_use:N \g__nm_env_int -
1283         \int_use:N \l__nm_final_i_int -
1284         \int_use:N \l__nm_final_j_int #2 )
1285     \dim_gset:Nn \g__nm_x_final_dim \pgf@x
1286     \dim_gset:Nn \g__nm_y_final_dim \pgf@y
1287   \end { tikzpicture }
1288 }
1289 \cs_generate_variant:Nn \_nm_retrieve_coords:nn { x x }

```

```

1290 \cs_new_protected:Nn \_nm_draw_Ldots:nn
1291 {
1292   \cs_if_free:cT { __nm _ dotted _ #1 - #2 }
1293   {
1294     \bool_set_false:N \l__nm_impossible_line_bool
1295     \_nm_find_extremities_of_line:nnnn { #1 } { #2 } \c_zero_int \c_one_int
1296     \bool_if:NF \l__nm_impossible_line_bool \_nm_actually_draw_Ldots:
1297   }
1298 }

```

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

```

1299 \cs_new_protected:Nn \_nm_actually_draw_Ldots:
1300 {
1301   \_nm_retrieve_coords:xx
1302   {
1303     \bool_if:NTF \l__nm_initial_open_bool
1304     {

```

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

```

1305     \cs_if_exist:cTF
1306     {
1307       pgf@sh@ns@nm
1308       - \int_use:N \g__nm_env_int
1309       - \int_use:N \l__nm_initial_i_int
1310       - \int_use:N \l__nm_initial_j_int - w
1311     }

```

²⁷In fact, with diagonal lines, or vertical lines in columns of type L or R, an adjustment of one of the coordinates may be done.

```

1312         { - w.base~west }
1313         { - medium.base~west }
1314     }
1315     { .base~east }
1316 }
1317 {
1318     \bool_if:NTF \l__nm_final_open_bool
1319     {
1320         \cs_if_exist:cTF
1321         {
1322             pgf@sh@ns@nm
1323             - \int_use:N \g__nm_env_int
1324             - \int_use:N \l__nm_final_i_int
1325             - \int_use:N \l__nm_final_j_int - w
1326         }
1327         { - w.base~east }
1328         { - medium.base~east }
1329     }
1330     { .base~west }
1331 }
1332 \bool_if:NT \l__nm_initial_open_bool
1333 { \dim_gset_eq:NN \g__nm_y_initial_dim \g__nm_y_final_dim }
1334 \bool_if:NT \l__nm_final_open_bool
1335 { \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.

```

1336 \dim_gadd:Nn \g__nm_y_initial_dim { 0.53 pt }
1337 \dim_gadd:Nn \g__nm_y_final_dim { 0.53 pt }
1338 \__nm_draw_tikz_line:
1339 }

```

```

1340 \cs_new_protected:Nn \__nm_draw_Cdots:nn
1341 {
1342     \cs_if_free:cT { __nm _ dotted _ #1 - #2 }
1343     {
1344         \bool_set_false:N \l__nm_impossible_line_bool
1345         \__nm_find_extremities_of_line:nnnn { #1 } { #2 } \c_zero_int \c_one_int
1346         \bool_if:NF \l__nm_impossible_line_bool
1347         {
1348             \__nm_retrieve_coords:xx
1349             {
1350                 \bool_if:NTF \l__nm_initial_open_bool
1351                 {
1352                     \cs_if_exist:cTF
1353                     {
1354                         pgf@sh@ns@nm
1355                         - \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 - w
1358                     }
1359                     { - w.mid~west }
1360                     { - medium.mid~west }
1361                 }
1362                 { .mid~east }
1363             }
1364             {
1365                 \bool_if:NTF \l__nm_final_open_bool
1366                 {
1367                     \cs_if_exist:cTF
1368                     {
1369                         pgf@sh@ns@nm
1370                         - \int_use:N \g__nm_env_int

```

```

1371         - \int_use:N \l__nm_final_i_int
1372         - \int_use:N \l__nm_final_j_int - w
1373     }
1374     { - w.mid~east }
1375     { - medium.mid~east }
1376 }
1377 { .mid~west }
1378 }
1379 \bool_if:NT \l__nm_initial_open_bool
1380 { \dim_gset_eq:NN \g__nm_y_initial_dim \g__nm_y_final_dim }
1381 \bool_if:NT \l__nm_final_open_bool
1382 { \dim_gset_eq:NN \g__nm_y_final_dim \g__nm_y_initial_dim }
1383 \__nm_draw_tikz_line:
1384 }
1385 }
1386 }

```

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?).

```

1387 \cs_new_protected:Nn \__nm_draw_Vdots:nn
1388 {
1389     \cs_if_free:cT { __nm _ dotted _ #1 - #2 }
1390     {
1391         \bool_set_false:N \l__nm_impossible_line_bool
1392         \__nm_find_extremities_of_line:nnnn { #1 } { #2 } \c_one_int \c_zero_int
1393         \bool_if:NF \l__nm_impossible_line_bool
1394         {
1395             \__nm_retrieve_coords:xx
1396             {
1397                 \bool_if:NTF \l__nm_initial_open_bool
1398                 { - medium.north~west }
1399                 { .south~west }
1400             }
1401             {
1402                 \bool_if:NTF \l__nm_final_open_bool
1403                 { - medium.south~west }
1404                 { .north~west }
1405             }
1406         }
1407     }
1408 }

```

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

```

1406 \bool_set:Nn \l_tmpa_bool
1407 { \dim_compare_p:nNn \g__nm_x_initial_dim = \g__nm_x_final_dim }
1408 \__nm_retrieve_coords:xx
1409 {
1410     \bool_if:NTF \l__nm_initial_open_bool
1411     { - medium.north }
1412     { .south }
1413 }
1414 {
1415     \bool_if:NTF \l__nm_final_open_bool
1416     { - medium.south }
1417     { .north }
1418 }

```

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

```

1419 \bool_set:Nn \l_tmpb_bool
1420 { \dim_compare_p:nNn \g__nm_x_initial_dim = \g__nm_x_final_dim }
1421 \bool_if:NF \l_tmpb_bool
1422 {
1423     \dim_gset:Nn \g__nm_x_initial_dim

```

```

1424         {
1425             \bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
1426             \g__nm_x_initial_dim \g__nm_x_final_dim
1427         }
1428         \dim_gset_eq:NN \g__nm_x_final_dim \g__nm_x_initial_dim
1429     }
1430     \__nm_draw_tikz_line:
1431 }
1432 }
1433 }

```

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.

```

1434 \cs_new_protected:Nn \__nm_draw_Ddots:nn
1435 {
1436     \cs_if_free:cT { __nm _ dotted _ #1 - #2 }
1437     {
1438         \bool_set_false:N \l__nm_impossible_line_bool
1439         \__nm_find_extremities_of_line:nnnn { #1 } { #2 } \c_one_int \c_one_int
1440         \bool_if:NF \l__nm_impossible_line_bool
1441         {
1442             \__nm_retrieve_coords:xx
1443             {
1444                 \bool_if:NTF \l__nm_initial_open_bool
1445                 { - medium.north-west }
1446                 { .south-east }
1447             }
1448             {
1449                 \bool_if:NTF \l__nm_final_open_bool
1450                 { - medium.south-east }
1451                 { .north-west }
1452             }
1453         }
1454     }
1455 }

```

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.

```

1453     \bool_if:NT \l__nm_parallelize_diags_bool
1454     {
1455         \int_incr:N \l__nm_ddots_int

```

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

```

1456         \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.

```

1457     {
1458         \dim_set:Nn \l__nm_delta_x_one_dim
1459         { \g__nm_x_final_dim - \g__nm_x_initial_dim }
1460         \dim_set:Nn \l__nm_delta_y_one_dim
1461         { \g__nm_y_final_dim - \g__nm_y_initial_dim }
1462     }

```

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`.

```

1463     {
1464         \dim_gset:Nn \g__nm_y_final_dim
1465         {
1466             \g__nm_y_initial_dim +
1467             ( \g__nm_x_final_dim - \g__nm_x_initial_dim ) *
1468             \dim_ratio:nn \l__nm_delta_y_one_dim \l__nm_delta_x_one_dim
1469         }
1470     }
1471 }

```

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

```

1472         \__nm_draw_tikz_line:
1473     }
1474 }
1475 }

```

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

```

1476 \cs_new_protected:Nn \__nm_draw_Iddots:nn
1477 {
1478     \cs_if_free:cT { __nm _ dotted _ #1 - #2 }
1479     {
1480         \bool_set_false:N \l__nm_impossible_line_bool
1481         \__nm_find_extremities_of_line:nnnn { #1 } { #2 } 1 { -1 }
1482         \bool_if:NF \l__nm_impossible_line_bool
1483         {
1484             \__nm_retrieve_coords:xx
1485             {
1486                 \bool_if:NTF \l__nm_initial_open_bool
1487                 { - medium.north~east }
1488                 { .south~west }
1489             }
1490             {
1491                 \bool_if:NTF \l__nm_final_open_bool
1492                 { - medium.south~west }
1493                 { .north~east }
1494             }
1495             \bool_if:NT \l__nm_parallelize_diags_bool
1496             {
1497                 \int_incr:N \l__nm_iddots_int
1498                 \int_compare:nNnTF \l__nm_iddots_int = \c_one_int
1499                 {
1500                     \dim_set:Nn \l__nm_delta_x_two_dim
1501                     { \g__nm_x_final_dim - \g__nm_x_initial_dim }
1502                     \dim_set:Nn \l__nm_delta_y_two_dim
1503                     { \g__nm_y_final_dim - \g__nm_y_initial_dim }
1504                 }
1505                 {
1506                     \dim_gset:Nn \g__nm_y_final_dim
1507                     {
1508                         \g__nm_y_initial_dim +
1509                         ( \g__nm_x_final_dim - \g__nm_x_initial_dim ) *
1510                         \dim_ratio:nn \l__nm_delta_y_two_dim \l__nm_delta_x_two_dim
1511                     }
1512                 }
1513             }
1514             \__nm_draw_tikz_line:
1515         }
1516     }
1517 }

```

14.10 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`.

```

1518 \cs_new_protected:Nn \__nm_draw_tikz_line:
1519 {

```

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.

```

1520 \dim_zero_new:N \l__nm_l_dim
1521 \dim_set:Nn \l__nm_l_dim
1522 {
1523   \fp_to_dim:n
1524   {
1525     sqrt
1526     (
1527       ( \dim_use:N \g__nm_x_final_dim
1528         - \dim_use:N \g__nm_x_initial_dim
1529       ) ^ 2
1530       +
1531       ( \dim_use:N \g__nm_y_final_dim
1532         - \dim_use:N \g__nm_y_initial_dim
1533       ) ^ 2
1534     )
1535   }
1536 }

```

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

```

1537 \dim_compare:nNf \l__nm_l_dim = \c_zero_dim

```

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

```

1538 {
1539   \bool_if:NTF \l__nm_initial_open_bool
1540   {
1541     \bool_if:NTF \l__nm_final_open_bool
1542     {
1543       \int_set:Nn \l_tmpa_int
1544       { \dim_ratio:nn \l__nm_l_dim { 0.45 em } }
1545     }
1546     {
1547       \int_set:Nn \l_tmpa_int
1548       { \dim_ratio:nn { \l__nm_l_dim - 0.3 em } { 0.45 em } }
1549     }
1550   }
1551   {
1552     \bool_if:NTF \l__nm_final_open_bool
1553     {
1554       \int_set:Nn \l_tmpa_int
1555       { \dim_ratio:nn { \l__nm_l_dim - 0.3 em } { 0.45 em } }
1556     }
1557     {
1558       \int_set:Nn \l_tmpa_int
1559       { \dim_ratio:nn { \l__nm_l_dim - 0.6 em } { 0.45 em } }
1560     }
1561   }
}

```

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

```

1562 \dim_set:Nn \l_tmpa_dim
1563 {
1564   ( \g__nm_x_final_dim - \g__nm_x_initial_dim ) *
1565   \dim_ratio:nn { 0.45 em } \l__nm_l_dim
1566 }
1567 \dim_set:Nn \l_tmpb_dim
1568 {
1569   ( \g__nm_y_final_dim - \g__nm_y_initial_dim ) *
1570   \dim_ratio:nn { 0.45 em } \l__nm_l_dim
1571 }

```

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`.

```

1572 \int_set:Nn \l_tmpb_int
1573 {
1574   \bool_if:NTF \l_nm_initial_open_bool
1575   { \bool_if:NTF \l_nm_final_open_bool 1 0 }
1576   { \bool_if:NTF \l_nm_final_open_bool 2 1 }
1577 }

```

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.

```

1578 \dim_gadd:Nn \g_nm_x_initial_dim
1579 {
1580   ( \g_nm_x_final_dim - \g_nm_x_initial_dim ) *
1581   \dim_ratio:nn
1582   { \l_nm_l_dim - 0.45 em * \l_tmpa_int } { \l_nm_l_dim * 2 } *
1583   \l_tmpb_int
1584 }

```

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

```

1585 \dim_gadd:Nn \g_nm_y_initial_dim
1586 {
1587   ( \g_nm_y_final_dim - \g_nm_y_initial_dim ) *
1588   \dim_ratio:nn
1589   { \l_nm_l_dim - 0.45 em * \l_tmpa_int }
1590   { \l_nm_l_dim * 2 } *
1591   \l_tmpb_int
1592 }
1593 \begin { tikzpicture } [ overlay ]
1594   \int_step_inline:nnnn 0 1 \l_tmpa_int
1595   {
1596     \pgfpathcircle
1597     { \pgfpoint { \g_nm_x_initial_dim } { \g_nm_y_initial_dim } }
1598     { 0.53 pt }
1599     \pgfusepath { fill }
1600     \dim_gadd:Nn \g_nm_x_initial_dim \l_tmpa_dim
1601     \dim_gadd:Nn \g_nm_y_initial_dim \l_tmpb_dim
1602   }
1603 \end { tikzpicture }
1604 }
1605 }

```

14.11 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).

```

1606 \cs_set_eq:NN \_nm_ldots \ldots
1607 \cs_set_eq:NN \_nm_cdots \cdots
1608 \cs_set_eq:NN \_nm_vdots \vdots
1609 \cs_set_eq:NN \_nm_ddots \ddots
1610 \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).

```

1611 \cs_new_protected:Nn \_nm_add_to_empty_cells:
1612 {
1613   \cs_gset:cpx
1614   { \_nm _ empty _ \int_use:N \g_nm_row_int - \int_use:N \g_nm_col_int }
1615   { \int_use:N \g_nm_env_int }
1616 }

```

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

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

```

1617 \NewDocumentCommand \__nm_Ldots { s }
1618 {
1619   \bool_if:nF { #1 } { \__nm_instruction_of_type:n { Ldots } }
1620   \bool_if:NF \l__nm_nullify_dots_bool { \phantom \__nm_ldots }
1621   \__nm_add_to_empty_cells:
1622 }

1623 \NewDocumentCommand \__nm_Cdots { s }
1624 {
1625   \bool_if:nF { #1 } { \__nm_instruction_of_type:n { Cdots } }
1626   \bool_if:NF \l__nm_nullify_dots_bool { \phantom \__nm_cdots }
1627   \__nm_add_to_empty_cells:
1628 }

1629 \NewDocumentCommand \__nm_Vdots { s }
1630 {
1631   \bool_if:nF { #1 } { \__nm_instruction_of_type:n { Vdots } }
1632   \bool_if:NF \l__nm_nullify_dots_bool { \phantom \__nm_vdots }
1633   \__nm_add_to_empty_cells:
1634 }

1635 \NewDocumentCommand \__nm_Ddots { s }
1636 {
1637   \bool_if:nF { #1 } { \__nm_instruction_of_type:n { Ddots } }
1638   \bool_if:NF \l__nm_nullify_dots_bool { \phantom \__nm_ddots }
1639   \__nm_add_to_empty_cells:
1640 }

1641 \NewDocumentCommand \__nm_Iddots { s }
1642 {
1643   \bool_if:nF { #1 } { \__nm_instruction_of_type:n { Iddots } }
1644   \bool_if:NF \l__nm_nullify_dots_bool { \phantom \__nm_iddots }
1645   \__nm_add_to_empty_cells:
1646 }

```

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

```

1647 \cs_new_protected:Nn \__nm_Hspace:
1648 {
1649   \__nm_add_to_empty_cells:
1650   \hspace
1651 }

```

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

```

1652 \cs_set_eq:NN \__nm_old_multicolumn \multicolumn
1653 \cs_new:Npn \__nm_multicolumn:nnn #1 #2 #3
1654 {
1655   \__nm_old_multicolumn { #1 } { #2 } { #3 }
1656   \int_compare:nNnT #1 > 1
1657   {
1658     \seq_gput_left:Nx \g__nm_multicolumn_cells_seq
1659     { \int_eval:n \g__nm_row_int - \int_use:N \g__nm_col_int }
1660     \seq_gput_left:Nn \g__nm_multicolumn_sizes_seq { #1 }
1661   }
1662   \int_gadd:Nn \g__nm_col_int { #1 - 1 }
1663 }

```

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`.

```
1664 \cs_new:Npn \__nm_Hdotsfor:
1665 {
1666   \multicolumn { 1 } { C } { }
1667   \__nm_Hdotsfor_i
1668 }
```

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

```
1669 \bool_if:NTF \c_nm_draft_bool
1670 {
1671   \NewDocumentCommand \__nm_Hdotsfor_i { 0 { } m }
1672   { \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { C } { } } }
1673 }
1674 {
1675   \NewDocumentCommand \__nm_Hdotsfor_i { 0 { } m }
1676   {
1677     \tl_gput_right:Nx \g__nm_Hdotsfor_lines_tl
1678     {
1679       \__nm_draw_Hdotsfor:nnn
1680       { \int_use:N \g__nm_row_int }
1681       { \int_use:N \g__nm_col_int }
1682       { #2 }
1683     }
1684     \prg_replicate:nn { #2 - 1 } { & \multicolumn { 1 } { C } { } }
1685   }
1686 }
```

```
1687 \cs_new_protected:Nn \__nm_draw_Hdotsfor:nnn
1688 {
1689   \bool_set_false:N \l_nm_initial_open_bool
1690   \bool_set_false:N \l_nm_final_open_bool
```

For the row, it's easy.

```
1691 \int_set:Nn \l_nm_initial_i_int { #1 }
1692 \int_set:Nn \l_nm_final_i_int { #1 }
```

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

```
1693 \int_compare:nNnTF #2 = 1
1694 {
1695   \int_set:Nn \l_nm_initial_j_int 1
1696   \bool_set_true:N \l_nm_initial_open_bool
1697 }
1698 {
1699   \int_set:Nn \l_tmpa_int { #2 - 1 }
1700   \__nm_if_not_empty_cell:nnTF \l_nm_initial_i_int \l_tmpa_int
1701   { \int_set:Nn \l_nm_initial_j_int { #2 - 1 } }
1702   {
1703     \int_set:Nn \l_nm_initial_j_int { #2 }
1704     \bool_set_true:N \l_nm_initial_open_bool
1705   }
1706 }
1707 \int_compare:nNnTF { #2 + #3 - 1 } = \g__nm_col_int
1708 {
1709   \int_set:Nn \l_nm_final_j_int { #2 + #3 - 1 }
1710   \bool_set_true:N \l_nm_final_open_bool
1711 }
1712 {
```

```

1713     \int_set:Nn \l_tmpa_int { #2 + #3 }
1714     \__nm_if_not_empty_cell:nnTF \l__nm_final_i_int \l_tmpa_int
1715     { \int_set:Nn \l__nm_final_j_int { #2 + #3 } }
1716     {
1717         \int_set:Nn \l__nm_final_j_int { #2 + #3 - 1 }
1718         \bool_set_true:N \l__nm_final_open_bool
1719     }
1720 }
1721 \bool_if:nT { \l__nm_initial_open_bool || \l__nm_final_open_bool }
1722   \__nm_create_extra_nodes:
1723   \__nm_actually_draw_ldots:

```

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

```

1724   \int_step_inline:nnn { #2 } { #2 + #3 - 1 }
1725   { \cs_set:cpn { __nm _ dotted _ #1 - ##1 } { } }
1726 }

```

14.12 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.

```

1727 \cs_new_protected:Nn \__nm_line:nn
1728 {
1729     \dim_zero_new:N \g__nm_x_initial_dim
1730     \dim_zero_new:N \g__nm_y_initial_dim
1731     \dim_zero_new:N \g__nm_x_final_dim
1732     \dim_zero_new:N \g__nm_y_final_dim
1733     \bool_set_false:N \l__nm_initial_open_bool
1734     \bool_set_false:N \l__nm_final_open_bool
1735     \begin { tikzpicture }
1736         \path~( #1 ) ~--~( #2 ) ~node[at~start]~( i ) ~{}~node[at~end]~( f ) ~{} ;
1737         \tikz@parse@node \pgfutil@firstofone ( i )
1738         \dim_gset:Nn \g__nm_x_initial_dim \pgf@x
1739         \dim_gset:Nn \g__nm_y_initial_dim \pgf@y
1740         \tikz@parse@node \pgfutil@firstofone ( f )
1741         \dim_gset:Nn \g__nm_x_final_dim \pgf@x
1742         \dim_gset:Nn \g__nm_y_final_dim \pgf@y
1743     \end { tikzpicture }
1744     \__nm_draw_tikz_line:
1745 }

```

The commands `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, and `\Iddots` don’t use this command because they have to do other settings (for example, the diagonal lines must be parallelized).

14.13 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 position²⁸ 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.

```

1746 \cs_generate_variant:Nn \dim_set:Nn { N v }

```

²⁸In 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.

```

1747 \bool_if:NTF \c_nm_draft_bool
1748 { \cs_set_eq:NN \__nm_dotfill: \prg_do_nothing: }
1749 {
1750   \cs_set:Npn \__nm_dotfill:
1751   {
1752     \cleaders \hbox_to_wd:nn { .44 em } { \hss . \hss } \hfill
1753     \skip_horizontal:n \c_zero_dim
1754   }
1755 }

```

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

```

1756 \cs_new:Npn \__nm_hdottedline:
1757 {
1758   \noalign
1759   {
1760     \bool_gset_true:N \g_nm_extra_nodes_bool
1761     \cs_if_exist:cTF { __nm_width_ \int_use:N \g_nm_env_int }
1762     { \dim_set:Nv \l_tmpa_dim { __nm_width_ \int_use:N \g_nm_env_int } }
1763     { \dim_set:Nn \l_tmpa_dim { 5 mm } }
1764     \hbox_overlap_right:n
1765     {
1766       \hbox_to_wd:nn
1767       {
1768         \l_tmpa_dim + 2 \arraycolsep
1769         - \l_nm_left_margin_dim - \g_nm_right_margin_dim
1770       }
1771       \__nm_dotfill:
1772     }
1773   }
1774 }

```

```

1775 \cs_new_protected:Nn \__nm_vdottedline:n
1776 {

```

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

```

1777 \int_compare:nNnTF #1 = \c_zero_int
1778 { \__nm_error:n { Use-of~::~in-first-position } }
1779 {
1780   \__nm_create_extra_nodes:
1781   \bool_if:NF \c_nm_draft_bool
1782   {
1783     \dim_zero_new:N \g_nm_x_initial_dim
1784     \dim_zero_new:N \g_nm_y_initial_dim
1785     \dim_zero_new:N \g_nm_x_final_dim
1786     \dim_zero_new:N \g_nm_y_final_dim
1787     \bool_set_true:N \l_nm_initial_open_bool
1788     \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”.

```

1789 \begin { tikzpicture } [ remember~picture ]
1790   \tikz@parse@node\pgfutil@firstofone
1791   ( 1 - #1 - large .north-east )
1792   \dim_gset:Nn \g_nm_x_initial_dim \pgf@x
1793   \dim_gset:Nn \g_nm_y_initial_dim \pgf@y
1794   \tikz@parse@node\pgfutil@firstofone
1795   ( \int_use:N \g_nm_row_int - #1 - large .south-east )
1796   \dim_gset:Nn \g_nm_x_final_dim \pgf@x
1797   \dim_gset:Nn \g_nm_y_final_dim \pgf@y
1798 \end { tikzpicture }

```

However, if the `w`-nodes are created in the previous column (that is if the previous column was constructed explicitly or implicitly²⁹ 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`.

```

1799         \cs_if_exist:cT
1800         { pgf@sh@ns@nm -\int_use:N \g__nm_env_int - 1 - #1 - w }
1801         {
1802             \begin { tikzpicture } [ remember~picture ]
1803             \tikz@parse@node\pgfutil@firstofone
1804             ( 1 - #1 - w .north~east )
1805             \dim_gset:Nn \g__nm_x_initial_dim \pgf@x
1806             \tikz@parse@node\pgfutil@firstofone
1807             ( \int_use:N \g__nm_row_int - #1 - w .south~east )
1808             \dim_gset:Nn \g__nm_x_final_dim \pgf@x
1809             \end { tikzpicture }
1810             \dim_gadd:Nn \g__nm_x_initial_dim \arraycolsep
1811             \dim_gadd:Nn \g__nm_x_final_dim \arraycolsep
1812         }
1813     \__nm_draw_tikz_line:
1814 }
1815 }
1816 }
```

14.14 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:nNnF \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).

```

1817 \cs_new_protected:Npn \__nm_vline:
1818 {
1819     \int_compare:nNnTF \l__nm_first_col_int = \c_zero_int
1820     {
1821         \int_compare:nNnTF \g__nm_col_int = \c_zero_int
1822         {
1823             \int_compare:nNnTF \l__nm_first_row_int = \c_zero_int
1824             {
1825                 \int_compare:nNnF \g__nm_row_int = \c_zero_int
1826                 {
1827                     \int_compare:nNnF \g__nm_row_int = \l__nm_last_row_int
1828                     \__nm_vline_i:
1829                 }
1830             }
1831         }
1832         \int_compare:nNnF \g__nm_row_int = \c_zero_int
1833         {
1834             \int_compare:nNnF \g__nm_row_int = \l__nm_last_row_int
1835             \__nm_vline_i:

```

²⁹A 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`.

```

1836     }
1837   }
1838 }
1839 {
1840   \int_compare:nNnF \g__nm_row_int = \c_zero_int
1841   {
1842     \int_compare:nNnF \g__nm_row_int = \l__nm_last_row_int
1843     \__nm_vline_i:
1844   }
1845 }
1846 }
1847 {
1848   \int_compare:nNnTF \g__nm_col_int = \c_zero_int
1849   {
1850     \int_compare:nNnF \g__nm_row_int = { -1 }
1851     {
1852       \int_compare:nNnF \g__nm_row_int = { \l__nm_last_row_int - 1 }
1853       \__nm_vline_i:
1854     }
1855   }
1856   {
1857     \int_compare:nNnF \g__nm_row_int = \c_zero_int
1858     {
1859       \int_compare:nNnF \g__nm_row_int = \l__nm_last_row_int
1860       \__nm_vline_i:
1861     }
1862   }
1863 }
1864 }

```

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

```

1865 \cs_set_eq:NN \__nm_vline_i: \vline

```

14.15 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.

```

1866 \bool_new:N \l__nm_block_auto_columns_width_bool

```

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

```

1867 \keys_define:nn { NiceMatrix / NiceMatrixBlock }
1868 {
1869   auto-columns-width .code:n =
1870   {
1871     \bool_set_true:N \l__nm_block_auto_columns_width_bool
1872     \dim_gzero_new:N \g__nm_max_cell_width_dim
1873     \bool_set_true:N \l__nm_auto_columns_width_bool
1874   }
1875 }

1876 \NewDocumentEnvironment { NiceMatrixBlock } { ! 0 { } }
1877 {
1878   \keys_set:nn { NiceMatrix / NiceMatrixBlock } { #1 }
1879   \int_zero_new:N \l__nm_first_env_block_int
1880   \int_set:Nn \l__nm_first_env_block_int { \g__nm_env_int + 1 }
1881 }

```

At the end of the environment `{NiceMatrixBlock}`, we write in the main `.aux` file instructions for the column width of all the environments of the block (that's why we have stored the number of the first environment of the block in the counter `\l_@@_first_env_block_int`).

```

1882 {
1883   \bool_if:NT \l__nm_block_auto_columns_width_bool
1884   {
1885     \iow_now:Nn \@mainaux \ExplSyntaxOn
1886     \int_step_inline:nnnn \l__nm_first_env_block_int 1 \g__nm_env_int
1887     {
1888       \iow_now:Nx \@mainaux
1889       {
1890         \cs_gset:cpn { __nm_max_cell_width_ ##1 }
1891         { \dim_use:N \g__nm_max_cell_width_dim }
1892       }
1893     }
1894     \iow_now:Nn \@mainaux \ExplSyntaxOff
1895   }
1896 }

```

14.16 The extra nodes

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

```

1897 \cs_generate_variant:Nn \dim_min:nn { v n }
1898 \cs_generate_variant:Nn \dim_max:nn { v n }

```

The macro `\@@_create_extra_nodes:` must *not* be used in the `code-after` because the `code-after` is executed in a scope of `prefix name`.

For each row i , we compute two dimensions `l_@@_row_i_min_dim` and `l_@@_row_i_max_dim`. The dimension `l_@@_row_i_min_dim` is the minimal y -value of all the cells of the row i . The dimension `l_@@_row_i_max_dim` is the maximal y -value of all the cells of the row i .

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

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

```

1899 \cs_new_protected:Nn \__nm_create_extra_nodes:
1900 {
1901   \begin { tikzpicture } [ remember-picture , overlay ]
1902   \int_step_variable:nnNn \g__nm_first_row_int \g__nm_row_total_int \__nm_i:
1903   {
1904     \dim_zero_new:c { l__nm_row_\__nm_i: _min_dim }
1905     \dim_set_eq:cN { l__nm_row_\__nm_i: _min_dim } \c_max_dim
1906     \dim_zero_new:c { l__nm_row_\__nm_i: _max_dim }
1907     \dim_set:cn { l__nm_row_\__nm_i: _max_dim } { - \c_max_dim }
1908   }
1909   \int_step_variable:nnNn \g__nm_first_col_int \g__nm_col_total_int \__nm_j:
1910   {
1911     \dim_zero_new:c { l__nm_column_\__nm_j: _min_dim }
1912     \dim_set_eq:cN { l__nm_column_\__nm_j: _min_dim } \c_max_dim
1913     \dim_zero_new:c { l__nm_column_\__nm_j: _max_dim }
1914     \dim_set:cn { l__nm_column_\__nm_j: _max_dim } { - \c_max_dim }
1915   }

```

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

```

1916   \int_step_variable:nnNn \g__nm_first_row_int \g__nm_row_total_int \__nm_i:
1917   {
1918     \int_step_variable:nnNn
1919     \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.

```

1920         { \cs_if_exist:cT
1921           { 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`.

```

1922         {
1923           \tikz@parse@node \pgfutil@firstofone
1924             ( nm - \int_use:N \g__nm_env_int
1925               - \__nm_i: - \__nm_j: .south-west )
1926           \dim_set:cn { l__nm_row_\__nm_i: _min_dim }
1927             { \dim_min:vn { l__nm_row_\__nm_i: _min_dim } \pgf@y }
1928           \seq_if_in:NxF \g__nm_multicolumn_cells_seq { \__nm_i: - \__nm_j: }
1929             {
1930               \dim_set:cn { l__nm_column_\__nm_j: _min_dim }
1931               { \dim_min:vn { l__nm_column_\__nm_j: _min_dim } \pgf@x }
1932             }

```

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`.

```

1933           \tikz@parse@node \pgfutil@firstofone
1934             ( nm - \int_use:N \g__nm_env_int - \__nm_i: - \__nm_j: .north-east )
1935           \dim_set:cn { l__nm_row_\__nm_i: _max_dim }
1936             { \dim_max:vn { l__nm_row_\__nm_i: _max_dim } \pgf@y }
1937           \seq_if_in:NxF \g__nm_multicolumn_cells_seq { \__nm_i: - \__nm_j: }
1938             {
1939               \dim_set:cn { l__nm_column_\__nm_j: _max_dim }
1940               { \dim_max:vn { l__nm_column_\__nm_j: _max_dim } \pgf@x }
1941             }
1942         }
1943     }
1944 }

```

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`).

```

1945     \tikzset { name~suffix = -medium }
1946     \__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.

```

1947     \int_set:Nn \g__nm_first_row_int 1
1948     \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`.

```

1949     \int_step_variable:nNn { \g__nm_row_int - 1 } \__nm_i:
1950     {
1951       \dim_set:cn { l__nm_row_\__nm_i: _min_dim }
1952       {
1953         (
1954           \dim_use:c { l__nm_row_\__nm_i: _min_dim } +
1955           \dim_use:c { l__nm_row_\int_eval:n { \__nm_i: + 1 } _max_dim }
1956         )
1957         / 2
1958       }
1959       \dim_set_eq:cc { l__nm_row_\int_eval:n { \__nm_i: + 1 } _max_dim }
1960       { l__nm_row_\__nm_i: _min_dim }
1961     }
1962     \int_step_variable:nNn { \g__nm_col_int - 1 } \__nm_j:
1963     {
1964       \dim_set:cn { l__nm_column_\__nm_j: _max_dim }
1965       {

```

```

1966      (
1967      \dim_use:c
1968      { l__nm_column _ \__nm_j: _ max _ dim } +
1969      \dim_use:c
1970      { l__nm_column _ \int_eval:n { \__nm_j: + 1 } _ min _ dim }
1971      )
1972      / 2
1973      }
1974      \dim_set_eq:cc { l__nm_column _ \int_eval:n { \__nm_j: + 1 } _ min _ dim }
1975      { l__nm_column _ \__nm_j: _ max _ dim }
1976      }
1977      \dim_sub:cn
1978      { l__nm_column _ 1 _ min _ dim }
1979      \g__nm_left_margin_dim
1980      \dim_add:cn
1981      { l__nm_column _ \int_use:N \g__nm_col_int _ max _ dim }
1982      \g__nm_right_margin_dim

```

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

```

1983      \tikzset { name~suffix = -large }
1984      \__nm_create_nodes:
1985      \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.

```

1986      \cs_set:Npn \__nm_create_extra_nodes: { }

```

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

```

1987      \begin { tikzpicture } [ remember~picture , overlay ]
1988      \tikz@parse@node \pgfutil@firstofone
1989      ( nm - \int_use:N \g__nm_env_int - 1 - 1 - large .north-west )
1990      \dim_gset:Nn \g_tmpa_dim \pgf@x
1991      \tikz@parse@node \pgfutil@firstofone
1992      ( nm - \int_use:N \g__nm_env_int - 1 -
1993        \int_use:N \g__nm_col_int - large .north-east )
1994      \dim_gset:Nn \g_tmpb_dim \pgf@x
1995      \end { tikzpicture }
1996      \iow_now:Nn \@mainaux \ExplSyntaxOn
1997      \iow_now:Nx \@mainaux
1998      {
1999      \cs_gset:cpn { __nm_width_ \int_use:N \g__nm_env_int }
2000      { \dim_eval:n { \g_tmpb_dim - \g_tmpa_dim } }
2001      }
2002      \iow_now:Nn \@mainaux \ExplSyntaxOff
2003      }

```

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.

```

2004 \cs_new_protected:Nn \__nm_create_nodes:
2005 {
2006   \int_step_variable:nnNn \g__nm_first_row_int \g__nm_row_total_int \__nm_i:
2007   {
2008     \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.

```

2009   {
2010     \coordinate ( __nm~south~west )

```

```

2011         at ( \dim_use:c { l__nm_column_ \__nm_j: _min_dim } ,
2012             \dim_use:c { l__nm_row_ \__nm_i: _min_dim } ) ;
2013     \coordinate ( __nm~north~east )
2014     at ( \dim_use:c { l__nm_column_ \__nm_j: _max_dim } ,
2015         \dim_use:c { l__nm_row_ \__nm_i: _max_dim } ) ;

```

We can eventually draw the rectangular node for the cell ($\@@_i\text{-}\@@_j$). 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`.

```

2016     \node
2017     [
2018         node~contents = { } ,
2019         fit = ( __nm~south~west ) ( __nm~north~east ) ,
2020         inner~sep = \c_zero_dim ,
2021         name = nm - \int_use:N \g__nm_env_int - \__nm_i: - \__nm_j: ,
2022         alias =
2023             \str_if_empty:NF \g__nm_name_str
2024             { \g__nm_name_str - \__nm_i: - \__nm_j: }
2025     ]
2026     ;
2027 }
2028 }

```

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 .

```

2029     \__nm_seq_mapthread_function:NNN
2030     \g__nm_multicolumn_cells_seq
2031     \g__nm_multicolumn_sizes_seq
2032     \__nm_node_for_multicolumn:n
2033 }
2034 \cs_new_protected:Npn \__nm_extract_coords: #1 - #2 \q_stop
2035 {
2036     \cs_set:Npn \__nm_i: { #1 }
2037     \cs_set:Npn \__nm_j: { #2 }
2038 }

```

The command `\@@_node_for_multicolumn:n` takes two arguments. The first is the position of the cell where the command `\multicolumn{n}{...}{...}` was issued in the format $i\text{-}j$ and the second is the value of n (the length of the “multi-cell”).

```

2039 \cs_new_protected:Nn \__nm_node_for_multicolumn:n
2040 {
2041     \__nm_extract_coords: #1 \q_stop
2042     \coordinate ( __nm~south~west ) at
2043     (
2044         \dim_use:c { l__nm_column_ \__nm_j: _min _ dim } ,
2045         \dim_use:c { l__nm_row_ \__nm_i: _min _ dim }
2046     ) ;
2047     \coordinate ( __nm~north~east ) at
2048     (
2049         \dim_use:c { l__nm_column_ \int_eval:n { \__nm_j: + #2 - 1 } _ max _ dim } ,
2050         \dim_use:c { l__nm_row_ \__nm_i: _ max _ dim }
2051     ) ;
2052     \node
2053     [
2054         node~contents = { } ,
2055         fit = ( __nm~south~west ) ( __nm~north~east ) ,
2056         inner~sep = \c_zero_dim ,
2057         name = nm - \int_use:N \g__nm_env_int - \__nm_i: - \__nm_j: ,
2058         alias =
2059             \str_if_empty:NF \g__nm_name_str { \g__nm_name_str - \__nm_i: - \__nm_j: }
2060     ]
2061     ;
2062 }

```

14.17 Block matrices

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

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

```
2063 \NewExpandableDocumentCommand \_nm_Block: { m D < > { } m }
2064 {
2065   \_nm_Block_i #1 \q_stop { #2 } { #3 }
2066 }
```

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

```
2067 \cs_new:Npn \_nm_Block_i #1-#2 \q_stop
2068 {
2069   \_nm_Block_ii:nnnn { #1 } { #2 }
2070 }
```

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

```
2071 \cs_new:Npn \_nm_Block_ii:nnnn #1 #2 #3 #4
2072 {
```

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

```
2073   \int_compare:nNnTF { #1 } = 1
2074   {
2075     \multicolumn { #2 } { C } { \hbox:n { #3 $#4$ } }
2076     \_nm_gobble_ampersands:n { #2 - 1 }
2077   }
2078   { \_nm_Block_iii:nnnn { #1 } { #2 } { #3 } { #4 } }
2079 }
```

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

```
2080 \cs_new_protected:Npn \_nm_Block_iii:nnnn #1 #2 #3 #4
2081 {
2082   \bool_gset_true:N \g__nm_extra_nodes_bool
```

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

```
2083   \tl_gput_left:Nx \g__nm_code_after_tl
2084   {
2085     \_nm_Block_iv:nnnnn
2086     { \int_use:N \g__nm_row_int }
2087     { \int_use:N \g__nm_col_int }
2088     { \int_eval:n { \g__nm_row_int + #1 - 1 } }
2089     { \int_eval:n { \g__nm_col_int + #2 - 1 } }
2090     \exp_not:n { { #3 $ #4 $ } }
2091   }
2092 }
```

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

```
2093 \group_begin:
2094   \char_set_catcode_letter:N \&
2095   \cs_new:Npn \_nm_gobble_ampersands:n #1
```

```

2096 {
2097   \int_compare:nNt { #1 } > 0
2098   {
2099     \peek_charcode_remove_ignore_spaces:NT &
2100     { \_nm_gobble_ampersands:n { #1 - 1 } }
2101   }
2102 }
2103 \group_end:

```

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

```

2104 \cs_new_protected:Npn \_nm_Block_iv:nnnnn #1 #2 #3 #4 #5
2105 {
2106   \bool_if:nTF
2107   {
2108     \int_compare_p:nNn { #3 } > \g__nm_row_int
2109     || \int_compare_p:nNn { #4 } > \g__nm_col_int
2110   }
2111   { \msg_error:nnnn { nicematrix } { Block-too-large } { #1 } { #2 } }
2112   {
2113     \begin{tikzpicture}
2114       \node
2115       [
2116         fit = ( #1 - #2 - medium . north-west )
2117             ( #3 - #4 - medium . south-east ) ,
2118         inner~sep = 0 pt ,
2119       ]

```

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

```

2120       (#1-#2) { } ;
2121       \node at (#1-#2.center) { #5 } ;
2122     \end{tikzpicture}
2123   }
2124 }

```

14.18 We process the options

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

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

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

```

2125 \ProcessKeysOptions { NiceMatrix }

```

14.19 Error messages of the package

```

2126 \_nm_msg_new:nn { Block-too-large }
2127 {
2128   You~try~to~draw~a~block~in~the~cell~#1-#2~of~your~matrix~but~the~matrix~is~
2129   too~small~for~that~block.\
2130   If~you~go~on,~this~command~line~will~be~ignored.
2131 }
2132 \_nm_msg_new:nn { Impossible~line }
2133 {
2134   A~dotted~line~can't~be~drawn~because~you~have~not~put~
2135   all~the~ampersands~required~on~the~row~#1.\
2136   If~you~go~on,~this~dotted~line~will~be~ignored.
2137 }
2138 \_nm_msg_new:nn { Wrong~last~row }
2139 {

```

```

2140   You~have~used~'last-row=\int_use:N \g__nm_last_row_int'~but~your~environment~
2141   \{\@currentvir\}~seems~to~have~\int_use:N \g__nm_row_int\
2142   rows~(remark~that~you~can~use~'last-row'~without~value).~If~you~go~on,
2143   ~the~value~of~\int_use:N \g__nm_row_int\
2144   will~be~used~for~last~row.
2145 }

2146 \__nm_msg_new:nn { Draft~mode }
2147 { The~compilation~is~in~draft~mode:~the~dotted~lines~won't~be~drawn. }

2148 \__nm_msg_new:nn { Yet~in~env }
2149 {
2150   Environments~\{NiceArray\}~(or~\{NiceMatrix\},~etc.)~can't~be~
2151   nested.\\
2152   This~error~is~fatal.
2153 }

2154 \__nm_msg_new:nn { Outside~math~mode }
2155 {
2156   The~environment~\{\@currentvir\}~can~be~used~only~in~math~mode~
2157   (and~not~in~\token_to_str:N \vcenter).\\
2158   This~error~is~fatal.
2159 }

2160 \__nm_msg_new:nn { Option~Transparent~suppressed }
2161 {
2162   The~option~'Transparent'~has~been~renamed~'transparent'.\\
2163   However,~you~can~go~on~for~this~time.
2164 }

2165 \__nm_msg_new:nn { Option~RenewMatrix~suppressed }
2166 {
2167   The~option~'RenewMatrix'~has~been~renamed~'renew-matrix'.\\
2168   However,~you~can~go~on~for~this~time.
2169 }

2170 \__nm_msg_new:nn { Bad~value~for~letter~for~dotted~lines }
2171 {
2172   The~value~of~key~'\tl_use:N\l_keys_key_tl'~must~be~of~length~1.\\
2173   If~you~go~on,~it~will~be~ignored.
2174 }

2175 \__nm_msg_new:nnn { Unknown~key~for~NiceMatrixOptions }
2176 {
2177   The~key~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~command~
2178   \token_to_str:N \NiceMatrixOptions. \\
2179   If~you~go~on,~it~will~be~ignored. \\
2180   For~a~list~of~the~available~keys,~type~H~<return>.
2181 }
2182 {
2183   The~available~keys~are~(in~alphabetic~order):~
2184   allow-duplicate-names,~
2185   code-for-first-col,~
2186   code-for-first-row,~
2187   code-for-last-col,~
2188   code-for-last-row,~
2189   exterior-arraycolsep,~
2190   hlines,~
2191   left-margin,~
2192   letter-for-dotted-lines,~
2193   nullify-dots,~
2194   parallelize-diags,~
2195   renew-dots,~
2196   renew-matrix,~
2197   right-margin,~
2198   and~transparent
2199 }

2200 \__nm_msg_new:nnn { Unknown~option~for~NiceArray }

```

```

2201 {
2202   The~option~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~environment~
2203   \{NiceArray\}. \\
2204   If~you~go~on,~it~will~be~ignored. \\
2205   For~a~list~of~the~available~options,~type~H~<return>.
2206 }
2207 {
2208   The~available~options~are~(in~alphabetic~order):~
2209   b,~
2210   c,~
2211   code-after,~
2212   code-for-first-col,~
2213   code-for-first-row,~
2214   code-for-last-col,~
2215   code-for-last-row,~
2216   columns-width,~
2217   create-extra-nodes,~
2218   extra-left-margin,~
2219   extra-right-margin,~
2220   hlines,~
2221   left-margin,~
2222   name,~
2223   nullify-dots,~
2224   parallelize-diags,~
2225   renew-dots,~
2226   right-margin,~
2227   and~t.
2228 }
2229 \__nm_msg_new:nnn { Unknown~option~for~NiceMatrix }
2230 {
2231   The~option~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~environment~
2232   \{NiceMatrix\}~and~its~variants. \\
2233   If~you~go~on,~it~will~be~ignored. \\
2234   For~a~list~of~the~available~options,~type~H~<return>.
2235 }
2236 {
2237   The~available~options~are~(in~alphabetic~order):~
2238   code-after,~
2239   columns-width,~
2240   create-extra-nodes,~
2241   extra-left-margin,~
2242   extra-right-margin,~
2243   hlines,~
2244   left-margin,~
2245   name,~
2246   nullify-dots,~
2247   parallelize-diags,~
2248   renew-dots~
2249   and~right-margin.
2250 }

```

Despite its name, the following set of keys will be used for {pNiceArray} but also {vNiceArray}, {VNiceArray}, etc. but not for {NiceArray}.

```

2251 \__nm_msg_new:nnn { Unknown~option~for~pNiceArray }
2252 {
2253   The~option~'\tl_use:N\l_keys_key_tl'~is~unknown~for~the~environment~
2254   \{@currentenv\}. \\
2255   If~you~go~on,~it~will~be~ignored. \\
2256   For~a~list~of~the~available~options,~type~H~<return>.
2257 }
2258 {
2259   The~available~options~are~(in~alphabetic~order):~
2260   code-after,~

```

```

2261 code-for-first-col,~
2262 code-for-first-row,~
2263 code-for-last-col,~
2264 code-for-last-row,~
2265 columns-width,~
2266 create-extra-nodes,~
2267 extra-left-margin,~
2268 extra-right-margin,~
2269 first-col,~
2270 first-row,~
2271 last-col,~
2272 last-row,~
2273 hlines,~
2274 left-margin,~
2275 name,~
2276 nullify-dots,~
2277 parallelize-diags,~
2278 renew-dots,~
2279 and-right-margin.
2280 }
2281 \_nm_msg_new:nnn { Duplicate-name }
2282 {
2283   The~name~'\l_keys_value_tl'~is~already~used~and~you~shouldn't~use~
2284   the~same~environment~name~twice.~You~can~go~on,~but,~
2285   maybe,~you~will~have~incorrect~results~especially~
2286   if~you~use~'columns-width=auto'. \\\
2287   For~a~list~of~the~names~already~used,~type~H~<return>. \\\
2288   If~you~don't~want~to~see~this~message~again,~use~the~option~
2289   'allow-duplicate-names'.
2290 }
2291 {
2292   The~names~already~defined~in~this~document~are:~
2293   \seq_use:Nnnn \g__nm_names_seq { ,~ } { ,~ } { ~and~ }.
2294 }
2295 \_nm_msg_new:nn { Option~auto~for~columns-width }
2296 {
2297   You~can't~give~the~value~'auto'~to~the~option~'columns-width'~here.~
2298   If~you~go~on,~the~option~will~be~ignored.
2299 }
2300 \_nm_msg_new:nn { Zero~row }
2301 {
2302   There~is~a~problem.~Maybe~your~environment~\{\@currenvir\}~is~empty.~
2303   Maybe~you~have~used~l,~c~and~r~instead~of~L,~C~and~R~in~the~preamble~
2304   of~your~environment. \\\
2305   If~you~go~on,~the~result~may~be~incorrect.
2306 }
2307 \_nm_msg_new:nn { Use~of~::~in~first~position }
2308 {
2309   You~can't~use~the~column~specifier~'\l_nm_letter_for_dotted_lines_str'~in~the~
2310   first~position~of~the~preamble~of~the~environment~\{\@currenvir\}. \\\
2311   If~you~go~on,~this~dotted~line~will~be~ignored.
2312 }

```

14.20 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.

```

2313 \cs_new:Npn \_nm_seq_mapthread_function:NNN #1 #2 #3
2314 {
2315   \group_begin:

```

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

```

2316 \int_step_inline:nn { \seq_count:N #1 }
2317 {
2318   \seq_pop:NN #1 \l_tmpa_tl
2319   \seq_pop:NN #2 \l_tmpb_tl
2320   \exp_args:NVV #3 \l_tmpa_tl \l_tmpb_tl
2321 }
2322 \group_end:
2323 }
2324 \cs_set_protected:Npn \__nm_renew_matrix:
2325 {
2326   \RenewDocumentEnvironment { pmatrix } { } {
2327     { \pNiceMatrix }
2328     { \endpNiceMatrix }
2329   } \RenewDocumentEnvironment { vmatrix } { } {
2330     { \vNiceMatrix }
2331     { \endvNiceMatrix }
2332   } \RenewDocumentEnvironment { Vmatrix } { } {
2333     { \VNiceMatrix }
2334     { \endVNiceMatrix }
2335   } \RenewDocumentEnvironment { bmatrix } { } {
2336     { \bNiceMatrix }
2337     { \endbNiceMatrix }
2338   } \RenewDocumentEnvironment { Bmatrix } { } {
2339     { \BNiceMatrix }
2340     { \endBNiceMatrix }
2341   }

```

14.21 Obsolete environments

```

2342 \NewDocumentEnvironment { pNiceArrayC } { } {
2343   {
2344     \bool_set_true:N \l__nm_last_col_bool
2345     \pNiceArray
2346   }
2347   { \endpNiceArray }
2348 \NewDocumentEnvironment { bNiceArrayC } { } {
2349   {
2350     \bool_set_true:N \l__nm_last_col_bool
2351     \bNiceArray
2352   }
2353   { \endbNiceArray }
2354 \NewDocumentEnvironment { BNiceArrayC } { } {
2355   {
2356     \bool_set_true:N \l__nm_last_col_bool
2357     \BNiceArray
2358   }
2359   { \endBNiceArray }
2360 \NewDocumentEnvironment { vNiceArrayC } { } {
2361   {
2362     \bool_set_true:N \l__nm_last_col_bool
2363     \vNiceArray
2364   }
2365   { \endvNiceArray }
2366 \NewDocumentEnvironment { VNiceArrayC } { } {
2367   {
2368     \bool_set_true:N \l__nm_last_col_bool
2369     \VNiceArray
2370   }
2371   { \endVNiceArray }

```

```

2372 \NewDocumentEnvironment { pNiceArrayRC } { }
2373 {
2374     \bool_set_true:N \l__nm_last_col_bool
2375     \int_set:Nn \l__nm_first_row_int \c_zero_int
2376     \pNiceArray
2377 }
2378 { \endpNiceArray }

2379 \NewDocumentEnvironment { bNiceArrayRC } { }
2380 {
2381     \bool_set_true:N \l__nm_last_col_bool
2382     \int_set:Nn \l__nm_first_row_int \c_zero_int
2383     \bNiceArray
2384 }
2385 { \endbNiceArray }

2386 \NewDocumentEnvironment { BNiceArrayRC } { }
2387 {
2388     \bool_set_true:N \l__nm_last_col_bool
2389     \int_set:Nn \l__nm_first_row_int \c_zero_int
2390     \BNiceArray
2391 }
2392 { \endBNiceArray }

2393 \NewDocumentEnvironment { vNiceArrayRC } { }
2394 {
2395     \bool_set_true:N \l__nm_last_col_bool
2396     \int_set:Nn \l__nm_first_row_int \c_zero_int
2397     \vNiceArray
2398 }
2399 { \endvNiceArray }

2400 \NewDocumentEnvironment { VNiceArrayRC } { }
2401 {
2402     \bool_set_true:N \l__nm_last_col_bool
2403     \int_set:Nn \l__nm_first_row_int \c_zero_int
2404     \VNiceArray
2405 }
2406 { \endVNiceArray }

2407 \NewDocumentEnvironment { NiceArrayCwithDelims } { }
2408 {
2409     \bool_set_true:N \l__nm_last_col_bool
2410     \NiceArrayWithDelims
2411 }
2412 { \endNiceArrayWithDelims }

2413 \NewDocumentEnvironment { NiceArrayRCwithDelims } { }
2414 {
2415     \bool_set_true:N \l__nm_last_col_bool
2416     \int_set:Nn \l__nm_first_row_int \c_zero_int
2417     \NiceArrayWithDelims
2418 }
2419 { \endNiceArrayWithDelims }

```

15 History

Changes between versions 1.0 and 1.1

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

Changes between versions 1.1 and 1.2

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

Changes between version 1.2 and 1.3

New environment `{pNiceArrayC}` and its variants.

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

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

The names of the options are changed. The old names were names in “camel style”. 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 StackExchange³⁰, Tikz externalization is now deactivated in the environments of the extension `nicematrix`.³¹

Changes between version 2.1 and 2.1.2

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

Changes between version 2.1.2 and 2.1.3

When searching the end of a dotted line from a command like `\Cdots` issued in the “main matrix” (not in the 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`).

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

³⁰cf. tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package

³¹Before 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`.

2232, 2233, 2254, 2255, 2286, 2287, 2304, 2310
 $\{$ 925, 981,
 2141, 2150, 2156, 2203, 2232, 2254, 2302, 2310
 $\}$ 925, 983,
 2141, 2150, 2156, 2203, 2232, 2254, 2302, 2310
 $|$ 937

 \backslash 2141, 2143

A
 \array 431
 \arraycolsep 156, 158, 160, 619, 620, 730, 768,
 770, 784, 841, 868, 953, 962, 1768, 1810, 1811
 \arrayrulewidth 445, 446
 \AtBeginDocument 62, 85

B
 \baselineskip 690, 700, 711, 722
 \begin 969, 975, 981, 987, 993, 1046,
 1274, 1593, 1735, 1789, 1802, 1901, 1987, 2113
 \bgroup 347
 \Block 522
 \BNiceArray 2357, 2390
 \bNiceArray 2351, 2383
 \BNiceMatrix 2339
 \bNiceMatrix 2336
bool commands:
 $\bool_do_until:Nn$ 1145, 1205
 $\bool_gset_eq:NN$ 471, 479, 480
 $\bool_gset_false:N$ 570
 $\bool_gset_true:N$ 558, 848, 1760, 2082
 $\bool_if:N\TF$.. 34, 93, 407, 423, 439, 453,
 484, 523, 568, 582, 588, 591, 617, 648, 650,
 654, 657, 659, 677, 733, 744, 781, 943, 949,
 998, 1043, 1083, 1101, 1110, 1169, 1197,
 1229, 1258, 1266, 1296, 1303, 1318, 1332,
 1334, 1346, 1350, 1365, 1379, 1381, 1393,
 1397, 1402, 1410, 1415, 1421, 1425, 1440,
 1444, 1449, 1453, 1482, 1486, 1491, 1495,
 1539, 1541, 1552, 1574, 1575, 1576, 1620,
 1626, 1632, 1638, 1644, 1669, 1747, 1781, 1883
 $\bool_if:n\TF$ 1079,
 1265, 1619, 1625, 1631, 1637, 1643, 1721, 2106
 $\bool_new:N$ 11,
 27, 51, 52, 53, 60, 61, 80, 81, 82, 83,
 84, 125, 126, 128, 129, 130, 133, 134, 1866
 $\bool_set:Nn$ 1406, 1419
 $\bool_set_false:N$ 957, 1019, 1033,
 1115, 1116, 1144, 1149, 1204, 1209, 1294,
 1344, 1391, 1438, 1480, 1689, 1690, 1733, 1734
 $\bool_set_true:N$
 12, 29, 32, 66, 88, 127, 181, 230,
 583, 597, 906, 944, 1041, 1153, 1159, 1165,
 1173, 1177, 1201, 1213, 1219, 1225, 1233,
 1238, 1262, 1696, 1704, 1710, 1718, 1787,
 1788, 1871, 1873, 2344, 2350, 2356, 2362,
 2368, 2374, 2381, 2388, 2395, 2402, 2409, 2415
 \l_tmpa_bool 1019, 1033, 1041, 1043, 1406, 1425
 \l_tmpb_bool 1419, 1421
box commands:
 $\box_clear_new:N$ 643
 $\box_dp:N$ 305, 317, 367, 778
 $\box_ht:N$ 307, 312, 315, 777

$\box_move_down:nn$ 368, 748
 $\box_move_up:nn$ 382, 739
 $\box_set_dp:Nn$ 778
 $\box_set_ht:Nn$ 777
 $\box_use:N$ 348, 366, 382, 833, 900
 $\box_use_drop:N$ 756, 769, 779
 $\box_wd:N$ 324, 372, 632, 640, 806, 866
 \l_tmpa_box 284, 305, 307, 312, 315,
 317, 324, 348, 357, 366, 625, 632, 633, 640,
 759, 777, 778, 779, 792, 806, 833, 852, 866, 900
 \l_tmpb_box 365, 367, 372, 382

C
 \Cdots 514
 \cdots 526, 1607
Cdots internal commands:
 $_nm_Cdots$ 514, 526, 1623
cdots internal commands:
 $_nm_cdots$ 1607, 1626
char commands:
 $\char_set_catcode_letter:N$ 2094
 \cleaders 1752
 \coordinate ... 376, 381, 2010, 2013, 2042, 2047
cs commands:
 $\cs_generate_variant:Nn$
 351, 1289, 1746, 1897, 1898
 $\cs_gset:Npn$ 1003, 1010, 1088, 1095, 1890, 1999
 $\cs_gset:Npx$ 1613
 $\cs_gset_eq:NN$ 106, 488
 $\cs_if_exist:N\TF$
 584, 600, 607, 945, 1020, 1034,
 1076, 1305, 1320, 1352, 1367, 1761, 1799, 1920
 $\cs_if_free:N\TF$ 456, 464,
 1024, 1189, 1250, 1292, 1342, 1389, 1436, 1478
 $\cs_new:Npn$
 434, 1653, 1664, 1756, 2067, 2071, 2095, 2313
 $\cs_new_protected:Nn$ 274,
 293, 319, 352, 996, 1067, 1073, 1137, 1268,
 1290, 1299, 1340, 1387, 1434, 1476, 1518,
 1611, 1647, 1687, 1727, 1775, 1899, 2004, 2039
 $\cs_new_protected:Npn$ 18,
 19, 20, 21, 22, 23, 24, 25, 54, 109, 300,
 410, 421, 436, 451, 1817, 2034, 2080, 2104
 $\cs_set:Npn$ 428, 482, 494,
 1139, 1179, 1240, 1725, 1750, 1986, 2036, 2037
 $\cs_set_eq:NN$ 97,
 425, 426, 427, 513, 514, 515, 516, 517, 518,
 519, 520, 521, 522, 525, 526, 527, 528,
 529, 530, 531, 540, 541, 542, 544, 1132,
 1606, 1607, 1608, 1609, 1610, 1652, 1748, 1865
 $\cs_set_protected:Npn$ 67, 91, 408, 590, 2324

D
 \Ddots 516
 \ddots 528, 1609
Ddots internal commands:
 $_nm_Ddots$ 516, 528, 1635
ddots internal commands:
 $_nm_ddots$ 1609, 1638
 \DeclareOption 12, 13
dim commands:
 $\dim_abs:n$ 1060
 $\dim_add:Nn$ 1980
 $\dim_compare:nN\TF$ 502, 688, 706, 1059, 1537

<code>\dim_compare_p:nNn</code>	1407, 1420	<code>\endvNiceMatrix</code>	2331
<code>\dim_eval:n</code>	2000	<code>\everycr</code>	492, 496
<code>\dim_gadd:Nn</code>		exp commands:	
1336, 1337, 1578, 1585, 1600, 1601, 1810, 1811		<code>\exp_after:wN</code>	113
<code>\dim_gset:Nn</code>	304, 306,	<code>\exp_args:NV</code>	553, 667
311, 314, 316, 323, 619, 620, 632, 640, 802,		<code>\exp_args:NVV</code>	2320
862, 1055, 1057, 1279, 1280, 1285, 1286,		<code>\exp_not:n</code>	2090
1423, 1464, 1506, 1738, 1739, 1741, 1742,		<code>\ExplSyntaxOff</code>	1014, 1099, 1894, 2002
1792, 1793, 1796, 1797, 1805, 1808, 1990, 1994		<code>\ExplSyntaxOn</code>	1000, 1085, 1885, 1996
<code>\dim_gset_eq:NN</code>	296,		
472, 473, 474, 1333, 1335, 1380, 1382, 1428		F	
<code>\dim_gzero:N</code>	297, 298	fi commands:	
<code>\dim_gzero_new:N</code>	545, 546, 547, 548, 549,	<code>\fi:</code>	58
550, 589, 950, 1270, 1271, 1272, 1273, 1872		fp commands:	
<code>\dim_max:nn</code>	305, 307, 312,	<code>\fp_to_dim:n</code>	1523
315, 317, 324, 804, 864, 1425, 1898, 1936, 1940			
<code>\dim_min:nn</code>	1425, 1897, 1927, 1931	G	
<code>\dim_new:N</code>	49,	group commands:	
135, 136, 137, 138, 139, 140, 141, 142, 143		<code>\group_begin:</code>	95, 623, 1075, 2093, 2315
<code>\dim_ratio:nn</code>	1468, 1510,	<code>\group_end:</code>	100, 641, 1135, 2103, 2322
1544, 1548, 1555, 1559, 1565, 1570, 1581, 1588		<code>\group_insert_after:N</code>	455, 586, 947
<code>\dim_set:Nn</code>	182, 236, 351, 367, 459,		
466, 692, 699, 713, 720, 1458, 1460, 1500,		H	
1502, 1521, 1562, 1567, 1746, 1762, 1763,		<code>\halign</code>	498, 500
1907, 1914, 1926, 1930, 1935, 1939, 1951, 1964		hbox commands:	
<code>\dim_set_eq:NN</code>	624, 635, 1905, 1912, 1959, 1974	<code>\hbox:n</code>	41, 43, 45, 378, 766, 2075
<code>\dim_sub:Nn</code>	1977	<code>\hbox_overlap_left:n</code>	808
<code>\dim_use:N</code>		<code>\hbox_overlap_right:n</code>	869, 1764
. . . 509, 510, 511, 1004, 1011, 1527, 1528,		<code>\hbox_set:Nn</code>	365, 625, 633, 759
1531, 1532, 1891, 1954, 1955, 1967, 1969,		<code>\hbox_set:Nw</code>	284, 357, 663, 792, 852
2011, 2012, 2014, 2015, 2044, 2045, 2049, 2050		<code>\hbox_set_end:</code>	322, 363, 674, 800, 860
<code>\dim_zero:N</code>	457, 703, 727	<code>\hbox_to_wd:nn</code>	372, 1752, 1766
<code>\dim_zero_new:N</code>	615, 616, 1105, 1106,	<code>\Hdotsfor</code>	520
1107, 1108, 1520, 1729, 1730, 1731, 1732,		<code>\hdotsfor</code>	531
1783, 1784, 1785, 1786, 1904, 1906, 1911, 1913		<code>\hdottedline</code>	518
<code>\c_max_dim</code>	1905, 1907, 1912, 1914	<code>\hfil</code>	374, 544
<code>\g_tmpa_dim</code>	1055, 1060, 1990, 2000	<code>\hfill</code>	1752
<code>\l_tmpa_dim</code>	367, 368, 382, 692, 699, 703,	<code>\hrule</code>	445
740, 765, 777, 1562, 1600, 1762, 1763, 1768		<code>\Hspace</code>	519
<code>\g_tmpb_dim</code>	1057, 1060, 1994, 2000	<code>\hspace</code>	1650
<code>\l_tmpb_dim</code>		<code>\hss</code>	544, 1752
. . . 713, 720, 727, 750, 772, 778, 1567, 1601			
<code>\c_zero_dim</code>	329, 330, 400, 502, 624,	I	
635, 813, 814, 880, 881, 1537, 1753, 2020, 2056		<code>\ialign</code>	482, 494
<code>\dots</code>	530	<code>\Iddots</code>	517
		<code>\iddots</code>	36, 529, 1610
E		lddots internal commands:	
<code>\egroup</code>	349	<code>_nm_Iddots</code>	517, 529, 1641
else commands:		iddots internal commands:	
<code>\else:</code>	56	<code>_nm_iddots</code>	1610, 1644
<code>\end</code>	971, 977, 983, 989, 995, 1058,	if commands:	
1287, 1603, 1743, 1798, 1809, 1985, 1995, 2122		<code>\if_mode_math:</code>	56
<code>\endarray</code>	670, 961	int commands:	
<code>\endBNiceArray</code>	2359, 2392	<code>\int_add:Nn</code>	1147, 1148, 1231, 1232
<code>\endbNiceArray</code>	2353, 2385	<code>\int_compare:nNnTF</code>	277,
<code>\endBNiceMatrix</code>	2340	279, 286, 289, 302, 309, 441, 443, 559, 595,	
<code>\endbNiceMatrix</code>	2337	645, 675, 679, 686, 704, 728, 735, 1037,	
<code>\endNiceArrayWithDelims</code>		1069, 1081, 1150, 1152, 1156, 1158, 1162,	
. . . 909, 915, 921, 927, 933, 939, 2412, 2419		1164, 1210, 1212, 1216, 1218, 1222, 1224,	
<code>\endpNiceArray</code>	2347, 2378	1456, 1498, 1656, 1693, 1707, 1777, 1819,	
<code>\endpNiceMatrix</code>	2328	1821, 1823, 1825, 1827, 1832, 1834, 1840,	
<code>\endvNiceArray</code>	2371, 2406	1842, 1848, 1850, 1852, 1857, 1859, 2073, 2097	
<code>\endvNiceArray</code>	2365, 2399	<code>\int_compare:nTF</code>	242
<code>\endvNiceMatrix</code>	2334	<code>\int_compare_p:nNn</code>	2108, 2109

`\int_eval:n`
 1659, 1955, 1959, 1970, 1974, 2049, 2088, 2089
`\int_gadd:Nn` 1662
`\int_gdecr:N` 1079
`\int_gincr:N` 276, 295, 587, 849, 948
`\int_gset:Nn` 282, 536, 569, 850
`\int_gset_eq:NN`
 475, 477, 478, 561, 682, 1078, 1080
`\int_gsub:Nn` 1082
`\int_gzero:N` 438
`\int_gzero_new:N` 535, 537, 538, 539, 567
`\int_incr:N` 1455, 1497
`\int_max:nn` 283, 851
`\int_new:N` 48, 71, 73, 74, 76, 77, 79
`\int_set:Nn` 72, 75, 78, 602,
 609, 1140, 1141, 1142, 1143, 1543, 1547,
 1554, 1558, 1572, 1691, 1692, 1695, 1699,
 1701, 1703, 1709, 1713, 1715, 1717, 1880,
 1947, 1948, 2375, 2382, 2389, 2396, 2403, 2416
`\int_step_inline:nn` 2316
`\int_step_inline:nnn` 1724
`\int_step_inline:nnnn` 1594, 1886
`\int_step_variable:nNn` 1949, 1962
`\int_step_variable:nnNn`
 1902, 1909, 1916, 1918, 2006, 2008
`\int_sub:Nn` 1171, 1172, 1207, 1208
`\int_use:N` 336, 337, 338, 343, 344,
 390, 391, 392, 397, 398, 416, 417, 456, 460,
 563, 600, 603, 822, 823, 829, 889, 890, 891,
 896, 897, 1003, 1021, 1027, 1028, 1029,
 1035, 1038, 1050, 1051, 1052, 1088, 1089,
 1096, 1129, 1182, 1183, 1192, 1193, 1194,
 1200, 1243, 1244, 1253, 1254, 1255, 1261,
 1276, 1277, 1278, 1282, 1283, 1284, 1308,
 1309, 1310, 1323, 1324, 1325, 1355, 1356,
 1357, 1370, 1371, 1372, 1614, 1615, 1659,
 1680, 1681, 1761, 1762, 1795, 1800, 1807,
 1921, 1924, 1934, 1981, 1989, 1992, 1993,
 1999, 2021, 2057, 2086, 2087, 2140, 2141, 2143
`\int_zero:N` 258, 260, 267, 269
`\int_zero_new:N`
 ... 1103, 1104, 1111, 1112, 1113, 1114, 1879
`\c_one_int` 242, 277, 279,
 309, 1082, 1295, 1345, 1392, 1439, 1456, 1498
`\l_tmpa_int` 1543, 1547, 1554,
 1558, 1582, 1589, 1594, 1699, 1700, 1713, 1714
`\l_tmpb_int` 1572, 1583, 1591
`\c_zero_int` 286, 302,
 645, 686, 728, 735, 1069, 1295, 1345, 1392,
 1777, 1819, 1821, 1823, 1825, 1832, 1840,
 1848, 1857, 2375, 2382, 2389, 2396, 2403, 2416
iow commands:
`\iow_now:Nn`
 1000, 1001, 1008, 1014, 1085, 1086,
 1093, 1099, 1885, 1888, 1894, 1996, 1997, 2002

K

`\kern` 45
keys commands:
`\keys_define:nn`
 . 144, 166, 177, 195, 220, 251, 253, 265, 1867
`\l_keys_key_tl` . 2172, 2177, 2202, 2231, 2253
`\keys_set:nn` 250, 592, 593, 951, 1878
`\l_keys_value_tl` 2283

L

`\Ldots` 513
`\ldots` 525, 1606
Ldots internal commands:
`__nm_Ldots` 513, 525, 530, 1617
ldots internal commands:
`__nm_ldots` 1606, 1620
`\left` 628, 637, 762, 969, 975, 981, 987, 993
`\line` 1132
`\lineskip` 694, 715
`\lineskiplimit` 689, 708
`\lVert` 993
`\lvert` 987

M

`\makebox` 366
math commands:
`\c_math_toggle_token` . 285, 321, 627, 629,
 636, 638, 666, 671, 761, 775, 793, 799, 853, 859
`\mathinner` 38
`\mkern` 40, 42, 44, 45
msg commands:
`\msg_error:nn` 18, 19
`\msg_error:nnn` 20, 1199, 1260
`\msg_error:nnnn` 2111
`\msg_fatal:nn` 21, 22
`\msg_new:nnn` 23
`\msg_new:nnnn` 24
`\msg_redirect_name:nnn` 26
`\msg_warning:nn` 35
`\multicolumn` .. 521, 1652, 1666, 1672, 1684, 2075
`\myfiledate` 8
`\myfileversion` 9

N

`\newcolumntype`
 354, 504, 505, 506, 509, 510, 511, 553
`\NewDocumentCommand`
 249, 1617, 1623, 1629, 1635, 1641, 1671, 1675
`\NewDocumentEnvironment` 578, 904,
 910, 916, 922, 928, 934, 940, 966, 972, 978,
 984, 990, 1876, 2342, 2348, 2354, 2360,
 2366, 2372, 2379, 2386, 2393, 2400, 2407, 2413
`\NewExpandableDocumentCommand` 2063
`\NiceArrayWithDelims`
 ... 907, 913, 919, 925, 931, 937, 2410, 2417
`\NiceMatrixOptions` 249, 2178
nm internal commands:
`__nm_actualization_for_first_and_-`
`last_row:` 300, 325, 801, 861
`__nm_actually_draw_Ldots:` 1296, 1299, 1723
`__nm_adapt_S_column:` 91, 106, 580
`__nm_add_to_empty_cells:`
 ... 1611, 1621, 1627, 1633, 1639, 1645, 1649
`__nm_after_array:` 586, 947, 1067
`__nm_after_array_i:` 1070, 1073
`__nm_array:` 421, 667, 958
`\l_nm_auto_columns_width_bool`
 130, 181, 453, 1873
`__nm_begin_of_row:` 280, 293, 791
`__nm_Block:` 522, 2063
`\l_nm_block_auto_columns_width_bool` .
 588, 949, 998, 1866, 1871, 1883
`__nm_Block_i` 2065, 2067

_nm_Block_ii:nnnn	2069, 2071	\g_nm_extra_nodes_bool	134, 471, 558, 1110, 1760, 2082
_nm_Block_iii:nnnn	2078, 2080	\l_nm_extra_nodes_bool	133, 153, 471
_nm_Block_iv:nnnnn	2085, 2104	\g_nm_extra_right_margin_dim	143, 474, 673, 964
\g_nm_Cdots_lines_tl	571, 1121	\l_nm_extra_right_margin_dim	142, 162, 474, 875
_nm_Cell:	116, 274, 358, 504, 505, 506	_nm_extract_coords:	2034, 2041
\g_nm_code_after_tl	191, 562, 1133, 1134, 2083	_nm_fatal:n	21, 57, 582, 943
\l_nm_code_for_first_col_tl	168, 794	_nm_fatal:nn	22
\l_nm_code_for_first_row_tl	172, 287	\l_nm_final_i_int	1113, 1142, 1147, 1150, 1171, 1176, 1182, 1193, 1200, 1283, 1324, 1371, 1692, 1714
\l_nm_code_for_last_col_tl	170, 854	\l_nm_final_j_int	1114, 1143, 1148, 1156, 1162, 1172, 1176, 1183, 1194, 1284, 1325, 1372, 1709, 1715, 1717
\l_nm_code_for_last_row_tl	174, 290	\l_nm_final_open_bool	1116, 1149, 1153, 1159, 1165, 1169, 1197, 1266, 1318, 1334, 1365, 1381, 1402, 1415, 1449, 1491, 1541, 1552, 1575, 1576, 1690, 1710, 1718, 1721, 1734, 1788
\g_nm_col_int	276, 277, 283, 338, 344, 392, 398, 417, 438, 538, 559, 561, 563, 849, 851, 891, 897, 1078, 1079, 1162, 1222, 1614, 1659, 1662, 1681, 1707, 1821, 1848, 1962, 1981, 1993, 2087, 2089, 2109	_nm_find_extremities_of_line:nnnn	1137, 1295, 1345, 1392, 1439, 1481
\g_nm_col_total_int	282, 283, 539, 850, 851, 1078, 1909, 1919, 2008	\g_nm_first_col_int	76, 478, 728, 1909, 1919, 1948, 2008
\c_nm_colortbl_loaded_bool	61, 66, 484	\l_nm_first_col_int	74, 75, 258, 267, 279, 478, 645, 1819
\l_nm_columns_width_dim	49, 182, 236, 457, 459, 466, 502, 509, 510, 511	\l_nm_first_env_block_int	1879, 1880, 1886
_nm_create_extra_nodes:	1110, 1265, 1266, 1722, 1780, 1899, 1986	\g_nm_first_row_int	73, 477, 735, 1902, 1916, 1947, 2006
_nm_create_nodes:	1946, 1984, 2004	\l_nm_first_row_int	71, 72, 260, 269, 477, 536, 686, 1823, 2375, 2382, 2389, 2396, 2403, 2416
\l_nm_ddots_int	1103, 1455, 1456	_nm_gobble_ampersands:n	2076, 2095, 2100
\g_nm_Ddots_lines_tl	574, 1119	_nm_Hdotsfor:	520, 531, 1664
\l_nm_delta_x_one_dim	1105, 1458, 1468	_nm_Hdotsfor_i	1667, 1671, 1675
\l_nm_delta_x_two_dim	1107, 1500, 1510	\g_nm_Hdotsfor_lines_tl	576, 1117, 1677
\l_nm_delta_y_one_dim	1106, 1460, 1468	_nm_hdottedline:	518, 1756
\l_nm_delta_y_two_dim	1108, 1502, 1510	\l_nm_hlines_bool	128, 146, 439
_nm_dotfill:	1748, 1750, 1771	_nm_Hspace:	519, 1647
\g_nm_dp_ante_last_row_dim	296, 548, 708, 715, 716, 723, 751	\g_nm_ht_last_row_dim	298, 314, 315, 549, 708, 715
\g_nm_dp_last_row_dim	296, 297, 316, 317, 550, 716, 723, 751	\g_nm_ht_row_one_dim	311, 312, 547, 689, 694, 695, 700, 740
\g_nm_dp_row_zero_dim	304, 305, 545, 689, 694	\g_nm_ht_row_zero_dim	306, 307, 546, 695, 700, 740
\c_nm_draft_bool	11, 12, 34, 407, 1669, 1747, 1781	_nm_i:	1902, 1904, 1905, 1906, 1907, 1916, 1921, 1925, 1926, 1927, 1928, 1934, 1935, 1936, 1937, 1949, 1951, 1954, 1955, 1959, 1960, 2006, 2012, 2015, 2021, 2024, 2036, 2045, 2050, 2057, 2059
_nm_draw_Cdots:nn	1340	\l_nm_iddots_int	1104, 1497, 1498
_nm_draw_Ddots:nn	1434	\g_nm_Iddots_lines_tl	575, 1120
_nm_draw_Hdotsfor:nnn	1679, 1687	_nm_if_not_empty_cell:nn	1017
_nm_draw_Iddots:nn	1476	_nm_if_not_empty_cell:nnTF	1176, 1236, 1700, 1714
_nm_draw_Ldots:nn	1290	\l_nm_impossible_line_bool	60, 1201, 1262, 1294, 1296, 1344, 1346, 1391, 1393, 1438, 1440, 1480, 1482
_nm_draw_tikz_line:	1338, 1383, 1430, 1472, 1514, 1518, 1744, 1813	\l_nm_in_env_bool	51, 582, 583, 943, 944
_nm_draw_Vdots:nn	1387	\l_nm_initial_i_int	1111, 1140, 1207, 1210, 1231, 1237, 1243, 1254, 1261, 1277, 1309, 1356, 1691, 1700
_nm_end_Cell:	118, 319, 362, 504, 505, 506		
\g_nm_env_int	48, 336, 390, 456, 460, 587, 600, 603, 822, 889, 948, 1003, 1027, 1040, 1050, 1088, 1129, 1192, 1253, 1276, 1282, 1308, 1323, 1355, 1370, 1615, 1761, 1762, 1800, 1880, 1886, 1921, 1924, 1934, 1989, 1992, 1999, 2021, 2057		
_nm_error:n	18, 224, 228, 235, 244, 248, 252, 263, 272, 681, 1071, 1778		
_nm_error:nn	19, 187		
_nm_error:nnn	20		
_nm_everycr:	434, 489, 492		
_nm_everycr_i:	435, 436		
\l_nm_exterior_arraycolsep_bool	125, 231, 650, 659, 957		
\l_nm_extra_left_margin_dim	141, 161, 665, 838, 955		

<code>\l_nm_initial_j_int</code>	<code>_nm_old_multicolumn</code>
1112, 1141, 1208, 1216, 1222, 1232, 1237,	1652, 1655
1244, 1255, 1278, 1310, 1357, 1695, 1701, 1703	<code>\l_nm_parallelize_diags_bool</code>
<code>\l_nm_initial_open_bool</code> .. 1115, 1209, 126, 127, 147, 1101, 1453, 1495
1213, 1219, 1225, 1229, 1258, 1265, 1303,	<code>\l_nm_pos_env_str</code>
1332, 1350, 1379, 1397, 1410, 1444, 1486,	.. 123, 124, 255, 256, 257, 432, 737, 746, 956
1539, 1574, 1689, 1696, 1704, 1721, 1733, 1787	<code>_nm_pre_array:</code>
<code>_nm_instruction_of_type:n</code>	451, 614, 952
.... 408, 410, 1619, 1625, 1631, 1637, 1643	<code>\c_nm_preamble_first_col_tl</code> 646, 787
<code>_nm_j:</code>	<code>\c_nm_preamble_last_col_tl</code> 655, 844
1909, 1911,	<code>\l_nm_renew_dots_bool</code>
1912, 1913, 1914, 1919, 1921, 1925, 1928,	149, 230, 523
1930, 1931, 1934, 1937, 1939, 1940, 1962,	<code>_nm_renew_matrix:</code> 222, 225, 229, 2324
1964, 1968, 1970, 1974, 1975, 2008, 2011,	<code>_nm_renew_NC@rewrite@S:</code>
2014, 2021, 2024, 2037, 2044, 2049, 2057, 2059	109, 568
<code>\l_nm_l_dim</code> 1520, 1521, 1537, 1544,	<code>_nm_renewcolumn_type:nn</code> 352, 543, 544
1548, 1555, 1559, 1565, 1570, 1582, 1589, 1590	<code>_nm_retrieve_coords:nn</code>
<code>\l_nm_last_col_bool</code>	1268, 1289, 1301, 1348, 1395, 1408, 1442, 1484
82,	<code>\c_nm_revtext_bool</code>
259, 268, 654, 2344, 2350, 2356, 2362,	27, 29, 32, 423
2368, 2374, 2381, 2388, 2395, 2402, 2409, 2415	<code>\g_nm_right_delim_dim</code> .. 616, 620, 640, 873
<code>\g_nm_last_col_found_bool</code>	<code>\g_nm_right_margin_dim</code>
..... 83, 570, 781, 848, 1079 138, 473, 672, 963, 1769, 1982
<code>\g_nm_last_row_int</code>	<code>\l_nm_right_margin_dim</code> . 136, 157, 473, 874
. 79, 443, 475, 675, 679, 682, 744, 1081, 2140	<code>\g_nm_row_int</code> 286, 289, 295, 302, 309, 337,
<code>\l_nm_last_row_int</code>	343, 391, 397, 416, 441, 443, 535, 536, 679,
..... 77, 78, 261, 270, 289, 475, 595,	682, 823, 829, 890, 896, 1069, 1080, 1082,
602, 609, 704, 1827, 1834, 1842, 1852, 1859	1150, 1614, 1659, 1680, 1795, 1807, 1825,
<code>\g_nm_last_row_without_value_bool</code> ..	1827, 1832, 1834, 1840, 1842, 1850, 1852,
..... 81, 480, 677, 1083	1857, 1859, 1949, 2086, 2088, 2108, 2141, 2143
<code>\l_nm_last_row_without_value_bool</code> ..	<code>\g_nm_row_total_int</code>
..... 80, 481, 597 537, 1080, 1089, 1096, 1902, 1916, 2006
<code>\g_nm_last_vdotted_col_int</code>	<code>_nm_seq_mapthread_function:NNN</code> 2029, 2313
..... 559, 561, 567, 569	<code>\c_nm_siunitx_loaded_bool</code> . 84, 88, 93, 568
<code>\g_nm_ldots_lines_tl</code>	<code>\l_nm_stop_loop_bool</code>
572, 1122	1144, 1145, 1173, 1177, 1204, 1205, 1233, 1238
<code>\g_nm_left_delim_dim</code> ... 615, 619, 632, 836	<code>\c_nm_table_collect_begin_tl</code> 101, 103, 116
<code>\g_nm_left_margin_dim</code> 137, 472, 1979	<code>\c_nm_table_print_tl</code>
<code>\l_nm_left_margin_dim</code>	104, 105, 118
..... 135, 155, 472, 664, 837, 954, 1769	<code>_nm_test_if_math_mode:</code> . 54, 581, 912,
<code>\l_nm_letter_for_dotted_lines_str</code> ..	918, 924, 930, 936, 942, 968, 974, 980, 986, 992
..... 243, 552, 553, 2309	<code>\l_nm_the_array_box</code> 643, 663, 756, 769
<code>_nm_line:nn</code>	<code>\g_nm_Vdots_lines_tl</code>
1132, 1727	573, 1118
<code>\g_nm_max_cell_width_dim</code>	<code>_nm_vdottedline:n</code>
.. 323, 324, 589, 950, 1004, 1011, 1872, 1891	563, 1775
<code>_nm_msg_new:nn</code>	<code>_nm_vline:</code>
..... 23, 2126, 2132, 2138, 2146,	590, 1817
2148, 2154, 2160, 2165, 2170, 2295, 2300, 2307	<code>_nm_vline_i:</code>
<code>_nm_msg_new:nnn</code> 67, 1828, 1835, 1843, 1853, 1860, 1865
..... 24, 2175, 2200, 2229, 2251, 2281	<code>\g_nm_width_first_col_dim</code> 140, 731, 802, 805
<code>_nm_msg_redirect_name:nn</code>	<code>\g_nm_width_last_col_dim</code> 139, 783, 862, 865
25, 238	<code>_nm_write_max_cell_width:</code> 455, 996
<code>_nm_multicolumn:nnn</code>	<code>\g_nm_x_final_dim</code> 1272, 1285, 1407, 1420,
521, 1653	1426, 1428, 1459, 1467, 1501, 1509, 1527,
<code>\g_nm_multicolumn_cells_seq</code>	1564, 1580, 1731, 1741, 1785, 1796, 1808, 1811
..... 533, 1658, 1928, 1937, 2030	<code>\g_nm_x_initial_dim</code>
<code>\g_nm_multicolumn_sizes_seq</code> 534, 1660, 2031	1270,
<code>\g_nm_name_str</code> 132, 476, 598, 607,	1279, 1407, 1420, 1423, 1426, 1428, 1459,
610, 1006, 1010, 1091, 1095, 2023, 2024, 2059	1467, 1501, 1509, 1528, 1564, 1578, 1580,
<code>\l_nm_name_str</code> 131, 189, 340, 342,	1597, 1600, 1729, 1738, 1783, 1792, 1805, 1810
394, 396, 462, 464, 467, 476, 826, 828, 893, 895	<code>\g_nm_y_final_dim</code> ... 1273, 1286, 1333,
<code>\g_nm_names_seq</code>	1335, 1337, 1380, 1382, 1461, 1464, 1503,
50, 186, 188, 2293	1506, 1531, 1569, 1587, 1732, 1742, 1786, 1797
<code>\g_nm_NiceArray_bool</code>	<code>\g_nm_y_initial_dim</code>
53, 479, 733 1271, 1280, 1333, 1335, 1336, 1380,
<code>\l_nm_NiceArray_bool</code>	1382, 1461, 1466, 1503, 1508, 1532, 1569,
..... 52, 479, 591, 617, 648, 657, 906	1585, 1587, 1597, 1601, 1730, 1739, 1784, 1793
<code>_nm_node_for_multicolumn:nn</code> .. 2032, 2039	<code>\noalign</code>
<code>\l_nm_nullify_dots_bool</code>	435, 488, 1758
.... 129, 151, 1620, 1626, 1632, 1638, 1644	<code>\node</code> .. 333, 387, 817, 884, 2016, 2052, 2114, 2121
	<code>\nulldelimiterspace</code>
	624, 635
	P
	<code>\path</code>
	1736

peek commands:

<code>\peek_charcode_remove_ignore_spaces:NTF</code>	2099
<code>\pgfpathcircle</code>	1596
<code>\pgfpoint</code>	1597
<code>\pgfpointanchor</code>	1054, 1056
<code>\pgfusepath</code>	1599
<code>\phantom</code>	1620, 1626, 1632, 1638, 1644
<code>\pNiceArray</code>	2345, 2376
<code>\pNiceMatrix</code>	2327
prg commands:	
<code>\prg_do_nothing:</code>	97, 106, 488, 1748
<code>\prg_replicate:nn</code>	1672, 1684
<code>\prg_return_false:</code>	1031, 1044, 1061
<code>\prg_return_true:</code>	1022, 1062
<code>\prg_set_conditional:Npnn</code>	1017
<code>\ProcessKeysOptions</code>	2125
<code>\ProcessOptions</code>	14
<code>\ProvideDocumentCommand</code>	36
<code>\ProvidesExplPackage</code>	6

Q

quark commands:

<code>\q_stop</code>	2034, 2041, 2065, 2067
----------------------	------------------------

R

<code>\raise</code>	41, 43, 45
<code>\relax</code>	14, 541, 542
<code>\renewcommand</code>	111
<code>\RenewDocumentEnvironment</code>	2326, 2329, 2332, 2335, 2338
<code>\RequirePackage</code>	2, 4, 5, 15, 16, 17
<code>\right</code>	628, 637, 774, 971, 977, 983, 989, 995
<code>\rVert</code>	995
<code>\rvert</code>	989

S

seq commands:

<code>\seq_count:N</code>	2316
<code>\seq_gclear_new:N</code>	533, 534
<code>\seq_gput_left:Nn</code>	188, 1658, 1660
<code>\seq_if_in:NnTF</code>	186, 1928, 1937
<code>\seq_new:N</code>	50
<code>\seq_pop:NN</code>	2318, 2319
<code>\seq_use:Nnnn</code>	2293

skip commands:

<code>\skip_horizontal:n</code>	557, 664, 665, 672, 673, 730, 731, 768, 770, 783, 784, 834, 841, 868, 871, 953, 954, 955, 962, 963, 964, 1753
<code>\skip_vertical:n</code>	446, 765, 772
<code>\c_zero_skip</code>	493, 497

str commands:

<code>\c_colon_str</code>	247
<code>\str_if_empty:NTF</code>	340, 394, 462, 598, 826, 893, 1006, 1091, 2023, 2059
<code>\str_if_eq:nnTF</code>	180, 234, 737, 746
<code>\str_new:N</code>	123, 131, 132
<code>\str_set:Nn</code>	124, 185, 243, 255, 256, 257, 956
<code>\str_set_eq:NN</code>	189
<code>\l_tmpa_str</code>	185, 186, 188, 189

T

<code>\tabskip</code>	493, 497
-----------------------	----------

TeX and L^AT_EX 2_ε commands:

<code>\@acol</code>	427
<code>\@acoll</code>	425
<code>\@acolr</code>	426
<code>\@addtopreamble</code>	590
<code>\@array@array</code>	429
<code>\@arrayacol</code>	425, 426, 427
<code>\@arrayrule</code>	590
<code>\@currenvir</code>	2141, 2156, 2254, 2302, 2310
<code>\@halignto</code>	428
<code>\@height</code>	445
<code>\@ifclassloaded</code>	28, 31
<code>\@ifnextchar</code>	540
<code>\@ifpackageloaded</code>	64, 87
<code>\@mainaux</code>	1000, 1001, 1008, 1014, 1085, 1086, 1093, 1099, 1885, 1888, 1894, 1996, 1997, 2002
<code>\@temptokena</code>	96, 99, 113, 115
<code>\c@MaxMatrixCols</code>	958
<code>\CT@arc@</code>	67
<code>\CT@everycr</code>	486
<code>\CT@row@color</code>	488
<code>\NC@find</code>	97, 120
<code>\NC@find@W</code>	542
<code>\NC@find@w</code>	541
<code>\NC@rewrite@S</code>	98, 111
<code>\new@ifnextchar</code>	540
<code>\p@</code>	41, 43, 45
<code>\pgf@x</code>	1055, 1057, 1279, 1285, 1738, 1741, 1792, 1796, 1805, 1808, 1931, 1940, 1990, 1994
<code>\pgf@y</code>	1280, 1286, 1739, 1742, 1793, 1797, 1927, 1936
<code>\pgfutil@firstofone</code>	1275, 1281, 1737, 1740, 1790, 1794, 1803, 1806, 1923, 1933, 1988, 1991
<code>\tikz@library@external@loaded</code>	584, 945, 1076
<code>\tikz@parse@node</code>	1275, 1281, 1737, 1740, 1790, 1794, 1803, 1806, 1923, 1933, 1988, 1991
tex commands:	
<code>\tex_the:D</code>	115
<code>\tikz</code>	326, 375, 380, 386, 810, 877
<code>\tikzset</code>	585, 946, 1077, 1123, 1945, 1983
tl commands:	
<code>\c_empty_tl</code>	192
<code>\tl_const:Nn</code>	787, 844
<code>\tl_count:n</code>	242
<code>\tl_gclear:N</code>	1134
<code>\tl_gclear_new:N</code>	571, 572, 573, 574, 575, 576
<code>\tl_gput_left:Nn</code>	2083
<code>\tl_gput_right:Nn</code>	412, 562, 1677
<code>\tl_gset:Nn</code>	99, 103, 105
<code>\tl_gset_eq:NN</code>	476
<code>\tl_item:Nn</code>	102, 103, 105
<code>\tl_new:N</code>	101, 104
<code>\tl_put_left:Nn</code>	646, 651
<code>\tl_put_right:Nn</code>	655, 660
<code>\tl_set:Nn</code>	102, 644, 1047
<code>\tl_set_rescan:Nnn</code>	551
<code>\tl_use:N</code>	2172, 2177, 2202, 2231, 2253
<code>\g_tmpa_tl</code>	99, 102, 105
<code>\l_tmpa_tl</code>	102, 103, 644, 646, 651, 655, 660, 667, 1047, 1054, 1056, 2318, 2320
<code>\l_tmpb_tl</code>	2319, 2320

14	Implementation	21
14.1	Declaration of the package and extensions loaded	21
14.2	Technical definitions	22
14.2.1	Variables for the exterior rows and columns	24
14.2.2	The column S of siunitx	25
14.3	The options	26
14.4	Code common to <code>{NiceArrayWithDelims}</code> and <code>{NiceMatrix}</code>	31
14.5	The environment <code>{NiceArrayWithDelims}</code>	38
14.6	The environment <code>{NiceMatrix}</code> and its variants	45
14.7	Automatic width of the cells	47
14.8	How to know whether a cell is “empty”	47
14.9	After the construction of the array	48
14.10	The actual instructions for drawing the dotted line with Tikz	58
14.11	User commands available in the new environments	60
14.12	The command <code>\line</code> accessible in code-after	63
14.13	The commands to draw dotted lines to separate columns and rows	63
14.14	The vertical rules	65
14.15	The environment <code>{NiceMatrixBlock}</code>	66
14.16	The extra nodes	67
14.17	Block matrices	71
14.18	We process the options	72
14.19	Error messages of the package	72
14.20	Code for <code>\seq_mapthread_function:NNN</code>	75
14.21	Obsolete environments	76
15	History	77
	Index	79