

The package **nicematrix**^{*}

F. Pantigny
fpantigny@wanadoo.fr

January 28, 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;
- a first row and a last column for labels;
- a control of the width of the columns.

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

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

An example for the continuous dotted lines

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

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

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

This code composes the matrix A on the right.

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

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

^{*}This document corresponds to the version 2.1.4 of **nicematrix**, at the date of 2019/01/28.

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>{pNiceArrayC}</code>	<code>{pNiceArrayRC}</code>
<code>{pNiceMatrix}</code>		<code>{bNiceArrayC}</code>	<code>{bNiceArrayRC}</code>
<code>{bNiceMatrix}</code>		<code>{BNiceArrayC}</code>	<code>{BNiceArrayRC}</code>
<code>{BNiceMatrix}</code>		<code>{vNiceArrayC}</code>	<code>{vNiceArrayRC}</code>
<code>{vNiceMatrix}</code>		<code>{VNiceArrayC}</code>	<code>{VNiceArrayRC}</code>
<code>{VNiceMatrix}</code>		<code>{NiceArrayCwithDelims}</code>	<code>{NiceArrayRCwithDelims}</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{...}{...}`, `|`, `>{...}`, `<{...}`, `@{...}`, `!{...}` and `*{n}{...}` but the letters p, m and b should not be used. See p. 7 the section relating to `{NiceArray}`.

The environments with C at the end of their name, `{pNiceArrayC}`, `{bNiceArrayC}`, `{BNiceArrayC}`, `{vNiceArrayC}` and `{VNiceArrayC}` are similar to the environment `{NiceArray}` (especially the special letters L, C and R) but create an exterior column (on the right of the closing delimiter). See p. 8 the section relating to `{pNiceArrayC}`.

The environments with RC, `{pNiceArrayRC}`, `{bNiceArrayRC}`, `{BNiceArrayRC}`, `{vNiceArrayRC}`, `{VNiceArrayRC}` are similar to the environment `{NiceArray}` but create an exterior row (above the main matrix) and an exterior column. See p. 8 the section relating to `{pNiceArrayRC}`.

3 The continuous dotted lines

Inside the environments of the extension `nicematrix`, new commands are defined: `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, and `\Iddots`.² These commands are intended to be used in place of `\dots`, `\cdots`, `\vdots`, `\ddots` and `\iddots`.

Each of them must be used alone in the cell of the array and it draws a dotted line between the first non-empty cells³ 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}
```

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

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}
```

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

²The command `\iddots`, defined in `nicematrix`, is a variant of `\ddots` with dots going forward: \cdots . If `mathdots` is loaded, the version of `mathdots` is used. It corresponds to the command `\adots` of `unicode-math`.

³The precise definition of a “non-empty cell” is given below (cf. p. 11).

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 PDF⁴).

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

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

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

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

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

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

There are also other means to change the size of the matrix. Someone might want to use the optional argument of the command `\Vdots` for the vertical dimension and a command `\hspace*` in a cell for the horizontal dimension.⁵

However, a command `\hspace*` might interfer 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}
```

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

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

⁵ Nevertheless, the best way to fix the width of a column is to use the environment `{NiceArray}` with a column of type `w` (or `W`).

3.1 The option `nullify-dots`

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

```
$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}
```

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

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

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

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

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).

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

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

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.

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

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

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

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

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

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` is compatible with the extension `colortbl`.

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`⁶ 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 & \cdots & 1 \\
0 & \ddots & & & \\
\vdots & \ddots & \ddots & \ddots & \\
0 & \cdots & 0 & \cdots & 1
\end{pmatrix}
```

⁶The command `\iddots` is not a command of LaTeX but is defined by the package `nicematrix`. If `mathdots` is loaded, the version of `mathdots` is used.

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.

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

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

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

⁷In the environments like `{pNiceArrayC}` and `{pNiceArrayRC}`, there is not “large nodes” created in the exterior row and column.

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`.

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”. In the following example, we have used `extra-left-margin` and `extra-right-margin` with the value 3 pt.

$$\left(\begin{array}{c|cc|c} a & a+b & a+b+c \\ a & a & a+b \\ \hline a & a & a \end{array} \right)$$

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.

$$\left(\begin{array}{c|cc|c} a & a+b & a+b+c \\ a & a & a+b \\ \hline a & a & a \end{array} \right)$$

We explain below how to fill the nodes created by `nicematrix`.

5 The code-after

The option `code-after` may be used to give some code that will be executed 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}
```

$$\left(\begin{array}{ccc} 0 & \cdots & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \cdot 0 \end{array} \right)$$

6 The environment `{NiceArray}`

The environment `{NiceArray}` is similar to the environment `{array}`. As for `{array}`, the mandatory argument is the preamble of the array. However, for technical reasons, in this preamble, the user must use the letters `L`, `C` and `R`⁹ instead of `l`, `c` and `r`. It's possible to use the constructions `w{...}{...}`, `W{...}{...}`, `|`, `>{...}`, `<{...}`, `@{...}`, `!{...}` and `*{n}{...}` but the letters `p`, `m` and `b` should not be used.¹⁰

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

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

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

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

⁹The column types `L`, `C` and `R` are defined locally inside `{NiceArray}` with `\newcolumntype` of `array`. This definition overrides an eventual previous definition.

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

An example where we use `{NiceArray}` because we want to use the types L and R for the columns:

```
$\left(\begin{array}{c|ccccc} a_{11} & \cdots & \cdots & \cdots & a_{1n} \\ a_{21} & & & & a_{2n} \\ \vdots & & & & \vdots \\ a_{n-1,1} & \cdots & \cdots & \cdots & a_{n-1,n} \\ \end{array}\right)
```

$$\left(\begin{array}{cccc|c} a_{11} & \cdots & \cdots & \cdots & a_{1n} \\ a_{21} & & & & a_{2n} \\ \vdots & & & & \vdots \\ a_{n-1,1} & \cdots & \cdots & \cdots & a_{n-1,n} \end{array} \right)$$

7 The environment `{pNiceArrayC}` and its variants

The environment `{pNiceArrayC}` composes a matrix with an exterior column.

The environment `{pNiceArrayC}` takes a mandatory argument which is the preamble of the array. The types of columns available are the same as for the environment `{NiceArray}`. **However, no specification must be given for the last column.** It will automatically (and necessarily) be a L column.

A special option, called `code-for-last-col`, specifies tokens that will be inserted before each cell of the last column. The option `columns-width` doesn't apply to this external column.

```
$\begin{pNiceArrayC}{*6C|C}[nullify-dots,code-for-last-col={\scriptstyle}]\\
1 & 1 & 1 & \cdots & 1 & \\ 
0 & 1 & 0 & \cdots & 0 & \scriptstyle L_2 \gets L_2 - L_1 \\ 
0 & 0 & 1 & \ddots & \vdots & \scriptstyle L_3 \gets L_3 - L_1 \\ 
& & & \ddots & \vdots & \\ 
\vdots & & & \ddots & 0 & \\ 
0 & & & \ddots & 0 & \scriptstyle L_n \gets L_n - L_1 \\ 
\end{pNiceArrayC}$
```

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

In fact, the environment `{pNiceArrayC}` and its variants are based upon a more general environment, called `{NiceArrayCwithDelims}`. 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 `{NiceArrayCwithDelims}` if we want to use atypical delimiters.

```
$\begin{NiceArrayCwithDelims}{\downarrow\downarrow}{\downarrow\downarrow}{CCC}\\
1 & 2 & 3 & L_1 \\ 
4 & 5 & 6 & L_2 \\ 
7 & 8 & 9 & L_3 \\ 
\end{NiceArrayCwithDelims}$
```

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

8 The environment `{pNiceArrayRC}` and its variants

The environment `{pNiceArrayRC}` composes a matrix with an exterior row and an exterior column. This environment `{pNiceArrayRC}` takes a mandatory argument which is the preamble of the array. As for the environment `{pNiceArrayC}`, no specification must be given for the last column (it will automatically be a L column).

A special option, called `code-for-first-row`, specifies tokens that will be inserted before each cell of the first row.

```
$\begin{pNiceArrayRC}{CCC}% (here, % is mandatory)
[columns-width = auto,
 code-for-first-row = \color{blue},
 code-for-last-col = \color{blue}]
C_1 & C_2 & C_3 \\
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{pNiceArrayRC}$
```

$$\begin{pmatrix} C_1 & C_2 & C_3 \\ 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \begin{matrix} L_1 \\ L_2 \\ L_3 \end{matrix}$$

The first row of an environment `{pNiceArrayRC}` has the number 0, and not 1. 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`).

For technical reasons, it's not possible to use the option of the command `\backslash` after the first row (the placement of the delimiters would be wrong).

In fact, the environment `{pNiceArrayRC}` and its variants are based upon an more general environment, called `{NiceArrayRCwithDelims}`. 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 `{NiceArrayRCwithDelims}` if we want to use atypical delimiters.

```
$\begin{NiceArrayRCwithDelims}
{\downarrow}{\downarrow}{CCC}[columns-width=auto]
C_1 & C_2 & C_3 \\
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{NiceArrayRCwithDelims}$
```

$$\begin{array}{ccc|c} C_1 & C_2 & C_3 \\ 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{array}$$

If we want to write a linear system, we can use the following code, with a preamble `CCC|C`:

```
$\begin{pNiceArrayRC}{CCC|C}
C_1 & \cdots & C_n \\
a_{11} & \cdots & a_{1n} & b_1 \\
\vdots & & \vdots & \vdots \\
a_{n1} & \cdots & a_{nn} & b_n \\
\end{pNiceArrayRC}$
```

$$\left(\begin{array}{ccc|c} C_1 & \cdots & C_n & b_1 \\ a_{11} & \cdots & a_{1n} & b_1 \\ \vdots & & \vdots & \vdots \\ a_{n1} & \cdots & a_{nn} & b_n \end{array} \right)$$

The resultat may seem disappointing. It's possible to suppress the vertical rule in the first row with the command `\multicolumn` in order to “reconstruct” the cell.

```
$\begin{pNiceArrayRC}{CCC|C}
C_1 & \cdots & C_n \\
a_{11} & \cdots & a_{1n} & b_1 \\
\vdots & & \vdots & \vdots \\
a_{n1} & \cdots & a_{nn} & b_n \\
\end{pNiceArrayRC}$
```

$$\left(\begin{array}{cc|c} C_1 & \cdots & C_n \\ a_{11} & \cdots & a_{1n} & b_1 \\ \vdots & & \vdots & \vdots \\ a_{n1} & \cdots & a_{nn} & b_n \end{array} \right)$$

On the other side, we may remark that an horizontal line (with `\hline` or `\hdashline` of `arydshln`) doesn't extend in the “exterior column” of an environment like `{pNiceArrayC}` or `{pNiceArrayRC}`.

```
$\begin{pNiceArrayC}{CCC}
a_{11} & \cdots & a_{1n} & L_1 \\
\vdots & & \vdots & \vdots \\
a_{n1} & \cdots & a_{nn} & L_n \\
\hdashline
S_1 & \cdots & S_n \\
\end{pNiceArrayC}$
```

$$\left(\begin{array}{cc|c} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \\ \hline S_1 & \cdots & S_n \end{array} \right) \begin{matrix} L_1 \\ \vdots \\ L_n \end{matrix}$$

9 The width of the columns

In the environments with an explicit preamble (like `{NiceArray}`, `{pNiceArrayC}`, `{pNiceArrayRC}`, 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{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}[\text{columns-width} = 1cm]
```

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

Note that the space inserted between two columns (equal to `2 \arraycolsep`) is not suppressed.

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}[\text{columns-width} = \text{auto}]
```

```
1 & 12 & -123 \\
```

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

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

It's possible to fix the width of the columns of all the matrices of a current scope with the command `\NiceMatrixOptions`.

```
\NiceMatrixOptions{\text{columns-width}=10mm}
```

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

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

```
\end{pNiceMatrix}
```

```
=
```

```
\begin{pNiceMatrix}
```

```
1 & 1245 \\ 345 & 2 \\
```

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

$$\left(\begin{array}{cc} a & b \\ c & d \end{array} \right) = \left(\begin{array}{cc} 1 & 1245 \\ 345 & 2 \end{array} \right)$$

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`.¹¹

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

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

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

```
\end{pNiceMatrix}
```

```
=
```

```
\begin{pNiceMatrix}
```

```
1 & 1245 \\ 345 & 2 \\
```

```
\end{pNiceMatrix}
```

```
\end{NiceMatrixBlock}
```

$$\left(\begin{array}{cc} a & b \\ c & d \end{array} \right) = \left(\begin{array}{cc} 1 & 1245 \\ 345 & 2 \end{array} \right)$$

¹¹At this time, this is the only usage of the environment `{NiceMatrixBlock}` but it may have other usages in the future.

10 Technical remarks

10.1 Diagonal lines

By default, all the diagonal lines¹² of a same array are “parallelized”. That means that the first diagonal line is drawn and, then, the other lines are drawn parallel to the first one (by rotation around the left-most extremity of the line). That’s why the position of the instructions `\Ddots` in the array can have a marked effect on the final result.

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

Example with parallelization (default):

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

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

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

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

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

The same example without parallelization:

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

10.2 The “empty” cells

An instruction like `\Ldots`, `\Cdots`, etc. tries to determine the first non-empty cells on both sides. However, a 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.

¹²We speak of the lines created by `\Ddots` and not the lines created by a command `\line` in `code-after`.

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

10.3 The option `exterior-arraycolsep`

The environment `{array}` inserts an horizontal space equal to `\arraycolsep` before and after each column. In particular, there is a space equal to `\arraycolsep` before and after the array. This feature of the environment `{array}` was probably not a good idea.¹³

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`.

10.4 The class option `draft`

The package `nicematrix` is rather slow when drawing the dotted lines (generated by `\Cdots`, `\Ldots`, `\Ddots`, etc.).¹⁴

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

10.5 A technical problem with the argument of `\backslash`

For technical reasons, if you use the optional argument of the command `\backslash`, 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}{}
\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}{}} \\ 
b & c
\end{pNiceMatrix} 
$$\begin{pmatrix} a & \frac{A}{B} \\ b & c \end{pmatrix}$$

```

¹³In 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}{@{}cccccc@{}}`.

¹⁴The 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 systeme of dotted lines.

10.6 A remark concerning a bug of Tikz

Due to a bug in Tikz, the construction `-- cycle` in a Tikz path is incompatible with the use of `name prefix` and `name suffix`.¹⁵

Since `name prefix` is implicitly used in the `code-after` of `nicematrix`, it's not possible to use `-- cycle` in `code-after`.

10.7 Compatibility with the extension `dcolumn`

If we want to make `nicematrix` compatible with `dcolumn`, it's necessary to patch the commands `\DC@endcentre` and `\DC@endright` as follow.

```
\def\DC@endcentre{$\egroup
\ifdim \wd\z@>\wd\tw@
\setbox\tw@=\hbox to\wd\z@{\unhbox\tw@\hfill}%
\else
\setbox\z@=\hbox to\wd\tw@{\hfill\unhbox\z@}\fi
\@@_Cell:\box\z@\box\tw@ \@@_end_Cell:}

\def\DC@endright{$\hfil\egroup \@@_Cell:\box\z@\box\tw@ \@@_end_Cell:}
```

11 Examples

11.1 Dotted lines

A tridiagonal matrix:

```
$\begin{pNiceMatrix}[nullify-dots]
a & b & 0 & & \cdots & 0 \\ 
b & a & b & & \vdots & \\ 
0 & b & a & & \ddots & \\ 
& \ddots & \ddots & \ddots & & \\ 
& \vdots & & & & \\ 
0 & & \cdots & 0 & b & a
\end{pNiceMatrix}$
```

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

A permutation matrix:

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

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

¹⁵cf. tex.stackexchange.com/questions/327007/tikz-fill-not-being-drawn-using-named-coordinates

An example with `\Iddots`:

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

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

An example with `\multicolumn`:

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

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

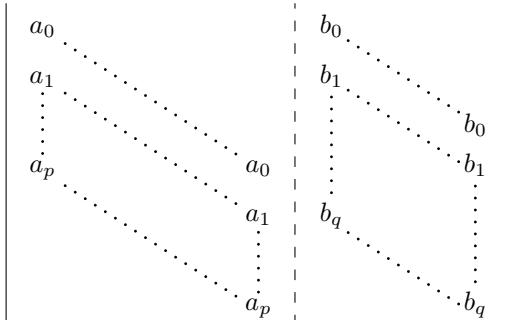
An example with `\Hdotsfor`:

```
\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}
```

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

An example for the resultant of two polynomials (the dashed line has been drawn with `arydshln`):

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



11.2 Width of the columns

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

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

$$\left(\begin{array}{cccc|c} 1 & 1 & 1 & 1 & 1 \\ 2 & 4 & 8 & 16 & 9 \\ 3 & 9 & 27 & 81 & 36 \\ 4 & 16 & 64 & 256 & 100 \end{array} \right) \quad \left(\begin{array}{cccc|c} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 3 & 7 & \frac{7}{2} \\ 0 & 0 & 3 & 18 & 6 \\ 0 & 0 & -2 & -14 & -\frac{9}{2} \end{array} \right) \begin{matrix} L_3 \leftarrow -3L_2 + L_3 \\ L_4 \leftarrow L_2 - L_4 \end{matrix}$$

$$\left(\begin{array}{cccc|c} 1 & 1 & 1 & 1 & 1 \\ 0 & 2 & 6 & 14 & 7 \\ 0 & 6 & 24 & 78 & 33 \\ 0 & 12 & 60 & 252 & 96 \end{array} \right) \begin{matrix} L_2 \leftarrow -2L_1 + L_2 \\ L_3 \leftarrow -3L_1 + L_3 \\ L_4 \leftarrow -4L_1 + L_4 \end{matrix} \quad \left(\begin{array}{cccc|c} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 3 & 7 & \frac{7}{2} \\ 0 & 0 & 1 & 6 & 2 \\ 0 & 0 & -2 & -14 & -\frac{9}{2} \end{array} \right) \begin{matrix} L_3 \leftarrow \frac{1}{3}L_3 \end{matrix}$$

$$\left(\begin{array}{cccc|c} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 3 & 7 & \frac{7}{2} \\ 0 & 3 & 12 & 39 & \frac{33}{2} \\ 0 & 1 & 5 & 21 & 8 \end{array} \right) \begin{matrix} L_2 \leftarrow \frac{1}{2}L_2 \\ L_3 \leftarrow \frac{1}{2}L_3 \\ L_4 \leftarrow \frac{1}{12}L_4 \end{matrix} \quad \left(\begin{array}{cccc|c} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 3 & 7 & \frac{7}{2} \\ 0 & 0 & 1 & 6 & 2 \\ 0 & 0 & 0 & -2 & -\frac{1}{2} \end{array} \right) \begin{matrix} L_4 \leftarrow 2L_3 + L_4 \end{matrix}$$

11.3 How to highlight cells of the matrix

In order to highlight a cell of a matrix, it's possible to “draw” one of the correspond 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`.

```
$\left(\begin{array}{>{\hspace{-\pgflinewidth/2}}c<{\hspace{-\pgflinewidth/2}}|c}
\begin{array}{cccc}
\begin{tikzpicture}[baseline=0, remember picture, overlay]
\draw[thick] (0,0) -- (1,0) -- (2,0) -- (3,0);
\draw[thick] (0,0) -- (0,1) -- (0,2) -- (0,3);
\end{tikzpicture}
& 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 2 & 4 & 8 & 16 \\ 3 & 9 & 27 & 81 \\ 4 & 16 & 64 & 256 \end{array} | c)
```

```

[name suffix = -large,
 every node/.style = {draw,
   inner sep = -\pgflinewidth/2}]
 \node [fit = (1-1)] {} ;
 \node [fit = (2-2)] {} ;
 \node [fit = (3-3)] {} ;
 \node [fit = (4-4)] {} ;
 \end{tikzpicture}]}
a_{11} & a_{12} & a_{13} & a_{14} \\
a_{21} & a_{22} & a_{23} & a_{24} \\
a_{31} & a_{32} & a_{33} & a_{34} \\
a_{41} & a_{42} & a_{43} & a_{44}
\end{NiceArray}\right],\right)\right]

```

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

The package **nicematrix** is constructed upon the environment **{array}** and, therefore, it's possible to use the package **colortbl** in the environments of **nicematrix**.

```

$\begin{bNiceMatrix}
0 & \cdots & 0 \\
\rowcolor{red!15} 1 & \cdots & 1 \\
0 & \cdots & 0 \\
\end{bNiceMatrix}$

```

The result may be disappointing. We therefore 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}}


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

```

$$\left[\begin{array}{ccc} 0 & \cdots & 0 \\ 1 & \cdots & 1 \\ 0 & \cdots & 0 \end{array} \right]$$

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 \pgf@sys@blend@mode#1{\special{ps:~/\tl_upper_case:n #1~.setblendmode}}}
\makeatother
\ExplSyntaxOff
```

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

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

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

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.

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

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.

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

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

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

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

11.4 Block matrices

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

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

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

12 Implementation

By default, the package `nicematrix` doesn’t patch any existing code.¹⁶

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.

¹⁶If we want `nicematrix` compatible with `dcolumn`, we have to patch `dcolumn`: cf. p. 13.

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.

12.1 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 \RequirePackage{tikz}
2 \usetikzlibrary{fit}
3 \RequirePackage{expl3}[2018-01-01]
```

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

```
4 \RequirePackage{l3keys2e}
5 \ProvidesExplPackage
6   {nicematrix}
7   {\myfiledate}
8   {\myfileversion}
9   {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.

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

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

We load `array` and `amsmath`.

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

12.2 Technical definitions

```
17 \cs_new_protected:Nn \c_@@_error:n
18   {\msg_error:nn {nicematrix} {#1}}
19 \cs_new_protected:Nn \c_@@_error:nn
20   {\msg_error:nn {nicematrix} {#1} {#2}}
21 \cs_new_protected:Nn \c_@@_bool_new:N
22   {\bool_if_exist:NTF #1
23    {\bool_set_false:N #1}
24    {\bool_new:N #1}}
```

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

```
25 \ProvideDocumentCommand \iddots {}
26   {\mathinner{\mkern 1mu
27     \raise \p@ \hbox{.}}
28   \mkern 2mu
29   \raise 4\p@ \hbox{.}}
30   \mkern 2mu
31   \raise 7\p@ \vbox{\kern 7pt
32                           \hbox{.}}}
```

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

33 \mkern 1mu}}

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.

34 \int_new:N \g_@@_env_int

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

35 \dim_new:N \l_@@_columns_width_dim

The sequence \g_@@_names_seq will be the list of all the names of environments used (via the option name) in the document: two environments must not have the same name.

36 \seq_new:N \g_@@_names_seq

The integer \l_@@_nb_first_row_int is the number of the first row of the array. The default value is 1, but, in the environments like {pNiceArrayRC}, the value will be 0.

37 \int_new:N \l_@@_nb_first_row_int

38 \int_set:Nn \l_@@_nb_first_row_int 1

The flag \l_@@_exterior_column_bool will indicate if we are in an environment of the type of {pNiceArrayC} or {pNiceArrayRC}. It will be used for the creation of the “large nodes”.

39 \bool_new:N \l_@@_exterior_column_bool

12.3 The options

The token list \l_@@_pos_env_tl 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}, {pNiceArrayC}, {pNiceArrayRC} 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.

40 \tl_new:N \l_@@_pos_env_tl

41 \tl_set:Nn \l_@@_pos_env_tl 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} (but neither for {NiceMatrix}, {pNiceArrayC}, {pNiceArrayRC} and their variants even if these environments rely upon {NiceArray}).

42 \bool_new:N \l_@@_exterior_arraycolsep_bool

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

43 \bool_new:N \l_@@_parallelize_diags_bool

44 \bool_set_true:N \l_@@_parallelize_diags_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 \phantom (and so the constructed matrix has exactly the same size as a matrix constructed with the classical {matrix} and \ldots, \vdots, etc.)

45 \bool_new:N \l_@@_nullify_dots_bool

The following flag will be used when the current options specify that all the columns of the array must have the same width equal to the largest width of a cell of the array (except the cell of the “exterior column” of an environment of the kind of {pNiceArrayC}).

46 \bool_new:N \l_@@_auto_columns_width_bool

The token list `\l_@@_code_for_last_col_tl` will contain code inserted at the beginning of each cell of the last column in the environment `{pNiceArrayC}` (and its variants). It corresponds to the option `code-for-last-col`.

```
47 \tl_new:N \l_@@_code_for_last_col_tl
```

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_tl` and `\g_@@_name_tl`.

The token list `\l_@@_name_tl` 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.

```
48 \tl_new:N \g_@@_name_tl
49 \tl_new:N \l_@@_name_tl
```

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

```
50 \bool_new:N \l_@@_extra_nodes_bool
51 \bool_new:N \g_@@_extra_nodes_bool
```

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

```
52 \dim_new:N \l_@@_left_margin_dim
53 \dim_new:N \l_@@_right_margin_dim
54 \dim_new:N \g_@@_left_margin_dim
55 \dim_new:N \g_@@_right_margin_dim
```

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

```
56 \dim_new:N \l_@@_extra_left_margin_dim
57 \dim_new:N \l_@@_extra_right_margin_dim
58 \dim_new:N \g_@@_extra_right_margin_dim
```

We define a set of options which will be used with the command `NiceMatrixOptions`.¹⁸

```
59 \keys_define:nn {NiceMatrix/NiceMatrixOptions}
60   {parallelize-diags .bool_set:N = \l_@@_parallelize_diags_bool,
61    parallelize-diags .default:n = true,
62    ParallelizeDiagonals .meta:n = parallelize-diags,
```

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

```
63 renew-dots .bool_set:N = \l_@@_renew_dots_bool,
64 renew-dots .default:n = true,
65 RenewDots .meta:n = renew-dots,
```

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

```
66 renew-matrix .code:n = \@@_renew_matrix:,,
67 renew-matrix .value_forbidden:n = true,
68 RenewMatrix .meta:n = renew-matrix,
69 transparent .meta:n = {renew-dots,renew-matrix},
70 transparent .value_forbidden:n = true,
71 Transparent .meta:n = transparent,
```

¹⁸Before the version 1.3, the names of the options were in “camel-case style” (like `ParallelizeDiagonals`) which was not a good idea. In version 1.4, the names are converted in lowercase with hyphens (like `parallelize-diags`). For compatibility, the old names are conversed.

Without the option `nullify-dots`, 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.). This option is set by default.

```
72     nullify-dots      .bool_set:N = \l_@@_nullify_dots_bool ,
73     nullify-dots      .default:n = true,
74     NullifyDots       .meta:n    = nullify-dots,
```

The following option is only for the environment `{pNiceArrayC}` and its variants. It will contain code inserted at the beginning of each cell of the last column.¹⁹

```
75     code-for-last-col .tl_set:N      = \l_@@_code_for_last_col_tl,
76     code-for-last-col .value_required:n = true,
```

Idem for the first row in environments like `{pNiceArrayRC}`.

```
77     code-for-first-row .tl_set:N      = \l_@@_code_for_first_row_tl,
78     code-for-first-row .value_required:n = true,
```

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

```
79     exterior-arraycolsep .bool_set:N = \l_@@_exterior_arraycolsep_bool ,
80     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.

```
81     columns-width      .code:n      = \str_if_eq:nnTF {#1} {auto}
82                                         {\l_@@_error:n {Option~auto~for~columns-width}}
83                                         {\dim_set:Nn \l_@@_columns_width_dim {#1}},

84     create-extra-nodes .bool_set:N   = \l_@@_extra_nodes_bool,
85     create-extra-nodes .default:n   = true,

86     left-margin        .dim_set:N   = \l_@@_left_margin_dim,
87     left-margin        .default:n   = \arraycolsep,
88     right-margin       .dim_set:N   = \l_@@_right_margin_dim,
89     right-margin       .default:n   = \arraycolsep,

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

91 \msg_new:nnn {nicematrix}
92             {Unknown~key~for~NiceMatrixOptions}
93             {The~key~"\tl_use:N\l_keys_key_tl"~is~unknown~for~the~command~
94              \token_to_str:N \NiceMatrixOptions.\\
95              If~you~go~on,~it~will~be~ignored.\\
96              For~a~list~of~the~available~keys,~type~H~<return>.}
97             {The~available~keys~are~(in~alphabetic~order):~
98               code-for-last-col,~
99               exterior-arraycolsep,~
100              left-margin,~
101              nullify-dots,~
102              parallelize-diags,~
103              renew-dots,~
104              renew-matrix,~
105              right-margin,~
106              and-transparent}

107 \msg_new:nnn {nicematrix}
108             {Option~auto~for~columns-width}
109             {You~can't~give~the~value~"auto"~to~the~option~"columns-width"~here.~
110              If~you~go~on,~the~option~will~be~ignored.}
```

¹⁹In an environment `{pNiceArrayC}`, the last column is composed outside the parentheses of the array.

`\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.

```

111 \NewDocumentCommand \NiceMatrixOptions {m}
112   {\keys_set:nn {NiceMatrix/NiceMatrixOptions} {#1}}
113
114 \keys_define:nn {NiceMatrix/NiceMatrix}
115   {parallelize-diags .bool_set:N = \l_@@_parallelize_diags_bool,
116    parallelize-diags .default:n = true,
117    renew-dots .bool_set:N = \l_@@_renew_dots_bool,
118    renew-dots .default:n = true,
119    nullify-dots .bool_set:N = \l_@@_nullify_dots_bool ,
120    nullify-dots .default:n = true,
121
122
123
124 \keys_define:nn {NiceMatrix/NiceMatrix}
125   {columns-width .code:n = \str_if_eq:nnTF {#1} {auto}
126     {\bool_set_true:N
127      \l_@@_auto_columns_width_bool
128      \dim_set:Nn \l_@@_columns_width_dim {#1}},
129     name .code:n = {\seq_if_in:NnTF \g_@@_names_seq {#1}
130       {\@_error:nn {Duplicate-name} {#1}}
131       {\seq_gput_left:Nn \g_@@_names_seq {#1}
132         \tl_set:Nn \l_@@_name_tl {#1}}},
133     name .value_required:n = true,
134     code-after .tl_set:N = \l_@@_code_after_tl,
135     code-after .initial:n = \c_empty_tl,
136     code-after .value_required:n = true,
137
138
139
140
141 \msg_new:nnnn {nicematrix}
142   {Unknown-option-for-NiceMatrix}
143   {The~option~"\tl_use:N\l_keys_key_tl"~is~unknown~for~the~environment~\\
144    \{NiceMatrix\}~and~its~variants.\\
145    If~you~go~on,~it~will~be~ignored.\\
146    For~a~list~of~the~available~options,~type~H~<return>.}
147   {The~available~options~are~(in~alphabetic~order):~\\
148    code-after,~\\
149    columns-width,~\\
150    create-extra-nodes,~\\
151    extra-left-margin,~\\
152    extra-right-margin,~\\
153    left-margin,~\\
154    name,~\\
155    nullify-dots,~\\
156    parallelize-diags,~\\
157    renew-dots~\\
158    and~right-margin.}
159
160

```

```

161 {The~name~"#1"~is~already~used~and~you~shouldn't~use~
162 the~same~environment~name~twice.~You~can~go~on,~but,~
163 maybe,~you~will~have~incorrect~results~especially~
164 if~you~use~"columns-width=auto".\\\
165 For~a~list~of~the~names~already~used,~type~H~<return>.}
166 {The~names~already~defined~in~this~document~are:~
167 \seq_use:Nnnn~\g_@@_names_seq{~,~}{,~}{~and~}.}

168 \keys_define:nn {NiceMatrix/NiceArray}
169   {parallelize-diags .bool_set:N = \l_@@_parallelize_diags_bool,
170    parallelize-diags .default:n = true,
171    renew-dots .bool_set:N = \l_@@_renew_dots_bool,
172    renew-dots .default:n = true,
173    nullify-dots .bool_set:N = \l_@@_nullify_dots_bool ,
174    nullify-dots .default:n = true,
175    columns-width .code:n = \str_if_eq:nnTF {#1} {auto}
176      {\bool_set_true:N \l_@@_auto_columns_width_bool}
177      {\dim_set:Nn \l_@@_columns_width_dim {#1}},
178    columns-width .value_required:n = true,
179    name .code:n = {\seq_if_in:NnTF \g_@@_names_seq {#1}
180      {\@_error:nn {Duplicate~name} {#1}}
181      {\seq_gput_left:Nn \g_@@_names_seq {#1}}
182      \tl_set:Nn \l_@@_name_tl {#1}},
183    name .value_required:n = true,

```

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

```

184   c .code:n = \tl_set:Nn \l_@@_pos_env_tl c,
185   t .code:n = \tl_set:Nn \l_@@_pos_env_tl t,
186   b .code:n = \tl_set:Nn \l_@@_pos_env_tl b,
187   code-after .tl_set:N = \l_@@_code_after_tl,
188   code-after .initial:n = \c_empty_tl,
189   code-after .value_required:n = true,
190   create-extra-nodes .bool_set:N = \l_@@_extra_nodes_bool,
191   create-extra-nodes .default:n = true,
192   left-margin .dim_set:N = \l_@@_left_margin_dim,
193   left-margin .default:n = \arraycolsep,
194   right-margin .dim_set:N = \l_@@_right_margin_dim,
195   right-margin .default:n = \arraycolsep,
196   extra-left-margin .dim_set:N = \l_@@_extra_left_margin_dim,
197   extra-right-margin .dim_set:N = \l_@@_extra_right_margin_dim,
198   unknown .code:n = \@_error:n {Unknown~option~for~NiceArray}

199 \msg_new:nnnn {nicematrix}
200   {Unknown~option~for~NiceArray}
201   {The~option~"\tl_use:N\l_keys_key_tl"~is~unknown~for~the~environment~
202    \{NiceArray\}.\\\
203    If~you~go~on,~it~will~be~ignored.\\\
204    For~a~list~of~the~available~options,~type~H~<return>.}
205   {The~available~options~are~(in~alphabetic~order):~
206    b,~
207    c,~
208    code-after,~
209    create-extra-nodes,~
210    columns-width,~
211    extra-left-margin,~
212    extra-right-margin,~
213    left-margin,~
214    name,~
215    nullify-dots,~
216    parallelize-diags,~
217    renew-dots,~
218    right-margin,~
219    and-t.}

```

12.4 The environments {NiceArray} 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}`).

```
220 \cs_new_protected:Nn \@@_Cell:
221 {
```

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

```
222     \int_gincr:N \g_@@_column_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.

```
223     \int_compare:nNnT \g_@@_column_int = 1
224         {\int_gincr:N \g_@@_row_int}
225     \int_gset:Nn \g_@@_column_total_int
226         {\int_max:nn \g_@@_column_total_int \g_@@_column_int}
227     \hbox_set:Nw \l_tmpa_box $ %
228     \int_compare:nNnT \g_@@_row_int = 0
229         {\l_@@_code_for_first_row_tl}
230 \cs_new_protected:Nn \@@_end_Cell:
231     {$ %
232     \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 last column of an environment of the kind of `{pNiceArrayC}`).

```
233 \dim_gset:Nn \g_@@_max_cell_width_dim
234     {\dim_max:nn \g_@@_max_cell_width_dim {\box_wd:N \l_tmpa_box}}
235 \int_compare:nNnT \g_@@_row_int = 0
236     {\dim_gset:Nn \g_@@_max_dp_row_zero_dim
237         {\dim_max:nn \g_@@_max_dp_row_zero_dim {\box_dp:N \l_tmpa_box}}
238     \dim_gset:Nn \g_@@_max_ht_row_zero_dim
239         {\dim_max:nn \g_@@_max_ht_row_zero_dim {\box_ht:N \l_tmpa_box}}}
240 \int_compare:nNnT \g_@@_row_int = 1
241     {\dim_gset:Nn \g_@@_max_ht_row_one_dim
242         {\dim_max:nn \g_@@_max_ht_row_one_dim {\box_ht:N \l_tmpa_box}}}
```

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

```
243 \tikz[remember~picture, inner~sep = 0pt, minimum~width = 0pt, baseline]
244     \node [anchor = base,
245             name = nm-\int_use:N \g_@@_env_int-
246                     \int_use:N \g_@@_row_int-
247                     \int_use:N \g_@@_column_int,
248             alias = \tl_if_empty:NF \l_@@_name_tl
249                 {\tl_use:N \l_@@_name_tl-
250                     \int_use:N \g_@@_row_int-
251                     \int_use:N \g_@@_column_int} ]
252     \bgroup
253     \box_use:N \l_tmpa_box
254     \egroup ;}
```

The environment `{NiceArray}` is the main environment of the extension `nicematrix`.

In order to clarify the explanations, we will first give the definition of the environment `{NiceMatrix}`. Our environment `{NiceMatrix}` must have the same second part as the environment `{matrix}` of `amsmath` (because of the programmation 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 must use `\NiceArray` and not `\begin{NiceArray}` (and, in the definition of `{NiceArray}`, we will have to use `\array`, and not `\begin{array}`: see below).

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

```

255 \NewDocumentEnvironment {NiceMatrix} {!O{}}
256   {\keys_set:nn {NiceMatrix/NiceMatrix} {#1}
257     \tl_set:Nn \l_@@_pos_env_tl c
258     \bool_set_false:N \l_@@_exterior_arraycolsep_bool
259       \NiceArray{*\c@MaxMatrixCols{C}}
260   }
261 {\endarray
262   \skip_horizontal:n {-\arraycolsep}
263   \skip_horizontal:n {\g_@@_right_margin_dim + \g_@@_extra_right_margin_dim}}

```

For the definition of `{NiceArray}` (just below), we have the following constraints:

- we must use `\array` in the first part of `{NiceArray}` and, therefore, `\endarray` in the second part;
- we have to put a `\group_insert_after:N \@@_after_array:` in the first part of `{NiceArray}` so that `\@@_draw_lines` will be executed at the end of the current environment (either `{NiceArray}` or `{NiceMatrix}`).

```
264 \cs_generate_variant:Nn \dim_set:Nn {Nx}
```

```

265 \msg_new:nnn {nicematrix}
266   {We~are~yet~in~an~environment~NiceArray}
267   {Environments~\{NiceArray\}~(or~\{NiceMatrix\},~etc.)~can't~be~
268    nested.~We~can~go~on,~but,~maybe,~you~will~have~errors~or~an~incorrect~
269    result.}

```

The command `\@@_define_dots:` will be used in the environment `{NiceArray}` to define the commands `\Ldots`, `\Cdots`, etc.

```

270 \cs_new_protected:Npn \@@_define_dots:
271   {\cs_set_eq:NN \Ldots \@@_Ldots
272    \cs_set_eq:NN \Cdots \@@_Cdots
273    \cs_set_eq:NN \Vdots \@@_Vdots
274    \cs_set_eq:NN \Ddots \@@_Ddots
275    \cs_set_eq:NN \Iddots \@@_Iddots
276    \bool_if:NT \l_@@_renew_dots_bool
277      {\cs_set_eq:NN \ldots \@@_Ldots
278       \cs_set_eq:NN \cdots \@@_Cdots
279       \cs_set_eq:NN \vdots \@@_Vdots
280       \cs_set_eq:NN \ddots \@@_Ddots
281       \cs_set_eq:NN \iddots \@@_Iddots
282       \cs_set_eq:NN \dots \@@_Ldots
283       \cs_set_eq:NN \hdotsfor \@@_Hdotsfor}}

```

With `\@@_define_dots_to_nil:`, the commands like `\Ldots`, `\Cdots`, are defined, but with no effect. This command will be used if the class option `draft` is used.

```

284 \cs_new_protected:Npn \@@_define_dots_to_nil:
285   {\cs_set_eq:NN \Ldots \prg_do_nothing:
286    \cs_set_eq:NN \Cdots \prg_do_nothing:
287    \cs_set_eq:NN \Vdots \prg_do_nothing:
288    \cs_set_eq:NN \Ddots \prg_do_nothing:
289    \cs_set_eq:NN \Iddots \prg_do_nothing:
290    \bool_if:NT \l_@@_renew_dots_bool
291      {\cs_set_eq:NN \ldots \prg_do_nothing:
292       \cs_set_eq:NN \cdots \prg_do_nothing:

```

```

293     \cs_set_eq:NN \vdots \prg_do_nothing:
294     \cs_set_eq:NN \ddots \prg_do_nothing:
295     \cs_set_eq:NN \iddots \prg_do_nothing:
296     \cs_set_eq:NN \dots \prg_do_nothing:
297     \cs_set_eq:NN \hdotsfor \@@_Hdotsfor}}

```

First, we test if we are yet in an environment `{NiceArray}` (nested environments are forbidden). It's easy to test whether we are in an environment `{NiceArray}`: a special command `\@@_in_NiceArray` is defined.

```

298 \NewDocumentEnvironment {NiceArray} {O{} m !O{}}
299   {\cs_if_exist:NT \@@_in_NiceArray:
300     {\@@_error:n {We~are~yet~in~an~environment~NiceArray}}}

```

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

```

301   \cs_if_exist:NT \tikz@library@external@loaded
302     {\tikzset{external/export = false}}
303   \cs_set:Npn \@@_in_NiceArray: {--Void--}
304   \group_insert_after:N \@@_after_array:
305   \tl_gclear_new:N \g_@@_lines_to_draw_tl

```

We increment the counter `\g_@@_env_int` which counts the environments `{NiceArray}`.

```

306   \int_gincr:N \g_@@_env_int
307   \bool_if:NF \l_@@_block_auto_columns_width_bool
308     {\dim_gzero_new:N \g_@@_max_cell_width_dim}

```

For the following variables, maybe we should create it only if we use the environment `{pNiceArrayRC}` or its variants.

```

309   \dim_gzero_new:N \g_@@_max_dp_row_zero_dim
310   \dim_gzero_new:N \g_@@_max_ht_row_zero_dim
311   \dim_gzero_new:N \g_@@_max_ht_row_one_dim
312   \keys_set:nn {NiceMatrix/NiceArray} {#1,#3}

```

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).

```

313   \bool_if:NT \l_@@_auto_columns_width_bool
314     {\group_insert_after:N \@@_write_max_cell_width:
315       \cs_if_free:cTF {_@@_max_cell_width_\int_use:N \g_@@_env_int}
316         {\dim_set:Nn \l_@@_columns_width_dim \c_zero_dim}
317         {\dim_set:Nx \l_@@_columns_width_dim
318           {\use:c {_@@_max_cell_width_\int_use:N \g_@@_env_int}}}}

```

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)).

```

319   \tl_if_empty:NF \l_@@_name_tl
320     {\cs_if_free:cF {_@@_max_cell_width_\l_@@_name_tl}
321       {\dim_set:Nx \l_@@_columns_width_dim
322         {\use:c {_@@_max_cell_width_\l_@@_name_tl}}}}
323   }

```

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.

```

324   \bool_gset_eq:NN \g_@@_extra_nodes_bool \l_@@_extra_nodes_bool
325   \dim_gset_eq:NN \g_@@_left_margin_dim \l_@@_left_margin_dim
326   \dim_gset_eq:NN \g_@@_right_margin_dim \l_@@_right_margin_dim
327   \dim_gset_eq:NN \g_@@_extra_right_margin_dim \l_@@_extra_right_margin_dim
328   \tl_gset_eq:NN \g_@@_code_after_tl \l_@@_code_after_tl
329   \tl_gset_eq:NN \g_@@_name_tl \l_@@_name_tl

```

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

```

330   \cs_set:Npn \ialign
331     {\everycr{\noalign{\int_gzero:N \g_@@_column_int}}}
332     \tabskip = \c_zero_skip
333     \cs_set:Npn \ialign {\everycr{}}
334       \tabskip = \c_zero_skip
335       \halign}
336   \halign}

```

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}`.

```

337   \dim_compare:nNnTF \l_@@_columns_width_dim = \c_zero_dim
338     {\newcolumntype{L}{>{\@{Cell:}l<{\@{end_Cell:}}}}
339     \newcolumntype{C}{>{\@{Cell:c}<{\@{end_Cell:}}}}
340     \newcolumntype{R}{>{\@{Cell:r}<{\@{end_Cell:}}}}}

```

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.

```

341   {\newcolumntype{L}{wl{\dim_use:N \l_@@_columns_width_dim}}}
342   \newcolumntype{C}{wc{\dim_use:N \l_@@_columns_width_dim}}
343   \newcolumntype{R}{wr{\dim_use:N \l_@@_columns_width_dim}}}

```

We nullify the definitions of the column types `w` and `W` because we want to avoid a warning in the log file for a redefinition of a column type.

```

344   \cs_set_eq:NN \NC@find@W \relax
345   \cs_set_eq:NN \NC@find@W \relax

```

We redefine the column types `w` and `W` of the package `array`.

```

346   \newcolumntype{w}[2]
347     {>{\hbox_set:Nw \l_tmpa_box
348       \@@_Cell:}
349       c
350       <{\@@_end_Cell:
351         \hbox_set_end:
352         \makebox[##2][##1]{\box_use:N \l_tmpa_box}}}
353   \newcolumntype{W}[2]
354     {>{\hbox_set:Nw \l_tmpa_box
355       \@@_Cell:}
356       c
357       <{\@@_end_Cell:
358         \hbox_set_end:
359         \cs_set_eq:NN \hss \hfil
360         \makebox[##2][##1]{\box_use:N \l_tmpa_box}}}

```

The commands `\Ldots`, `\Cdots`, etc. will be defined only in the environment `{NiceArray}`. If the class option `draft` is used, these commands will be defined to be no-op (the dotted lines are not drawn).

```

361   \bool_if:NTF \c_@@_draft_bool
362     \@@_define_dots_to_nil:
363     \@@_define_dots:
364     \cs_set_eq:NN \Hspace \@@_Hspace:
365     \cs_set_eq:NN \Hdotsfor \@@_Hdotsfor
366     \cs_set_eq:NN \multicolumn \@@_multicolumn:nnn

```

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

```

367   \seq_gclear_new:N \g_@@_empty_cells_seq

```

²⁰With this programmation, 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}`)

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

```
368   \seq_gclear_new:N \g_@@_multicolumn_cells_seq
369   \seq_gclear_new:N \g_@@_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). At the end of the environment `{array}`, this counter will give the total number of rows of the matrix.

```
370   \int_gzero_new:N \g_@@_row_int
371   \int_gset:Nn \g_@@_row_int {\l_@@_nb_first_row_int - 1}
```

The counter `\g_@@_column_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_@@_column_total_int`. These counters are updated in the command `\@@_Cell`: executed at the beginning of each cell.

```
372   \int_gzero_new:N \g_@@_column_int
373   \int_gzero_new:N \g_@@_column_total_int
374   \cs_set_eq:NN \c@ifnextchar \new@cifnextchar
```

The extra horizontal spaces on both sides of an environment `{array}` should be considered as a bad idea of standard LaTeX. In the environment `{matrix}` the package `amsmath` prefers to suppress these spaces with instructions “`\hskip -\arraycolsep`”. In the same way, we decide to suppress them in `{NiceArray}`. However, for better compatibility, we give an option `exterior-arraycolsep` to control this feature.

```
375   \bool_if:NF \l_@@_exterior_arraycolsep_bool
376     {\skip_horizontal:n {-\arraycolsep}}
377   \skip_horizontal:n {\l_@@_left_margin_dim + \l_@@_extra_left_margin_dim}
```

Eventually, the environment `{NiceArray}` is defined upon the environment `{array}`. The token list `\l_@@_pos_tl` will contain one of the values `t`, `c` or `b`.

```
378   \array[\l_@@_pos_env_tl]{#2}

379   {\endarray
380   \bool_if:NF \l_@@_exterior_arraycolsep_bool
381     {\skip_horizontal:n {-\arraycolsep}}
382   \skip_horizontal:n {\g_@@_right_margin_dim + \g_@@_extra_right_margin_dim}}
```

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

```
383 \NewDocumentEnvironment {pNiceMatrix} {}
384   {\left(\begin{NiceMatrix}\right)}
385   {\end{NiceMatrix}\right) }

386 \NewDocumentEnvironment {bNiceMatrix} {}
387   {\left[\begin{NiceMatrix}\right]}
388   {\end{NiceMatrix}\right] }

389 \NewDocumentEnvironment {BNiceMatrix} {}
390   {\left.\begin{NiceMatrix}\right\}}
391   {\end{NiceMatrix}\right\} }

392 \NewDocumentEnvironment {vNiceMatrix} {}
393   {\left.\begin{NiceMatrix}\right.}
394   {\end{NiceMatrix}\right. }

395 \NewDocumentEnvironment {VNiceMatrix} {}
396   {\left.\begin{NiceMatrix}\right.}
397   {\end{NiceMatrix}\right. }
```

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`.

```

398 \cs_new_protected:Nn \@@_write_max_cell_width:
399   {\bool_if:NF \l_@@_block_auto_columns_width_bool
400     {\iow_now:Nn \mainaux {\ExplSyntaxOn}
401       \iow_now:Nx \mainaux {\cs_gset:cpn
402         {\@@_max_cell_width_\int_use:N \g_@@_env_int}
403         {\dim_use:N \g_@@_max_cell_width_dim} }
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
877
878
879
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1045
1046
1047
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1055
1056
1057
1057
1058
1059
1059
1060
1061
1062
1063
1063
1064
1065
1065
1066
1067
1067
1068
1069
1069
1070
1071
1071
1072
1073
1073
1074
1075
1075
1076
1077
1077
1078
1079
1079
1080
1081
1081
1082
1083
1083
1084
1085
1085
1086
1087
1087
1088
1089
1089
1090
1091
1091
1092
1093
1093
1094
1095
1095
1096
1097
1097
1098
1099
1099
1100
1101
1101
1102
1103
1103
1104
1105
1105
1106
1107
1107
1108
1109
1109
1110
1111
1111
1112
1113
1113
1114
1115
1115
1116
1117
1117
1118
1119
1119
1120
1121
1121
1122
1123
1123
1124
1125
1125
1126
1127
1127
1128
1129
1129
1130
1131
1131
1132
1133
1133
1134
1135
1135
1136
1137
1137
1138
1139
1139
1140
1141
1141
1142
1143
1143
1144
1145
1145
1146
1147
1147
1148
1149
1149
1150
1151
1151
1152
1153
1153
1154
1155
1155
1156
1157
1157
1158
1159
1159
1160
1161
1161
1162
1163
1163
1164
1165
1165
1166
1167
1167
1168
1169
1169
1170
1171
1171
1172
1173
1173
1174
1175
1175
1176
1177
1177
1178
1179
1179
1180
1181
1181
1182
1183
1183
1184
1185
1185
1186
1187
1187
1188
1189
1189
1190
1191
1191
1192
1193
1193
1194
1195
1195
1196
1197
1197
1198
1199
1199
1200
1201
1201
1202
1203
1203
1204
1205
1205
1206
1207
1207
1208
1209
1209
1210
1211
1211
1212
1213
1213
1214
1215
1215
1216
1217
1217
1218
1219
1219
1220
1221
1221
1222
1223
1223
1224
1225
1225
1226
1227
1227
1228
1229
1229
1230
1231
1231
1232
1233
1233
1234
1235
1235
1236
1237
1237
1238
1239
1239
1240
1241
1241
1242
1243
1243
1244
1245
1245
1246
1247
1247
1248
1249
1249
1250
1251
1251
1252
1253
1253
1254
1255
1255
1256
1257
1257
1258
1259
1259
1260
1261
1261
1262
1263
1263
1264
1265
1265
1266
1267
1267
1268
1269
1269
1270
1271
1271
1272
1273
1273
1274
1275
1275
1276
1277
1277
1278
1279
1279
1280
1281
1281
1282
1283
1283
1284
1285
1285
1286
1287
1287
1288
1289
1289
1290
1291
1291
1292
1293
1293
1294
1295
1295
1296
1297
1297
1298
1299
1299
1300
1301
1301
1302
1303
1303
1304
1305
1305
1306
1307
1307
1308
1309
1309
1310
1311
1311
1312
1313
1313
1314
1315
1315
1316
1317
1317
1318
1319
1319
1320
1321
1321
1322
1323
1323
1324
1325
1325
1326
1327
1327
1328
1329
1329
1330
1331
1331
1332
1333
1333
1334
1335
1335
1336
1337
1337
1338
1339
1339
1340
1341
1341
1342
1343
1343
1344
1345
1345
1346
1347
1347
1348
1349
1349
1350
1351
1351
1352
1353
1353
1354
1355
1355
1356
1357
1357
1358
1359
1359
1360
1361
1361
1362
1363
1363
1364
1365
1365
1366
1367
1367
1368
1369
1369
1370
1371
1371
1372
1373
1373
1374
1375
1375
1376
1377
1377
1378
1379
1379
1380
1381
1381
1382
1383
1383
1384
1385
1385
1386
1387
1387
1388
1389
1389
1390
1391
1391
1392
1393
1393
1394
1395
1395
1396
1397
1397
1398
1399
1399
1400
1401
1401
1402
1403
1403
1404
1405
1405
1406
1407
1407
1408
1409
1409
1410
1411
1411
1412
1413
1413
1414
1415
1415
1416
1417
1417
1418
1419
1419
1420
1421
1421
1422
1423
1423
1424
1425
1425
1426
1427
1427
1428
1429
1429
1430
1431
1431
1432
1433
1433
1434
1435
1435
1436
1437
1437
1438
1439
1439
1440
1441
1441
1442
1443
1443
1444
1445
1445
1446
1447
1447
1448
1449
1449
1450
1451
1451
1452
1453
1453
1454
1455
1455
1456
1457
1457
1458
1459
1459
1460
1461
1461
1462
1463
1463
1464
1465
1465
1466
1467
1467
1468
1469
1469
1470
1471
1471
1472
1473
1473
1474
1475
1475
1476
1477
1477
1478
1479
1479
1480
1481
1481
1482
1483
1483
1484
1485
1485
1486
1487
1487
1488
1489
1489
1490
1491
1491
1492
1493
1493
1494
1495
1495
1496
1497
1497
1498
1499
1499
1500
1501
1501
1502
1503
1503
1504
1505
1505
1506
1507
1507
1508
1509
1509
1510
1511
1511
1512
1513
1513
1514
1515
1515
1516
1517
1517
1518
1519
1519
1520
1521
1521
1522
1523
1523
1524
1525
1525
1526
1527
1527
1528
1529
1529
1530
1531
1531
1532
1533
1533
1534
1535
1535
1536
1537
1537
1538
1539
1539
1540
1541
1541
1542
1543
1543
1544
1545
1545
1546
1547
1547
1548
1549
1549
1550
1551
1551
1552
1553
1553
1554
1555
1555
1556
1557
1557
1558
1559
1559
1560
1561
1561
1562
1563
1563
1564
1565
1565
1566
1567
1567
1568
1569
1569
1570
1571
1571
1572
1573
1573
1574
1575
1575
1576
1577
1577
1578
1579
1579
1580
1581
1581
1582
1583
1583
1584
1585
1585
1586
1587
1587
1588
1589
1589
1590
1591
1591
1592
1593
1593
1594
1595
1595
1596
1597
1597
1598
1599
1599
1600
1601
1601
1602
1603
1603
1604
1605
1605
1606
1607
1607
1608
1609
1609
1610
1611
1611
1612
1613
1613
1614
1615
1615
1616
1617
1617
1618
1619
1619
1620
1621
1621
1622
1623
1623
1624
1625
1625
1626
1627
1627
1628
1629
1629
1630
1631
1631
1632
1633
1633
1634
1635
1635
1636
1637
1637
1638
1639
1639
1640
1641
1641
1642
1643
1643
1644
1645
1645
1646
1647
1647
1648
1649
1649
1650
1651
1651
1652
1653
1653
1654
1655
1655
1656
1657
1657
1658
1659
1659
1660
1661
1661
1662
1663
1663
1664
1665
1665
1666
1667
1667
1668
1669
1669
1670
1671
1671
1672
1673
1673
1674
1675
1675
1676
1677
1677
1678
1679
1679
1680
1681
1681
1682
1683
1683
1684
1685
1685
1686
1687
1687
1688
1689
1689
1690
1691
1691
1692
1693
1693
1694
1695
1695
1696
1697
1697
1698
1699
1699
1700
1701
1701
1702
1703
1703
1704
1705
1705
1706
1707
1707
1708
1709
1709
1710
1711
1711
1712
1713
1713
1714
1715
1715
1716
1717
1717
1718
1719
1719
1720
1721
1721
1722
1723
1723
1724
1725
1725
1726
1727
1727
1728
1729
1729
1730
1731
1731
1732
1733
1733
1734
1735
1735
1736
1737
1737
1738
1739
1739
1740
1741
1741
1742
1743
1743
1744
1745
1745
1746
1747
1747
1748
1749
1749
1750
1751
1751
1752
1753
1753
1754
1755
1755
1756
1757
1757
1758
1759
1759
1760
1761
1761
1762
1763
1763
1764
1765
1765
1766
1767
1767
1768
1769
1769
1770
1771
1771
1772
1773
1773
1774
1775
1775
1776
1777
1777
1778
1779
1779
1780
1781
1781
1782
1783
1783
1784
1785
1785
1786
1787
1787
1788
1789
1789
1790
1791
1791
1792
1793
1793
1794
1795
1795
1796
1797
1797
1798
1799
1799
1800
1801
1801
1802
1803
1803
1804
1805
1805
1806
1807
1807
1808
1809
1809
1810
1811
1811
1812
1813
1813
1814
1815
1815
1816
1817
1817
1818
1819
1819
1820
1821
1821
1822
1823
1823
1824
1825
1825
1826
1827
1827
1828
1829
1829
1830
1831
1831
1832
1833
1833
1834
1835
1835
1836
1837
1837
1838
1839
1839
1840
1841
1841
1842
1843
1843
1844
1845
1845
1846
1847
1847
1848
1849
1849
1850
1851
1851
1852
1853
1853
1854
1855
1855
1856
1857
1857
1858
1859
1859
1860
1861
1861
1862
1863
1863
1864
1865
1865
1866
1867
1867
1868
1869
1869
1870
1871
1871
1872
1873
1873
1874
1875
1875
1876
1877
1877
1878
1879
1879
1880
1881
1881
1882
1883
1883
1884
1885
1885
1886
1887
1887
1888
1889
1889
1890
1891
1891
1892
1893
1893
1894
1895
1895
1896
1897
1897
1898
1899
1899
1900
1901
1901
1902
1903
1903
1904
1905
1905
1906
1907
1907
1908
1909
1909
1910
1911
1911
1912
1913
1913
1914
1915
1915
1916
1917
1917
1918
1919
1919
1920
1921
1921
1922
1923
1923
1924
1925
1925
1926
1927
1927
1928
1929
1929
1930
1931
1931
1932
1933
1933
1934
1935
1935
1936
1937
1937
1938
1939
1939
1940
1941
1941
1942
1943
1943
1944
1945
1945
1946
1947
1947
1948
1949
1949
1950
1951
1951
1952
1953
1953
1954
1955
1955
1956
1957
1957
1958
1959
1959
1960
1961
1961
1962
1963
1963
1964
1965
1965
1966
1967
1967
1968
1969
1969
1970
1971
1971
1972
1973
1973
1974
1975
1975
1976
1977
1977
1978
1979
1979
1980
1981
1981
1982
1983
1983
1984
1985
1985
1986
1987
1987
1988
1989
1989
1990
1991
1991
1992
1993
1993
1994
1995
1995
1996
1997
1997
1998
1999
1999
2000
2001
2001
2002
2003
2003
2004
2005
2005
2006
2007
2007
2008
2009
2009
2010
2011
2011
2012
2013
2013
2014
2015
2015
2016
2017
2017
2018
2019
2019
2020
2021
2021
2022
2023
2023
2024
2025
2025
2026
2027
2027
2028
2029
2029
2030
2031
2031
2032
2033
2033
2034
2035
2035
2036
2037
2037
2038
2039
2039
2040
2041
2041
2042
2043
2043
2044
2045
2045
2046
2047
2047
2048
2049
2049
2050
2051
2051
2052
2053
2053
2054
2055
2055
2056
2057
2057
2058
2059
2059
2060
2061
2061
2062
2063
2063
2064
2065
2065
2066
2067
2067
2068
2069
2069
2070
2071
2071
2072
2073
2073
2074
2075
2075
2076
2077
2077
2078
2079
2079
2080
2081
2081
2082
2083
2083
2084
2085
2085
2086
2087
2087
2088
2089
2089
2090
2091
2091
2092
2093
2093
2094
2095
2095
2096
2097
2097
2098
2099
2099
2100
2101
2101
2102
2103
2103
2104
2105
2105
2106
2107
2107
2108
2109
2109
2110
2111
2111
211
```

For example, for the following matrix,

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

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

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

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

429 \cs_new_protected:Nn \@@_instruction_of_type:n
430   {\tl_gput_right:Nx \g_@@_lines_to_draw_tl
431     {\exp_not:c {\@@_draw_#1:nn}
432       {\int_use:N \g_@@_row_int}
433       {\int_use:N \g_@@_column_int}}}}
```

12.5 After the construction of the array

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

```
434 \cs_new_protected:Nn \@@_after_array:
435   {\group_begin:
436     \cs_if_exist:NT \tikz@library@external@loaded
437       {\tikzset{external/export = false}}}
```

Now, the definition of the counters `\g_@@_column_int` and `\g_@@_column_total_int` change: `\g_@@_column_int` will be the number of columns without the exterior column (in an environment like `{pNiceArrayC}`) and `\g_@@_column_total_int` will be the number of columns with this exterior column.

```
438   \int_gset_eq:NN \g_@@_column_int \g_@@_column_total_int
439   \bool_if:NT \l_@@_exterior_column_bool {\int_gdecr:N \g_@@_column_int}
```

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

```
440   \seq_gclear_new:N \g_@@_yet_drawn_seq
```

By default, the diagonal lines will be parallelized²¹. There are two types of diagonals lines: the `\Ddots` diagonals and the `\Idots` 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.

```
441   \bool_if:NT \l_@@_parallelize_diags_bool
442     {\int_zero_new:N \l_@@_ddots_int
443      \int_zero_new:N \l_@@_iddots_int}
```

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

```
444   \dim_zero_new:N \l_@@_delta_x_one_dim
445   \dim_zero_new:N \l_@@_delta_y_one_dim
446   \dim_zero_new:N \l_@@_delta_x_two_dim
447   \dim_zero_new:N \l_@@_delta_y_two_dim}
```

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).

```
448   \bool_if:NT \g_@@_extra_nodes_bool \@@_create_extra_nodes:
```

²¹It's possible to use the option `parallelize-diags` to disable this parallelization.

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

```

449   \tl_if_empty:NF \g_@@_lines_to_draw_tl
450     {\int_zero_new:N \l_@@_initial_i_int
451      \int_zero_new:N \l_@@_initial_j_int
452      \int_zero_new:N \l_@@_final_i_int
453      \int_zero_new:N \l_@@_final_j_int
454      \bool_new:N \l_@@_initial_open_bool
455      \bool_new:N \l_@@_final_open_bool
456      \g_@@_lines_to_draw_tl}
457   \tl_gclear:N \g_@@_lines_to_draw_tl

```

Now, the code-after.

```

458 \tikzset{every picture/.style = {overlay,
459           remember picture,
460           name-prefix = nm-\int_use:N \g_@@_env_int-}}
461 \cs_set_eq:NN \line \@@_line:nn
462 \g_@@_code_after_tl
463 \group_end:}

```

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” (the medium and large nodes are created with `\@@_create_extra_nodes`: if they have not been created yet).

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

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 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 wether the extremities are open or not.

```

464 \cs_new_protected:Nn \@@_find_extremities_of_line:nnnn
465   { \int_set:Nn \l_@@_initial_i_int {#1}
466     \int_set:Nn \l_@@_initial_j_int {#2}
467     \int_set:Nn \l_@@_final_i_int {#1}
468     \int_set:Nn \l_@@_final_j_int {#2}
469     \bool_set_false:N \l_@@_initial_open_bool
470     \bool_set_false:N \l_@@_final_open_bool

```

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

```

471      \@@_bool_new:N \l_@@_stop_loop_bool
472      \bool_do_until:Nn \l_@@_stop_loop_bool
473          {\int_add:Nn \l_@@_final_i_int {#3}
474           \int_add:Nn \l_@@_final_j_int {#4}}

```

We test if we are still in the matrix.

```

475          \bool_if:nTF { \int_compare_p:nNn
476              \l_@@_final_i_int < {\l_@@_nb_first_row_int - 1}
477              || \int_compare_p:nNn
478                  \l_@@_final_i_int > \g_@@_row_int
479              || \int_compare_p:nNn
480                  \l_@@_final_j_int < 1
481              || \int_compare_p:nNn
482                  \l_@@_final_j_int > \g_@@_column_total_int

```

If you arrive in the column C of an environment with such columns (like `{pNiceArrayC}`), you must consider that we are *outside* the matrix except if we are drawing a vertical line (included in the column C).

```

483          || \int_compare_p:nNn
484              \l_@@_final_j_int > \g_@@_column_int
485              && \int_compare_p:nNn {#4} > 0 }

```

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

```
486      \bool_set_true:N \l_@@_final_open_bool
```

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).

```

487          \int_sub:Nn \l_@@_final_i_int {#3}
488          \int_sub:Nn \l_@@_final_j_int {#4}
489          \bool_set_true:N \l_@@_stop_loop_bool}

```

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

```

490          \@@_if_not_empty_cell:nnT
491              \l_@@_final_i_int
492              \l_@@_final_j_int
493              {\bool_set_true:N \l_@@_stop_loop_bool}
494      }

```

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

```

495          \bool_set_false:N \l_@@_stop_loop_bool
496          \bool_do_until:Nn \l_@@_stop_loop_bool
497              {\int_sub:Nn \l_@@_initial_i_int {#3}
498               \int_sub:Nn \l_@@_initial_j_int {#4}
499               \bool_if:nTF
500                   { \int_compare_p:nNn
501                       \l_@@_initial_i_int < \l_@@_nb_first_row_int
502                       || \int_compare_p:nNn
503                           \l_@@_initial_i_int > \g_@@_row_int
504                       || \int_compare_p:nNn
505                           \l_@@_initial_j_int < 1
506                       || \int_compare_p:nNn
507                           \l_@@_initial_j_int > \g_@@_column_total_int}
508               {\bool_set_true:N \l_@@_initial_open_bool
509                \int_add:Nn \l_@@_initial_i_int {#3}
510                \int_add:Nn \l_@@_initial_j_int {#4}
511                \bool_set_true:N \l_@@_stop_loop_bool}
512               \@@_if_not_empty_cell:nnT
513                   \l_@@_initial_i_int
514                   \l_@@_initial_j_int
515                   {\bool_set_true:N \l_@@_stop_loop_bool}}
516      }

```

If we have at least one open extremity, we create the “medium nodes” in the matrix (in the case of an open extremity, the dotted line uses the “medium node” of the last empty cell). We remind that, when used once, the command `\@_create_extra_nodes:` becomes no-op in the current TeX group.

```
517     \bool_if:nT {\l_@@_initial_open_bool || \l_@@_final_open_bool}
518         \@_create_extra_nodes: }
```

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

```
519 \prg_set_conditional:Npn \@_if_yet_drawn: {F}
520     {\tl_set:Nx \l_tmpa_tl {\int_use:N \l_@@_initial_i_int-
521         \int_use:N \l_@@_initial_j_int-
522             \int_use:N \l_@@_final_i_int-
523                 \int_use:N \l_@@_final_j_int}
524     \seq_if_in:NNTF \g_@@_yet_drawn_seq \l_tmpa_tl }
```

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

```
525     {\prg_return_true:}
526     {\seq_gput_left:NV \g_@@_yet_drawn_seq \l_tmpa_tl
527     \prg_return_false:}}
```

The command `\@_retrieve_coords:nn` retrieves the Tikz coordinates of the two extremities of the dotted line we will have to draw ²². 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 prefix 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}`.

```
528 \cs_new_protected:Nn \@_retrieve_coords:nn
529     {\dim_gzero_new:N \g_@@_x_initial_dim
530     \dim_gzero_new:N \g_@@_y_initial_dim
531     \dim_gzero_new:N \g_@@_x_final_dim
532     \dim_gzero_new:N \g_@@_y_final_dim
533     \begin{tikzpicture}[remember picture]
534     \tikz@parse@node\pgfutil@firstofone
535         (\mm-\int_use:N \g_@@_env_int-
536             \int_use:N \l_@@_initial_i_int-
537                 \int_use:N \l_@@_initial_j_int #1)
538     \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
539     \dim_gset:Nn \g_@@_y_initial_dim \pgf@y
540     \tikz@parse@node\pgfutil@firstofone
541         (\mm-\int_use:N \g_@@_env_int-
542             \int_use:N \l_@@_final_i_int-
543                 \int_use:N \l_@@_final_j_int #2)
544     \dim_gset:Nn \g_@@_x_final_dim \pgf@x
545     \dim_gset:Nn \g_@@_y_final_dim \pgf@y
546     \end{tikzpicture}}
547 \cs_generate_variant:Nn \@_retrieve_coords:nn {xx}

548 \cs_new_protected:Nn \@_draw_Ldots:nn
549     {\@_find_extremities_of_line:nnnn {#1} {#2} 0 1
550     \@_if_yet_drawn:F \@_actually_draw_Ldots:}
```

²²In fact, with diagonal lines, or vertical lines in columns of type L or R, an adjustment of one of the coordinates may be done.

The command `\@@_actually_draw_Ldots`: actually 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`.

```

551 \cs_new_protected:Nn \@@_actually_draw_Ldots:
552     {\@@_retrieve_coords:xx {\bool_if:NTF \l_@@_initial_open_bool
553         {-medium.base~west}
554         {.base~east}}
555         {\bool_if:NTF \l_@@_final_open_bool
556             {-medium.base~east}
557             {.base~west}}
558     \bool_if:NT \l_@@_initial_open_bool
559         {\dim_gset_eq:NN \g_@@_y_initial_dim \g_@@_y_final_dim }
560     \bool_if:NT \l_@@_final_open_bool
561         {\dim_gset_eq:NN \g_@@_y_final_dim \g_@@_y_initial_dim }

```

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

```

562     \dim_gadd:Nn \g_@@_y_initial_dim {0.53pt}
563     \dim_gadd:Nn \g_@@_y_final_dim {0.53pt}
564     \@@_draw_tikz_line:}

565 \cs_new_protected:Nn \@@_draw_Cdots:nn
566     {\@@_find_extremities_of_line:nnnn {#1} {#2} 0 1
567     \@@_if_yet_drawn:F
568         {\@@_retrieve_coords:xx {\bool_if:NTF \l_@@_initial_open_bool
569             {-medium.mid~west}
570             {.mid~east}}
571             {\bool_if:NTF \l_@@_final_open_bool
572                 {-medium.mid~east}
573                 {.mid~west}}
574     \bool_if:NT \l_@@_initial_open_bool
575         {\dim_gset_eq:NN \g_@@_y_initial_dim \g_@@_y_final_dim }
576     \bool_if:NT \l_@@_final_open_bool
577         {\dim_gset_eq:NN \g_@@_y_final_dim \g_@@_y_initial_dim }
578     \@@_draw_tikz_line:}}

```

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?).

```

579 \cs_new_protected:Nn \@@_draw_Vdots:nn
580     {\@@_find_extremities_of_line:nnnn {#1} {#2} 1 0
581     \@@_if_yet_drawn:F
582         {\@@_retrieve_coords:xx {\bool_if:NTF \l_@@_initial_open_bool
583             {-medium.north~west}
584             {.south~west}}
585             {\bool_if:NTF \l_@@_final_open_bool
586                 {-medium.south~west}
587                 {.north~west}}}

```

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

```

588 \bool_set:Nn \l_tmpa_bool
589     {\dim_compare_p:nNn \g_@@_x_initial_dim = \g_@@_x_final_dim}
590     \@@_retrieve_coords:xx {\bool_if:NTF \l_@@_initial_open_bool
591         {-medium.north}
592         {.south}}
593         {\bool_if:NTF \l_@@_final_open_bool
594             {-medium.south}
595             {.north}}}

```

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

```

596     \bool_set:Nn \l_tmpb_bool
597         {\dim_compare_p:nNn \g_@@_x_initial_dim = \g_@@_x_final_dim}
598     \bool_if:NTF \l_tmpb_bool
599         {\dim_gset:Nn \g_@@_x_initial_dim
600             {\bool_if:NTF \l_tmpa_bool \dim_min:nn \dim_max:nn
601                 \g_@@_x_initial_dim \g_@@_x_final_dim}
602             \dim_gset_eq:NN \g_@@_x_final_dim \g_@@_x_initial_dim}
603     \@@_draw_tikz_line:}

```

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.

```

604 \cs_new_protected:Nn \@@_draw_Ddots:nn
605     {\@@_find_extremities_of_line:nnnn {#1} {#2} 1 1
606     \@@_if_yet_drawn:F
607     {\@@_retrieve_coords:xx {\bool_if:NTF \l_@@_initial_open_bool
608         {-medium.north~west}
609         {.south~east}}
610         {\bool_if:NTF \l_@@_final_open_bool
611             {-medium.south~east}
612             {.north~west}}}

```

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.

```

613     \bool_if:NT \l_@@_parallelize_diags_bool
614     {\int_incr:N \l_@@_ddots_int

```

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

```

615     \int_compare:nNnTF \l_@@_ddots_int = 1

```

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

```

616     {\dim_set:Nn \l_@@_delta_x_one_dim {\g_@@_x_final_dim - \g_@@_x_initial_dim }
617     \dim_set:Nn \l_@@_delta_y_one_dim {\g_@@_y_final_dim - \g_@@_y_initial_dim }}

```

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`.

```

618     {\dim_gset:Nn \g_@@_y_final_dim
619         {\g_@@_y_initial_dim +
620             (\g_@@_x_final_dim - \g_@@_x_initial_dim)
621             * \dim_ratio:nn \l_@@_delta_y_one_dim \l_@@_delta_x_one_dim }}}

```

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

```

622     \@@_draw_tikz_line:}

```

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

```

623 \cs_new_protected:Nn \@@_draw_Iddots:nn
624     {\@@_find_extremities_of_line:nnnn {#1} {#2} 1 {-1}
625     \@@_if_yet_drawn:F
626     {\@@_retrieve_coords:xx {\bool_if:NTF \l_@@_initial_open_bool
627         {-medium.north~east}
628         {.south~west}}
629         {\bool_if:NTF \l_@@_final_open_bool
630             {-medium.south~west}
631             {.north~east}}}
632     \bool_if:NT \l_@@_parallelize_diags_bool
633     {\int_incr:N \l_@@_iddots_int
634     \int_compare:nNnTF \l_@@_iddots_int = 1
635         {\dim_set:Nn \l_@@_delta_x_two_dim {\g_@@_x_final_dim - \g_@@_x_initial_dim}
636         \dim_set:Nn \l_@@_delta_y_two_dim {\g_@@_y_final_dim - \g_@@_y_initial_dim}}

```

```

637      {\dim_gset:Nn \g_@@_y_final_dim
638          {\g_@@_y_initial_dim +
639              (\g_@@_x_final_dim - \g_@@_x_initial_dim)
640                  * \dim_ratio:nn \l_@@_delta_y_two_dim \l_@@_delta_x_two_dim }}}
641  \@@_draw_tikz_line:}

```

12.6 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`.

```

642 \cs_new_protected:Nn \@@_draw_tikz_line:
643     {

```

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.

```

644             \dim_zero_new:N \l_@@_l_dim
645             \dim_set:Nn \l_@@_l_dim
646                 { \fp_to_dim:n
647                     { sqrt( ( \dim_use:N \g_@@_x_final_dim
648                         - \dim_use:N \g_@@_x_initial_dim) ^2
649                         + ( \dim_use:N \g_@@_y_final_dim
650                             - \dim_use:N \g_@@_y_initial_dim) ^2 ) }
651                 }

```

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

```

652         \dim_compare:nNnF \l_@@_l_dim = \c_zero_dim

```

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

```

653             {\bool_if:NTF \l_@@_initial_open_bool
654                 {\bool_if:NTF \l_@@_final_open_bool
655                     {\int_set:Nn \l_tmpa_int
656                         { \dim_ratio:nn {\l_@@_l_dim} {0.45em} } }
657                     {\int_set:Nn \l_tmpa_int
658                         { \dim_ratio:nn {\l_@@_l_dim - 0.3em} {0.45em} } }
659                 {\bool_if:NTF \l_@@_final_open_bool
660                     {\int_set:Nn \l_tmpa_int
661                         { \dim_ratio:nn {\l_@@_l_dim - 0.3em} {0.45em} } }
662                     {\int_set:Nn \l_tmpa_int
663                         { \dim_ratio:nn {\l_@@_l_dim - 0.6em} {0.45em} } } }

```

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

```

664             \dim_set:Nn \l_tmpa_dim {(\g_@@_x_final_dim - \g_@@_x_initial_dim)
665                         * \dim_ratio:nn {0.45em} \l_@@_l_dim}
666             \dim_set:Nn \l_tmpb_dim {(\g_@@_y_final_dim - \g_@@_y_initial_dim)
667                         * \dim_ratio:nn {0.45em} \l_@@_l_dim}

```

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`.

```

668             \int_set:Nn \l_tmpb_int
669                 {\bool_if:NTF \l_@@_initial_open_bool
670                     {\bool_if:NTF \l_@@_final_open_bool 1 0}
671                     {\bool_if:NTF \l_@@_final_open_bool 2 1} }

```

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.

```

672             \dim_gadd:Nn \g_@@_x_initial_dim

```

```

673   { (\g_@@_x_final_dim - \g_@@_x_initial_dim)
674     * \dim_ratio:nn {\l_@@_l_dim - 0.45 em * \l_tmpa_int}
675     {\l_@@_l_dim * 2}
676     * \l_tmpb_int}

```

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

```

677   \dim_gadd:Nn \g_@@_y_initial_dim
678   { (\g_@@_y_final_dim - \g_@@_y_initial_dim)
679     * \dim_ratio:nn {\l_@@_l_dim - 0.45 em * \l_tmpa_int}
680     {\l_@@_l_dim * 2}
681     * \l_tmpb_int}
682   \begin{tikzpicture}[overlay]
683   \int_step_inline:nnnn 0 1 \l_tmpa_int
684   { \pgfpathcircle{\pgfpoint{\g_@@_x_initial_dim}
685     {\g_@@_y_initial_dim}}}
686   {0.53pt}
687   \pgfusepath{fill}
688   \dim_gadd:Nn \g_@@_x_initial_dim \l_tmpa_dim
689   \dim_gadd:Nn \g_@@_y_initial_dim \l_tmpb_dim }
690   \end{tikzpicture}}
691 }

```

12.7 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).

```

692 \cs_set_eq:NN \@@_ldots \ldots
693 \cs_set_eq:NN \@@_cdots \cdots
694 \cs_set_eq:NN \@@_vdots \vdots
695 \cs_set_eq:NN \@@_ddots \ddots
696 \cs_set_eq:NN \@@_iddots \iddots

```

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

```

697 \cs_new_protected:Nn \@@_add_to_empty_cells:
698   {\seq_gput_right:Nx \g_@@_empty_cells_seq
699     {\int_use:N \g_@@_row_int-
700      \int_use:N \g_@@_column_int}}

```

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}`).

```

701 \NewDocumentCommand \@@_Ldots {s}
702   {\bool_if:nF {#1} {\@@_instruction_of_type:n {Ldots}}
703    \bool_if:NF \l_@@_nullify_dots_bool {\phantom \@@_ldots}
704    \@@_add_to_empty_cells:}

705 \NewDocumentCommand \@@_Cdots {s}
706   {\bool_if:nF {#1} {\@@_instruction_of_type:n {Cdots}}
707    \bool_if:NF \l_@@_nullify_dots_bool {\phantom \@@_cdots}
708    \@@_add_to_empty_cells:}

709 \NewDocumentCommand \@@_Vdots {s}
710   {\bool_if:nF {#1} {\@@_instruction_of_type:n {Vdots}}
711    \bool_if:NF \l_@@_nullify_dots_bool {\phantom \@@_vdots}
712    \@@_add_to_empty_cells:}

```

```

713 \NewDocumentCommand \@@_Ddots {s}
714   {\bool_if:nF {#1} {\@@_instruction_of_type:n {Ddots}}
715    \bool_if:NF \l_@@_nullify_dots_bool {\phantom \@@_ddots}
716    \@@_add_to_empty_cells:}

717 \NewDocumentCommand \@@_Iddots {s}
718   {\bool_if:nF {#1} {\@@_instruction_of_type:n {Iddots}}
719    \bool_if:NF \l_@@_nullify_dots_bool {\phantom \@@_iddots}
720    \@@_add_to_empty_cells:}

```

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

```

721 \cs_new_protected:Nn \@@_Hspace:
722   {\@@_add_to_empty_cells:
723    \hspace}

```

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

```

724 \cs_set_eq:NN \@@_old_multicolumn \multicolumn
725 \cs_new:Nn \@@_multicolumn:nnn
726   {\@@_old_multicolumn{#1}{#2}{#3}
727    \int_compare:nNnT #1 > 1
728      {\seq_gput_left:Nx \g_@@_multicolumn_cells_seq
729       {\int_eval:n \g_@@_row_int -
730        \int_use:N \g_@@_column_int}
731      \seq_gput_left:Nn \g_@@_multicolumn_sizes_seq {#1}}
732      \int_gadd:Nn \g_@@_column_int {#1-1}}

```

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`.

```

733 \NewDocumentCommand {\@@_Hdotsfor} {O{} m}
734   {\tl_gput_right:Nx \g_@@_lines_to_draw_tl
735    {\exp_not:N \@@_draw_Hdotsfor:nnn
736     {\int_use:N \g_@@_row_int}
737     {\int_use:N \g_@@_column_int}
738     {#2}}
739    \prg_replicate:nn {#2-1} {&}

740 \cs_new_protected:Nn \@@_draw_Hdotsfor:nnn
741   {\bool_set_false:N \l_@@_initial_open_bool
742    \bool_set_false:N \l_@@_final_open_bool}

```

For the row, it's easy.

```

743 \int_set:Nn \l_@@_initial_i_int {#1}
744 \int_set:Nn \l_@@_final_i_int {#1}

```

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

```

745 \int_compare:nNnTF #2 = 1
746   {\int_set:Nn \l_@@_initial_j_int 1
747    \bool_set_true:N \l_@@_initial_open_bool}
748   {\int_set:Nn \l_tmpa_int {#2-1}
749    \@@_if_not_empty_cell:nnTF \l_@@_initial_i_int \l_tmpa_int
750      {\int_set:Nn \l_@@_initial_j_int {#2-1}}
751      {\int_set:Nn \l_@@_initial_j_int {#2}
752       \bool_set_true:N \l_@@_initial_open_bool}}
753   \int_compare:nNnTF {#2+#3-1} = \g_@@_column_int
754     {\int_set:Nn \l_@@_final_j_int {#2+#3-1}
755      \bool_set_true:N \l_@@_final_open_bool}
756     {\int_set:Nn \l_tmpa_int {#2+#3}
757      \@@_if_not_empty_cell:nnTF \l_@@_final_i_int \l_tmpa_int

```

```

758     {\int_set:Nn \l_@@_final_j_int {#2+#3}}
759     {\int_set:Nn \l_@@_final_j_int {#2+#3-1}
760      \bool_set_true:N \l_@@_final_open_bool}}
761 \bool_if:nT {\l_@@_initial_open_bool || \l_@@_final_open_bool}
762     \@@_create_extra_nodes:
763     \@@_actually_draw_Ldots:

```

12.8 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.

```

764 \cs_new_protected:Nn \@@_line:nn
765   {\dim_zero_new:N \g_@@_x_initial_dim
766    \dim_zero_new:N \g_@@_y_initial_dim
767    \dim_zero_new:N \g_@@_x_final_dim
768    \dim_zero_new:N \g_@@_y_final_dim
769    \bool_new:N \l_@@_initial_open_bool
770    \bool_new:N \l_@@_final_open_bool
771    \begin{tikzpicture}
772      \path~(#1)~~~(#2)~node[at~start]~(i)~{}~node[at~end]~(f)~{}~;
773      \tikz@parse@node\pgfutil@firstofone (i)
774      \dim_gset:Nn \g_@@_x_initial_dim \pgf@x
775      \dim_gset:Nn \g_@@_y_initial_dim \pgf@y
776      \tikz@parse@node\pgfutil@firstofone (f)
777      \dim_gset:Nn \g_@@_x_final_dim \pgf@x
778      \dim_gset:Nn \g_@@_y_final_dim \pgf@y
779    \end{tikzpicture}
780    \@@_draw_tikz_line:}

```

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

12.9 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.

```
781 \bool_new:N \l_@@_block_auto_columns_width_bool
```

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

```

782 \keys_define:nn {NiceMatrix/NiceMatrixBlock}
783   {auto-columns-width .code:n =
784    {\bool_set_true:N \l_@@_block_auto_columns_width_bool
785     \dim_gzero_new:N \g_@@_max_cell_width_dim
786     \bool_set_true:N \l_@@_auto_columns_width_bool}}
787 \NewDocumentEnvironment {NiceMatrixBlock} {!O{}}
788   {\keys_set:nn {NiceMatrix/NiceMatrixBlock} {#1}
789    \int_zero_new:N \l_@@_first_env_block_int
790    \int_set:Nn \l_@@_first_env_block_int {\g_@@_env_int + 1}}

```

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`).

```

791 \bool_if:NT \l_@@_block_auto_columns_width_bool
792   {\iow_now:Nn \mainaux \ExplSyntaxOn
793    \int_step_inline:nnnn \l_@@_first_env_block_int 1 \g_@@_env_int
794    {\iow_now:Nx \mainaux
795     {\cs_gset:cpn {\g_@@_max_cell_width_##1}
796      {\dim_use:N \g_@@_max_cell_width_dim}}}
797   \iow_now:Nn \mainaux \ExplSyntaxOff}

```

12.10 The environment {pNiceArrayC} and its variants

The code in this section can be removed without affecting the previous code.

First, we define a set of options for the environment {pNiceArrayC} and its variants. This set of keys is named NiceMatrix/NiceArrayC even though there is no environment called {NiceArrayC}.

```

798 \keys_define:nn {NiceMatrix/NiceArrayC}
799   {parallelize-diags .bool_set:N      = \l_@@_parallelize_diags_bool,
800    parallelize-diags .default:n     = true,
801    renew-dots       .bool_set:N      = \l_@@_renew_dots_bool,
802    renew-dots       .default:n     = true,
803    nullify-dots     .bool_set:N      = \l_@@_nullify_dots_bool ,
804    nullify-dots     .default:n     = true,
805    code-for-last-col .tl_set:N      = \l_@@_code_for_last_col_tl,
806    code-for-last-col .value_required:n = true,
807    columns-width     .code:n        = \str_if_eq:nnTF {#1} {auto}
808                                {\bool_set_true:N
809                                 \l_@@_auto_columns_width_bool}
810                                {\dim_set:Nn \l_@@_columns_width_dim {#1}},
811    columns-width     .value_required:n = true,
812    name              .code:n        = {\seq_if_in:NnTF \g_@@_names_seq {#1}
813                                         {\@_error:nn {Duplicate~name} {#1}}
814                                         {\seq_gput_left:Nn \g_@@_names_seq {#1}}
815                                         \tl_set:Nn \l_@@_name_tl {#1}},
816    name              .value_required:n = true,
817    code-after         .tl_set:N      = \l_@@_code_after_tl,
818    code-after         .initial:n    = \c_empty_tl,
819    code-after         .value_required:n = true,
820    create-extra-nodes .bool_set:N    = \l_@@_extra_nodes_bool,
821    create-extra-nodes .default:n    = true,
822    left-margin        .dim_set:N    = \l_@@_left_margin_dim,
823    left-margin        .default:n    = \arraycolsep,
824    right-margin       .dim_set:N    = \l_@@_right_margin_dim,
825    right-margin       .default:n    = \arraycolsep,
826    extra-left-margin .dim_set:N    = \l_@@_extra_left_margin_dim,
827    extra-right-margin .dim_set:N    = \l_@@_extra_right_margin_dim,
828    unknown            .code:n        = \@_error:n {Unknown~option~for~NiceArrayC}

829 \msg_new:nnnn {nicematrix}
830   {Unknown~option~for~NiceArrayC}
831   {The~option~"\tl_use:N\l_keys_key_tl"~is~unknown~for~the~environment~
832    {\currenvir}.\\
833    If~you~go~on,~it~will~be~ignored.\\
834    For~a~list~of~the~available~options,~type~H~<return>.}
835   {The~available~options~are~(in~alphabetic~order):~
836    code-after,~
837    code-for-last-col,~
838    columns-width,~
839    create-extra-nodes,~
840    extra-left-margin,~
841    extra-right-margin,~
842    left-margin,~
843    name,~
844    nullify-dots,~
845    parallelize-diags~
846    renew-dots~
847    and-right-margin.}

```

In the environment {pNiceArrayC} (and its variants), the last column is composed with instructions `\hbox_overlap_right:n` (this instruction may be seen as the `expl3` equivalent of the classical command `\rlap`). After the composition of the array, an horizontal skip is inserted to compensate for these overlapping boxes.

The command `\@@_NiceArrayC:n` will be used in `{NiceArrayCwithDelims}` but also in the environment `{NiceArrayRCwithDelims}`.

```
848 \cs_new_protected:Nn \@@_NiceArrayC:n
849   {\bool_set_true:N \l_@@_exterior_column_bool
850     \begin{NiceArray}
```

The beginning of the preamble is the argument of the environment `{pNiceArrayC}`.

```
851   {#1
```

However, we add a last column with its own specification. For a cell in this last column, the first operation is to store the content of the cell in the box `\l_tmpa_box`. This is allowed in `expl3` with the construction `\hbox_set:Nw \l_tmpa_box ... \hbox_set_end:`.

```
852   >{\int_gincr:N \g_@@_column_int
853     \int_gset:Nn \g_@@_column_total_int
854       {\int_max:nn \g_@@_column_total_int \g_@@_column_int}
855       \hbox_set:Nw \l_tmpa_box $ % $
856         \l_@@_code_for_last_col_tl
857       }
858 }
```

We actualize the value of `\g_@@_width_last_col_dim` which, at the end of the array, will contain the maximal width of the cells of the last column (thus, it will be equal to the width of the last column).

```
859   <{ $ % $
860     \hbox_set_end:
861     \dim_gset:Nn \g_@@_width_last_col_dim
862       {\dim_max:nn \g_@@_width_last_col_dim
863         {\box_wd:N \l_tmpa_box}}
864       \skip_horizontal:n {-2\arraycolsep}
```

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

```
865   \hbox_overlap_right:n
866     {\skip_horizontal:n
867       { 2\arraycolsep + \l_@@_right_margin_dim
868         + \l_@@_extra_right_margin_dim}
869       \tikz[remember picture, inner-sep=0pt, minimum-width=0pt, baseline]
870         \node [anchor=base,
871           name = nm-\int_use:N \g_@@_env_int-
872             \int_use:N \g_@@_row_int-
873               \int_use:N \g_@@_column_int,
874             alias = \tl_if_empty:NF \l_@@_name_tl
875               {\l_@@_name_tl-
876                 \int_use:N \g_@@_row_int-
877                   \int_use:N \g_@@_column_int}]
878             {\box_use:N \l_tmpa_box} ; } }}
```

This ends the preamble of the array that will be constructed (a rather long preamble, indeed).

The environments of the type of `{pNiceArrayC}` will be constructed over `{NiceArrayCwithDelims}`. The first two arguments of this environment are the left and the right delimiter.

```
879 \NewDocumentEnvironment{NiceArrayCwithDelims} {mm O{} m !O{}}
880   {\dim_gzero_new:N \g_@@_width_last_col_dim
881     \keys_set:nn {NiceMatrix/NiceArrayC} {#3,#5}
882     \bool_set_false:N \l_@@_exterior_arraycolsep_bool
883     \tl_set:Nn \l_@@_pos_env_tl c
884     \left#1
885     \@@_NiceArrayC:n {#4}
886   \end{NiceArray}
887   \right#2
888   \skip_horizontal:n \g_@@_width_last_col_dim
889 }
```

In the following environments, we don't use the form with `\begin{...}` and `\end{...}` because we use `\@currenvir` in the error message for an unknown option.

```
890 \NewDocumentEnvironment {pNiceArrayC} {}
```

```

891 {\NiceArrayCwithDelims{{}{}}}
892 {\endNiceArrayCwithDelims}
893 \NewDocumentEnvironment {vNiceArrayC} {}
894   {\NiceArrayCwithDelims{|}{|}}
895   {\endNiceArrayCwithDelims}
896 \NewDocumentEnvironment {VNiceArrayC} {}
897   {\NiceArrayCwithDelims{\|}{\|}}
898   {\endNiceArrayCwithDelims}
899 \NewDocumentEnvironment {bNiceArrayC} {}
900   {\NiceArrayCwithDelims{[]}{[]}}
901   {\endNiceArrayCwithDelims}
902 \NewDocumentEnvironment {BNiceArrayC} {}
903   {\NiceArrayCwithDelims{\{}{\}}
904   {\endNiceArrayCwithDelims}

```

12.11 The environment {pNiceArrayRC}

The code in this section can be removed without affecting the previous code.

```

905 \keys_define:nn {NiceMatrix/NiceArrayRC}
906   {parallelize-diags .bool_set:N = \l_@@_parallelize_diags_bool,
907    parallelize-diags .default:n = true,
908    renew-dots .bool_set:N = \l_@@_renew_dots_bool,
909    renew-dots .default:n = true,
910    nullify-dots .bool_set:N = \l_@@_nullify_dots_bool ,
911    nullify-dots .default:n = true,
912    code-for-first-row .tl_set:N = \l_@@_code_for_first_row_tl,
913    code-for-first-row .value_required:n = true,
914    code-for-last-col .tl_set:N = \l_@@_code_for_last_col_tl,
915    code-for-last-col .value_required:n = true,
916    columns-width .code:n = \str_if_eq:nTF {#1} {auto}
917                                {\bool_set_true:N
918                                 \l_@@_auto_columns_width_bool}
919                                {\dim_set:Nn \l_@@_columns_width_dim {#1}},
920    columns-width .value_required:n = true,
921    name .code:n = {\seq_if_in:NnTF \g_@@_names_seq {#1}
922                           {\@_error:nn {Duplicate-name} {#1}}
923                           {\seq_gput_left:Nn \g_@@_names_seq {#1}}
924                           \tl_set:Nn \l_@@_name_tl {#1}},
925    code-after .tl_set:N = \l_@@_code_after_tl,
926    create-extra-nodes .bool_set:N = \l_@@_extra_nodes_bool,
927    create-extra-nodes .default:n = true,
928    left-margin .dim_set:N = \l_@@_left_margin_dim,
929    left-margin .default:n = \arraycolsep,
930    right-margin .dim_set:N = \l_@@_right_margin_dim,
931    right-margin .default:n = \arraycolsep,
932    extra-left-margin .dim_set:N = \l_@@_extra_left_margin_dim,
933    extra-right-margin .dim_set:N = \l_@@_extra_right_margin_dim,
934    unknown .code:n = \@_error:n {Unknown-option-for-NiceArrayRC}}
935 \msg_new:nnnn {nicematrix}
936   {Unknown-option-for-NiceArrayRC}
937   {The~option~"\tl_use:N\l_keys_key_tl"~is~unknown~for~the~environment~
938    {\currenvir}.\\
939    If~you~go~on,~it~will~be~ignored.\\
940    For~a~list~of~the~available~options,~type~H~<return>.}
941   {The~available~options~are~(in~alphabetic~order):~
942    code-after,~
943    code-for-last-col,~
944    code-for-first-row,~
945    columns-width,~}

```

```

946     create-extra-nodes, ~
947     extra-left-margin, ~
948     extra-right-margin, ~
949     left-margin, ~
950     name, ~
951     nullify-dots, ~
952     parallelize-diags, ~
953     renew-dots ~
954     and-right-margin. }

```

The first and the second argument of the environment `{NiceArrayRCwithDelims}` are the delimiters which will be used in the array. Usually, the final user will not use directly this environment `{NiceArrayRCwithDelims}` because he will use one of the variants `{pNiceArrayRC}`, `{vNiceArrayRC}`, etc.

```

955 \NewDocumentEnvironment {NiceArrayRCwithDelims} {mm O{} m !O{}}
956   {\int_zero:N \l_@@_nb_first_row_int
957    \dim_gzero_new:N \g_@@_width_last_col_dim
958    \keys_set:nn {NiceMatrix/NiceArrayRC} {#3,#5}
959    \bool_set_false:N \l_@@_exterior_arraycolsep_bool
960    \tl_set:Nn \l_@@_pos_env_tl c
961    \box_clear_new:N \l_@@_the_array_box
962    \hbox_set:Nw \l_@@_the_array_box
963      $ % $
964      \@@_NiceArrayC:n {#4}
965    {
966      \end{NiceArray}
967      $ % $
968      \hbox_set_end:
969      \dim_set:Nn \l_tmpa_dim
970        { ( \dim_max:nn {12pt}
971            { \g_@@_max_ht_row_one_dim + \g_@@_max_dp_row_zero_dim})
972            + \g_@@_max_ht_row_zero_dim
973            - \g_@@_max_ht_row_one_dim }
974      \hbox_set:Nn \l_tmpa_box
975        {$ % $
976        \left#1
977        \vcenter {\skip_vertical:n {- \l_tmpa_dim}
978                  \box_use_drop:N \l_@@_the_array_box}
979        \right#2
980        $ % $
981        \skip_horizontal:n \g_@@_width_last_col_dim}
982      \box_set_ht:Nn \l_tmpa_box {\box_ht:N \l_tmpa_box + \l_tmpa_dim}
983      \box_use_drop:N \l_tmpa_box
984    }

```

In the following environments, we don't use the form with `\begin{...}` and `\end{...}` because we use `\@currenvir` in the error message for an unknown option.

```

984 \NewDocumentEnvironment {pNiceArrayRC} {}
985   {\NiceArrayRCwithDelims{{}{}}}
986   {\endNiceArrayRCwithDelims}
987 \NewDocumentEnvironment {bNiceArrayRC} {}
988   {\NiceArrayRCwithDelims{[][]}}
989   {\endNiceArrayRCwithDelims}
990 \NewDocumentEnvironment {vNiceArrayRC} {}
991   {\NiceArrayRCwithDelims{|}{|}}
992   {\endNiceArrayRCwithDelims}
993 \NewDocumentEnvironment {VNiceArrayRC} {}
994   {\NiceArrayRCwithDelims{\|}{\|}}
995   {\endNiceArrayRCwithDelims}
996 \NewDocumentEnvironment {BNiceArrayRC} {}
997   {\NiceArrayRCwithDelims{\}{\}{\}}}
998   {\endNiceArrayRCwithDelims}

```

12.12 The extra nodes

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

```
999 \cs_generate_variant:Nn \dim_min:nn {vn}
1000 \cs_generate_variant:Nn \dim_max:nn {vn}
```

For each row i , we compute two dimensions $l_{\text{@@row}_i \text{min_dim}}$ and $l_{\text{@@row}_i \text{max_dim}}$. The dimension $l_{\text{@@row}_i \text{min_dim}}$ is the minimal y -value of all the cells of the row i . The dimension $l_{\text{@@row}_i \text{max_dim}}$ is the maximal y -value of all the cells of the row i .

Similarly, for each column j , we compute two dimensions $l_{\text{@@column}_j \text{min_dim}}$ and $l_{\text{@@column}_j \text{max_dim}}$. The dimension $l_{\text{@@column}_j \text{min_dim}}$ is the minimal x -value of all the cells of the column j . The dimension $l_{\text{@@column}_j \text{max_dim}}$ 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`.

```
1001 \cs_new_protected:Nn \@@_create_extra_nodes:
1002   {\begin{tikzpicture}[remember picture, overlay]
1003     \int_step_variable:nnNn \l_nb_first_row_int 1 \g_@@_row_int \@@_i
1004     {\dim_zero_new:c {l_@@_row_\@@_i _min_dim}
1005      \dim_set_eq:cN {l_@@_row_\@@_i _min_dim} \c_max_dim
1006      \dim_zero_new:c {l_@@_row_\@@_i _max_dim}
1007      \dim_set:cn {l_@@_row_\@@_i _max_dim} {-\c_max_dim}}
1008     \int_step_variable:nnNn 1 1 \g_@@_column_total_int \@@_j
1009     {\dim_zero_new:c {l_@@_column_\@@_j _min_dim}
1010      \dim_set_eq:cN {l_@@_column_\@@_j _min_dim} \c_max_dim
1011      \dim_zero_new:c {l_@@_column_\@@_j _max_dim}
1012      \dim_set:cn {l_@@_column_\@@_j _max_dim} {-\c_max_dim}}
```

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

```
1013   \int_step_variable:nnNn \l_nb_first_row_int 1 \g_@@_row_int \@@_i
1014     {\int_step_variable:nnNn 1 1 \g_@@_column_total_int \@@_j}
```

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

```
1015   {\cs_if_exist:cT {\pgf@sh@ns@nm-\int_use:N \g_@@_env_int-\@@_i-\@@_j}}
```

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

```
1016   {\tikz@parse@node \pgfutil@firstofone
1017     (nm-\int_use:N \g_@@_env_int-\@@_i-\@@_j.south-west)
1018     \dim_set:cn {l_@@_row_\@@_i _min_dim}
1019       {\dim_min:vn {l_@@_row_\@@_i _min_dim} \pgf@y}
1020     \seq_if_in:NxF \g_@@_multicolumn_cells_seq {\@@_i-\@@_j}
1021       {\dim_set:cn {l_@@_column_\@@_j _min_dim}
1022         {\dim_min:vn {l_@@_column_\@@_j _min_dim} \pgf@x}}}
```

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`.

```
1023   {\tikz@parse@node \pgfutil@firstofone
1024     (nm-\int_use:N \g_@@_env_int-\@@_i-\@@_j.north-east)
1025     \dim_set:cn {l_@@_row_\@@_i _max_dim}
1026       {\dim_max:vn {l_@@_row_\@@_i _max_dim} \pgf@y}
1027     \seq_if_in:NxF \g_@@_multicolumn_cells_seq {\@@_i-\@@_j}
1028       {\dim_set:cn {l_@@_column_\@@_j _max_dim}
1029         {\dim_max:vn {l_@@_column_\@@_j _max_dim} \pgf@x}}}
1030   }}
```

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`).

```
1031   \tikzset{name-suffix = -medium}
1032   \@@_create_nodes:
```

For “large nodes”, the eventual “first row” and “last column” (in environments like `{pNiceArrayRC}`) 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_@@_column_int` (and not `\g_@@_column_total_int`).²³

```
1033     \int_set:Nn \l_@@_nb_first_row_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.
1034     \int_step_variable:nnNn 1 1 {\g_@@_row_int-1} \@@_i
1035         {\dim_set:cn {l_@@_row_\@@_i_min_dim}
1036             {(\dim_use:c {l_@@_row_\@@_i_min_dim}
1037                 + \dim_use:c {l_@@_row_\int_eval:n{\@@_i+1}_max_dim}) / 2}
1038             \dim_set_eq:cc {l_@@_row_\int_eval:n{\@@_i+1}_max_dim}
1039                 {l_@@_row_\@@_i_min_dim} }
1040     \int_step_variable:nnNn 1 1 {\g_@@_column_int-1} \@@_j
1041         {\dim_set:cn {l_@@_column_\@@_j_max_dim}
1042             {(\dim_use:c {l_@@_column_\@@_j_max_dim}
1043                 + \dim_use:c {l_@@_column_\int_eval:n{\@@_j+1}_min_dim}) / 2}
1044             \dim_set_eq:cc {l_@@_column_\int_eval:n{\@@_j+1}_min_dim}
1045                 {l_@@_column_\@@_j_max_dim} }
1046     \dim_sub:cn {l_@@_column_1_min_dim} \g_@@_left_margin_dim
1047     \dim_add:cn {l_@@_column_\int_use:N \g_@@_column_int_max_dim}
1048             \g_@@_right_margin_dim
```

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

```
1049     \tikzset{name~suffix = -large}
1050     \@@_create_nodes:
1051     \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.

```
1052     \cs_set:Nn \@@_create_extra_nodes: {}
```

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.

```
1053 \cs_new_protected:Nn \@@_create_nodes:
1054     {\int_step_variable:nnNn \l_@@_nb_first_row_int 1 \g_@@_row_int \@@_i
1055         {\int_step_variable:nnNn 1 1 \g_@@_column_total_int \@@_j}
```

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

```
1056     {\coordinate (\@@_south-west)
1057         at (\dim_use:c {l_@@_column_\@@_j_min_dim},
1058                 \dim_use:c {l_@@_row_\@@_i_min_dim}) ;
1059     \coordinate (\@@_north-east)
1060         at (\dim_use:c {l_@@_column_\@@_j_max_dim},
1061                 \dim_use:c {l_@@_row_\@@_i_max_dim}) ;
```

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

```
1062     \draw node [fit = {(\@@_south-west) (\@@_north-east)},
1063                 inner_sep=0pt,
1064                 name = nm-\int_use:N \g_@@_env_int-\@@_i-\@@_j,
1065                 alias = \tl_if_empty:NF \g_@@_name_tl
1066                     {\tl_use:N \g_@@_name_tl-\@@_i-\@@_j}]
```

²³We recall that `\g_@@_column_total_int` is equal to `\g_@@_column_int` except if there is an exterior column. In this case, `\g_@@_column_total_int` is equal to `\g_@@_column_int + 1`.

```

1067         {} ;
1068     }
1069 }

```

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 correspondant values of n .

```

1070     \@@_seq_mapthread_function:NNN \g_@@_multicolumn_cells_seq
1071             \g_@@_multicolumn_sizes_seq
1072             \@@_node_for_multicolumn:nn
1073 }

```

```

1074 \cs_set:Npn \@@_extract_coords: #1-#2\q_stop{\cs_set:Npn \@@_i {#1}
1075                                     \cs_set:Npn \@@_j {#2}}

```

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

```

1076 \cs_new_protected:Nn \@@_node_for_multicolumn:nn
1077     {\@@_extract_coords: #1\q_stop
1078         \coordinate (@@-south-west)
1079             at (\dim_use:c {l_@@_column_\@@_j _min_dim},
1080                 \dim_use:c {l_@@_row_\@@_i _min_dim}) ;
1081         \coordinate (@@-north-east)
1082             at (\dim_use:c {l_@@_column_\int_eval:n{\@@_j+#2-1}_max_dim},
1083                 \dim_use:c {l_@@_row_\@@_i _max_dim}) ;
1084         \draw node [fit = {(@@-south-west) (@@-north-east)},
1085                     inner_sep=0pt,
1086                     name = nm-\int_use:N \g_@@_env_int-\@@_i-\@@_j,
1087                     alias = \tl_if_empty:NF \g_@@_name_tl
1088                             {\tl_use:N \g_@@_name_tl-\@@_i-\@@_j}]
1089             {} ;
1090     }

```

12.13 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` execute the code `\cs_set_eq:NN \env@matrix \NiceMatrix`.

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

```
1091 \ProcessKeysOptions {NiceMatrix}
```

12.14 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.

```

1092 \cs_new:Npn \@@_seq_mapthread_function:NNN #1#2#3
1093     {\group_begin:

```

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

```

1094     \int_step_inline:nnnn 1 1 {\seq_count:N #1}
1095         {\seq_pop:NN #1 \l_tmpa_tl
1096             \seq_pop:NN #2 \l_tmpb_tl
1097             \exp_args:NVV #3 \l_tmpa_tl \l_tmpb_tl}
1098     \group_end:
1099 }

```

```

1100 \cs_set:Nn \@@_renew_matrix:
1101   {\RenewDocumentEnvironment {pmatrix} {}
1102     {\begin{pNiceMatrix}}
1103     {\end{pNiceMatrix}}
1104   \RenewDocumentEnvironment {vmatrix} {}
1105     {\begin{vNiceMatrix}}
1106     {\end{vNiceMatrix}}
1107   \RenewDocumentEnvironment {Vmatrix} {}
1108     {\begin{VNiceMatrix}}
1109     {\end{VNiceMatrix}}
1110   \RenewDocumentEnvironment {bmatrix} {}
1111     {\begin{bNiceMatrix}}
1112     {\end{bNiceMatrix}}
1113   \RenewDocumentEnvironment {Bmatrix} {}
1114     {\begin{BNiceMatrix}}
1115     {\end{BNiceMatrix}}}

```

13 History

13.1 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.

13.2 Changes between versions 1.1 and 1.2

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

13.3 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).

13.4 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.

13.5 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.

13.6 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”.

13.7 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 StackExchange²⁴, Tikz externalization is now deactivated in the environments of the extension `nicematrix`.²⁵

13.8 Changes between version 2.1 and 2.1.2

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

13.9 Changes between version 2.1.2 and 2.1.3

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

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

13.10 Changes between version 2.1.3 and 2.1.4

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

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

²⁴cf. tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package

²⁵Before 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`.