The package nicematrix*

F. Pantigny
fpantigny@wanadoo.fr

June 21, 2025

Abstract

The LaTeX package nicematrix provides new environments similar to the classical environments `{tabular}`, `{array}` and `{matrix}` of array and amsmath but with extended features.

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

Product	dimensions (cm)			Price
	L	l	h	
small	3	5.5	1	30
standard	5.5	8	1.5	50.5
premium	8.5	10.5	2	80
extra	8.5	10	1.5	85.5
special	12	12	0.5	70

The package nicematrix is entirely contained in the file `nicematrix.sty`. This file may be put in the current directory or in a `texmf` tree. However, the best is to install nicematrix with a TeX distribution such as MiKTeX, TeX Live or MacTeX.

Remark
If you use LaTeX via Internet with, for example, Overleaf, you can upload the file `nicematrix.sty` in the repertory of your project in order to take full advantage of the latest version de nicematrix.[1]

This package can be used with `xelatex`, `lualatex`, `pdflatex` but also by the classical workflow `latex-dvips-ps2pdf` (or Adobe Distiller). *However, the file nicematrix.dtx of the present documentation should be compiled with LuaLaTeX.*

This package requires and **loads** the packages l3keys2e, array, amsmath, pgfcore and the module shapes of PGF (tikz, which is a layer over PGF, is *not* loaded). The final user only has to load the package with `\usepackage{nicematrix}`.

The idea of nicematrix is to create PGF nodes under the cells and the positions of the rules of the tabular created by array and to use these nodes to develop new features. As usual with PGF, the coordinates of these nodes are written in the `aux` to be used on the next compilation and that's why nicematrix may need **several compilations**[2]. One must not use the command `\nofiles` (which prevents the creation of the `aux` file).

Most features of nicematrix may be used without explicit use of PGF or TikZ (which, in fact, is not loaded by default).

A command `\NiceMatrixOptions` is provided to fix the options (the scope of the options fixed by this command is the current TeX group: they are semi-global).

*This document corresponds to the version 7.1d of nicematrix, at the date of 2025/06/21.
[1]The latest version of the file `nicematrix.sty` may be downloaded on the Github depot of nicematrix: `https://github.com/fpantigny/nicematrix/releases`
[2]If you use Overleaf, Overleaf will do automatically a sufficient number of compilations.

1 The environments of this package

The package nicematrix defines the following new environments.

`{NiceTabular}`	`{NiceArray}`	`{NiceMatrix}`
`{NiceTabular*}`	`{pNiceArray}`	`{pNiceMatrix}`
`{NiceTabularX}`	`{bNiceArray}`	`{bNiceMatrix}`
	`{BNiceArray}`	`{BNiceMatrix}`
	`{vNiceArray}`	`{vNiceMatrix}`
	`{VNiceArray}`	`{VNiceMatrix}`

The environments `{NiceArray}`, `{NiceTabular}` and `{NiceTabular*}` are similar to the environments `{array}`, `{tabular}` and `{tabular*}` of the package array (which is loaded by nicematrix).

The environments `{pNiceArray}`, `{bNiceArray}`, etc. have no equivalent in array.

The environments `{NiceMatrix}`, `{pNiceMatrix}`, etc. are similar to the corresponding environments of amsmath (which is loaded by nicematrix): `{matrix}`, `{pmatrix}`, etc.

The environment `{NiceTabularX}` is similar to the environment `{tabularx}` from the eponymous package.[3].

It's recommended to use primarily the classical environments and to use the environments of nicematrix only when some feature provided by these environments is used (this will save memory).

All the environments of the package nicematrix accept, between square brackets, an optional list of *key=value* pairs. **There must be no space before the opening bracket (`[`) of this list of options.**

2 The vertical space between the rows

It's well known that some rows of the arrays created by default with LaTeX are, by default, too close to each other. Here is a classical example.

```
$\begin{pmatrix}
\frac{1}{2} & -\frac{1}{2} \\
\frac{1}{3} & \frac{1}{4} \\
\end{pmatrix}$
```

$$\begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{3} & \frac{1}{4} \end{pmatrix}$$

Inspired by the package cellspace which deals with that problem, the package nicematrix provides two keys `cell-space-top-limit` and `cell-space-bottom-limit` similar to the parameters of cellspace called `\cellspacetoplimit` and `\cellspacebottomlimit`.
There is also a key `cell-space-limits` to set both parameters at once.
The initial value of these parameters is 0 pt in order to have for the environments of nicematrix the same behaviour as those of array and amsmath. However, a value of 1 pt would probably be a good choice and we suggest to set them with `\NiceMatrixOptions`.[4]

```
\NiceMatrixOptions{cell-space-limits = 1pt}
```

```
$\begin{pNiceMatrix}
\frac12 & -\frac12 \\
\frac13 & \frac14 \\
\end{pNiceMatrix}$
```

$$\begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{3} & \frac{1}{4} \end{pmatrix}$$

[3]In fact, it's possible to use directly the `X` columns in the environment `{NiceTabular}` (and the required width for the tabular is fixed by the key `width`): cf. p. 28

[4]One should remark that these parameters apply also to the columns of type `S` of siunitx whereas the package cellspace is not able to act on such columns of type `S`.

It's also possible to change these parameters for only a few rows by using the command `\RowStyle` provided by nicematrix (cf. p. 26).

3 The vertical position of the arrays

The package nicematrix provides a option `baseline` for the vertical position of the arrays. This option takes in as value an integer which is the number of the row on which the array will be aligned.

```
$A = \begin{pNiceMatrix}[baseline=2]
\frac{1}{\sqrt{1+p^2}} & p & 1-p \\
1 & 1 & 1 \\
1 & p & 1+p
\end{pNiceMatrix}$
```

$$A = \begin{pmatrix} \frac{1}{\sqrt{1+p^2}} & p & 1-p \\ 1 & 1 & 1 \\ 1 & p & 1+p \end{pmatrix}$$

It's also possible to use the option `baseline` with one of the special values `t`, `c` or `b`. These letters may also be used absolutely like the option of the environments `{tabular}` and `{array}` of array. The initial value of `baseline` is `c`.

In the following example, we use the option `t` (equivalent to `baseline=t`) immediately after an `\item` of list. One should remark that the presence of a `\hline` at the beginning of the array doesn't prevent the alignment of the baseline with the baseline of the first row (with `{tabular}` or `{array}` of array, one must use `\firsthline`).

```
\begin{enumerate}
\item an item
\smallskip
\item \renewcommand{\arraystretch}{1.2}
$\begin{NiceArray}[t]{lcccccc}
\hline
n   & 0 & 1 & 2 & 3 & 4  & 5 \\
u_n & 1 & 2 & 4 & 8 & 16 & 32
\hline
\end{NiceArray}$
\end{enumerate}
```

1. an item
2.

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

However, it's also possible to use the tools of booktabs[5]: `\toprule`, `\bottomrule`, `\midrule`, etc.

```
\begin{enumerate}
\item an item
\smallskip
\item
$\begin{NiceArray}[t]{lcccccc}
\toprule
n   & 0 & 1 & 2 & 3 & 4  & 5 \\
\midrule
u_n & 1 & 2 & 4 & 8 & 16 & 32
\bottomrule
\end{NiceArray}$
\end{enumerate}
```

1. an item
2.

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

It's also possible to use the key `baseline` to align a matrix on an horizontal rule (drawn by `\hline`). In this aim, one should give the value `line-`*i* where *i* is the number of the row *following* the horizontal rule.

```
\NiceMatrixOptions{cell-space-limits=1pt}
```

[5]The extension booktabs is *not* loaded by nicematrix.

```
$A=\begin{pNiceArray}{cc|cc}[baseline=line-3]
\dfrac{1}{A} & \dfrac{1}{B} & 0 & 0 \\
\dfrac{1}{C} & \dfrac{1}{D} & 0 & 0 \\
\hline
0 & 0 & A & B \\
0 & 0 & D & D \\
\end{pNiceArray}$
```

$$A=\left(\begin{array}{cc|cc}\dfrac{1}{A} & \dfrac{1}{B} & 0 & 0 \\ \dfrac{1}{C} & \dfrac{1}{D} & 0 & 0 \\ \hline 0 & 0 & A & B \\ 0 & 0 & D & D \end{array}\right)$$

4 The blocks

4.1 General case

In the environments of nicematrix, it's possible to use the command `\Block` in order to place an element in the center of a rectangle of merged cells of the array.[6]

The command `\Block` must be used in the upper leftmost cell of the cells of the block with two mandatory arguments.

- The first argument is the size of the block with the syntax i-j where i is the number of rows of the block and j its number of columns.

  If this argument is empty, its default value is `1-1`. If the number of rows is not specified, or equal to `*`, the block extends until the last row (idem for the columns).

- The second argument is the content of the block. In `{NiceTabular}`, `{NiceTabular*}` and `{NiceTabularX}`, the content of the block is composed in text mode whereas, in the other environments, it is composed in math mode.

Here is an example of utilisation of the command `\Block` in mathematical matrices.

```
$\begin{bNiceArray}{cw{c}{1cm}c|c}[margin]
\Block{3-3}{A} & & & 0 \\
& & & \Vdots \\
& & & 0 \\
\hline
0 & \Cdots& 0 & 0
\end{bNiceArray}$
```

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

One may wish to raise the size of the "A" placed in the block of the previous example. Since this element is composed in math mode, it's not possible to use directly a command like `\large`, `\Large` and `\LARGE`. That's why the command `\Block` provides an option between angle brackets to specify some TeX code which will be inserted before the beginning of the math mode.[7]

```
$\begin{bNiceArray}{cw{c}{1cm}c|c}[margin]
\Block{3-3}<\Large>{A} & & & 0 \\
0 & & & \Vdots \\
& & & 0 \\
\hline
0 & \Cdots& 0 & 0
\end{bNiceArray}$
```

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

[6]The spaces after a command `\Block` are deleted.

[7]This argument between angular brackets may also be used to insert a command of font such as `\bfseries` when the command `\\` is used in the content of the block. It's also possible to put in that optional argument the command `\rotate` provided by nicematrix (cf. part 14.5, p. 51).

In fact, the command `\Block` accepts as first optional argument (between square brackets) a list of pairs *key=value*.
First, there are keys which are quick tools to control the appearance of the block.

- the key `fill` takes in as value a color and fills the block with that color;
- the key `opacity` sets the opacity of the filling color specified by `fill`;[8]
- the key `draw` takes in as value a color and strokes the frame of the block with that color (the default value of that key is the current color of the rules of the array);
- the key `color` takes in as value a color and apply that color the content of the block but draws also the frame of the block with that color;
- the keys `hlines`, `vlines` and `hvlines` draw all the corresponding rules in the block;[9]
- the key `line-width` is the width of the rules (is relevant only when one of the keys `draw`, `hvlines`, `vlines` and `hlines` is used);
- the key `rounded-corners` requires rounded corners (for the frame drawn by `draw` and the shape drawn by `fill`) with a radius equal to the value of that key (the default value is 4 pt[10]).

Sometimes, these tools are not sufficient to control the appearance of the block. The following keys are more powerful but also more difficult to use. Moreover, the require the loading of TikZ by the user (with `\usepackage{tikz}`). By default, nicematrix does not load TikZ but only PGF, which is a sublayer of TikZ.

- The key `borders` provides the ability to draw only some borders of the blocks; the value of that key is a (comma-separated) list of elements covered by `left`, `right`, `top` and `bottom`; it's possible, in fact, in the list which is the value of the key `borders`, to add an entry of the form `tikz={`*list*`}` where *list* is a list of couples *key=value* of TikZ specifying the graphical characteristics of the lines that will be drawn (for an example, see p. 69).
- When the key `tikz` is used, the TikZ path corresponding of the rectangle which delimits the block is executed with TikZ[11] by using as options the value of that key `tikz` (which must be a list of keys allowed for a TikZ path).

  In fact, in the list of the keys provided by the user as value of `tikz`, it's possible to put a key `offset`. That key is not provided by TikZ but by nicematrix. It will narrow the rectangular frame corresponding to the block by a margin (horizontally and vertically) equal to the value (of that key `offset`). That new frame, a bit narrower, will be executed by TikZ with options which are the other keys in the list of keys provided as value to the key `tikz` of `\Block`.

  For examples, cf. p. 63.

There is also some technical keys:

- the key `name` provides a name to the rectangular TikZ node corresponding to the block; it's possible to use that name with TikZ in the `\CodeAfter` of the environment (cf. p. 39);
- the key `respect-arraystretch` prevents the setting of `\arraystretch` to 1 at the beginning of the block (which is the behaviour by default) ;
- By default, the rules are not drawn in the blocks (see the section about the rules: section 5 p. 11). However, if the key `transparent` is used, the rules are drawn. For an example, see section 18.1 on page 63. Caution: that key does not imply that the content of the block will be transparent!

[8]Caution: that feature creates instructions of transparency in the PDF and some readers of PDF don't support such instructions.
[9]However, the rules are not drawn in the sub-blocks of the block, as always with nicematrix: the rules are not drawn in the blocks, except when they have the key `transparent` (cf. section 5 p. 11).
[10]This value is the initial value of the *rounded corners* of TikZ.
[11]TikZ should be loaded (by default, nicematrix only loads PGF) and, if it's not, an error will be raised.

There is also keys for the horizontal and vertical positions of the content of the block: cf. 4.5 p. 7.

One must remark that, by default, the commands `\Blocks` don't create space. There is exception only for the blocks mono-column and the blocks mono-row under some conditions as explained just below.

In the following example, we have had to enlarge by hand the columns 2 and 3 (with the construction `w{c}{...}` of array).

```
\begin{NiceTabular}{cw{c}{2cm}w{c}{3cm}c}
rose & tulip & daisy & dahlia \\
violet
  & \Block[draw=red,fill=[RGB]{204,204,255},rounded-corners]{2-2}
          {\LARGE Some beautiful flowers} & & marigold \\
   & & marigold \\
iris & & & lis \\
arum & periwinkle & forget-me-not &  hyacinth
\end{NiceTabular}
```

rose	tulip	daisy	dahlia
violet	Some beautiful flowers		marigold
iris			lis
arum	periwinkle	forget-me-not	hyacinth

4.2 The mono-column blocks

The mono-column blocks have a special behaviour.

- The natural width of the contents of these blocks is taken into account for the width of the current column.

  In the columns with a fixed width (columns `w{...}{...}`, `W{...}{...}`, `p{...}`, `b{...}`, `m{...}`, `V` and `X`), the content of the block is formatted as a paragraph of that width.

- The specification of the horizontal position provided by the type of column (`c`, `r` or `l`) is taken into account for the blocks. For a block in a column ot type `p{...}`, `b{...}`, `m{...}`, `V{...}` ou `X`, the alignment `c` will be used by default. However, those types of columns may have an optional argument for the horizontal alignment (eg: `p[l]{...}`) and, in that case, that type of alignement is passed to the block.

  Of course, the `\Block` may also have its own specification of alignment: cf. 4.5 p. 7.

- The specifications of font specified for the column by a construction `>{...}` in the preamble of the array are taken into account for the mono-column blocks of that column (this behaviour is probably expected).

```
\begin{NiceTabular}{@{}>{\bfseries}lr@{}} \hline
\Block{2-1}{John}    & 12 \\
                     & 13 \\ \hline
Steph                &  8 \\ \hline
\Block{3-1}{Sarah}   & 18 \\
                     & 17 \\
                     & 15 \\ \hline
Ashley               & 20 \\ \hline
Henry                & 14 \\ \hline
\Block{2-1}{Madison} & 15 \\
                     & 19 \\ \hline
\end{NiceTabular}
```

John	12
	13
Steph	8
	18
Sarah	17
	15
Ashley	20
Henry	14
Madison	15
	19

4.3 The mono-row blocks

For the mono-row blocks, the natural height and depth are taken into account for the height and depth of the current row (as does a standard `\multicolumn` of LaTeX), except when an option of vertical position has been used for the block (one of the keys `t`, `b`, `m`, `T` and `B` described in the part 4.6, p. 9).

4.4 The mono-cell blocks

A mono-cell block inherits all the properties of the mono-row blocks and mono-column blocks.

At first sight, one may think that there is no point using a mono-cell block. However, there are some good reasons to use such a block.

- It's possible to use the command `\\` in a (mono-cell) block.
- It's possible to use the option of horizontal alignment of the block in derogation of the type of column given in the preamble of the array.
- It's possible do draw a frame around the cell with the key `draw` of the command `\Block` and to fill the background with rounded corners with the keys `fill` and `rounded-corners`.[12]
- It's possible to draw one or several borders of the cell with the key `borders`.

```
\begin{NiceTabular}{cc}
\toprule
Writer & \Block[l]{}{year\\ of birth} \\
\midrule
Hugo & 1802 \\
Balzac & 1799 \\
\bottomrule
\end{NiceTabular}
```

Writer	year of birth
Hugo	1802
Balzac	1799

We recall that if the first mandatory argument of `\Block` is left blank, the block is mono-cell.[13]

4.5 Horizontal position of the content of the block

The command `\Block` accepts the keys `l`, `c` and `r` for the horizontal position of its content.

```
$\begin{bNiceArray}{cw{c}{1cm}c|c}[margin]
\Block[r]{3-3}<\LARGE>{A} & & & 0 \\
& & & \Vdots \\
& & & 0 \\
\hline
0 & \Cdots& 0 & 0
\end{bNiceArray}$
```

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

By default, the horizontal position of the content of a block is computed by using the positions of the *contents* of the columns implied in that block. That's why, in the following example, the header "First group" is correctly centered despite the instruction `!{\qquad}` in the preamble which has been used to increase the space between the columns (this is not the behaviour of `\multicolumn`).

[12] If one simply wishes to color the background of a unique cell, there is no point using the command `\Block`: it's possible to use the command `\cellcolor`.

[13] One may consider that the default value of the first mandatory argument of `\Block` is `1-1`.

```
\begin{NiceTabular}{@{}c!{\qquad}ccc!{\qquad}ccc@{}}
\toprule
Rank & \Block{1-3}{First group} & & & \Block{1-3}{Second group} \\
     & 1A & 1B & 1C & 2A & 2B & 2C \\
\midrule
 1 & 0.657 & 0.913 & 0.733 & 0.830 & 0.387 & 0.893\\
 2 & 0.343 & 0.537 & 0.655 & 0.690 & 0.471 & 0.333\\
 3 & 0.783 & 0.885 & 0.015 & 0.306 & 0.643 & 0.263\\
 4 & 0.161 & 0.708 & 0.386 & 0.257 & 0.074 & 0.336\\
\bottomrule
\end{NiceTabular}
```

Rank	First group			Second group		
	1A	1B	1C	2A	2B	2C
1	0.657	0.913	0.733	0.830	0.387	0.893
2	0.343	0.537	0.655	0.690	0.471	0.333
3	0.783	0.885	0.015	0.306	0.643	0.263
4	0.161	0.708	0.386	0.257	0.074	0.336

In order to have an horizontal positioning of the content of the block computed with the limits of the columns of the LaTeX array (and not with the contents of those columns), one may use the key `L`, `R` and `C` of the command `\Block`.[14]

Here is the same example with the key `C` for the first block.

```
\begin{NiceTabular}{@{}c!{\qquad}ccc!{\qquad}ccc@{}}
\toprule
Rank & \Block[C]{1-3}{First group} & & & \Block{1-3}{Second group} \\
     & 1A & 1B & 1C & 2A & 2B & 2C \\
\midrule
 1 & 0.657 & 0.913 & 0.733 & 0.830 & 0.387 & 0.893\\
 2 & 0.343 & 0.537 & 0.655 & 0.690 & 0.471 & 0.333\\
 3 & 0.783 & 0.885 & 0.015 & 0.306 & 0.643 & 0.263\\
 4 & 0.161 & 0.708 & 0.386 & 0.257 & 0.074 & 0.336\\
\bottomrule
\end{NiceTabular}
```

Rank	First group			Second group		
	1A	1B	1C	2A	2B	2C
1	0.657	0.913	0.733	0.830	0.387	0.893
2	0.343	0.537	0.655	0.690	0.471	0.333
3	0.783	0.885	0.015	0.306	0.643	0.263
4	0.161	0.708	0.386	0.257	0.074	0.336

The command `\Block` supports also the keys `p` and `j`. With the key `p`, the content of the block is formatted like a paragraph (as in a column of type `p`). That key may be used in conjunction with a key `l`, `c` or `r`, and, in that case, the paragraph is formatted with `\raggedright`, `\centering` or

[14]Remark: The keys `L`, `C` and `R` require less computations than the keys `l`, `c` and `r`. If you are concernend about efficiency, you should use by default `\Block[C]`.

\raggedleft (or \RaggedRight, \Centering and \RaggedLeft when ragged2e) est chargée. With the key j (which inforces the key p), the paragraph is justified.

It's possible to put an environment {itemize} or {enumerate} in a block which uses the key p or j (in the other case, you will have a LaTeX error: Not allowed in LR mode). For the following example, we have loaded the package enumitem (for the key left of the environment {itemize}).

```
\begin{NiceTabular}[hvlines]{ccc}
one & two two & three three \\
one &
\Block[p]{*-2}{%
\begin{itemize}[left=0pt]
\item one two three four five
\item two
\item three
\end{itemize}
} \\
one & \\
one & \\
one & \\
one & \\
one & \\
one & \\
\end{NiceTabular}
```

<table>
<tr><td>one</td><td>two two</td><td>three three</td></tr>
<tr><td>one</td><td colspan="2" rowspan="7">one two three four five
two
three</td></tr>
<tr><td>one</td></tr>
<tr><td>one</td></tr>
<tr><td>one</td></tr>
<tr><td>one</td></tr>
<tr><td>one</td></tr>
<tr><td>one</td></tr>
</table>

4.6 Vertical position of the content of the block

For the vertical position, the command \Blocks accepts the keys m, t, b, T and B.

- With the key m[15], the content of the block is vertically centered.
- With the key t, the baseline of the content of the block is aligned with the baseline of the first row concerned by the block.
- with the key b, the baseline of the last row of the content of the block (we recall that the content of a block may contain several lines of text separated by \\) is aligned with the baseline of the last of the rows of the array involved in the block.
- With the key T, the content of the block is set upwards.

  No vertical margin is added. However, the contents of the block is (always) composed by nicematrix in a {minipage}, a {tabular} or an {array} and, hence, there will still remain a margin (in most cases). If needed, it's always possible to add a \strut...
- With the key B, the content of the block is set downwards.

When no key is given, the key m applies (except in the mono-row blocks).

```
\NiceMatrixOptions{rules/color=[gray]{0.75}, hvlines}
```

[15]That key has an alias: v-center.

```
\begin{NiceTabular}{ccc}
\Block[fill=red!10,t,l]{4-2}{two\\lines}
 & & \Huge Un\\
 & & deux  \\
 & & trois  \\
 & & \Huge quatre  \\
text & text \\
\end{NiceTabular}
```

two lines		Un
		deux
		trois
		quatre
text	text	

```
\begin{NiceTabular}{ccc}
\Block[fill=red!10,b,r]{4-2}{two\\lines}
 & & \Huge Un\\
 & & deux  \\
 & & trois  \\
 & & \Huge quatre  \\
text & text \\
\end{NiceTabular}
```

two lines		Un
		deux
		trois
		quatre
text	text	

```
\begin{NiceTabular}{ccc}
\Block[fill=red!10,T,L]{4-2}{two\\lines}
 & & \Huge Un\\
 & & deux  \\
 & & trois  \\
 & & \Huge quatre  \\
text & text \\
\end{NiceTabular}
```

two lines		Un
		deux
		trois
		quatre
text	text	

```
\begin{NiceTabular}{ccc}
\Block[fill=red!10,B,R]{4-2}{two\\lines}
 & & \Huge Un\\
 & & deux  \\
 & & trois  \\
 & & \Huge quatre  \\
text & text \\
\end{NiceTabular}
```

two lines		Un
		deux
		trois
		quatre
text	text	

4.7 \\ and & in the blocks

The extension nicematrix provides the ability to use `\\` and `&` directly in the content of a block (in order to format its contents) but there is some rectrictions.

- One must not use both `\\` and `&` in the same block.
- For `\\`, there is no other restriction. It's possible to use `\\` in a block to format a text on several lines.
- In order to use `&`, the key `ampersand-in-blocks` (alias: `&-in-blocks`) must been activated[16]. Then, the block is divided in sub-blocks as illustrated below. Be careful: when `ampersand-in-blocks` is in force, the (main) argument of the command `\Block` is syntactically divided into sub-blocks by splitting on the ampersands `&`, the ampersands between curly braced are protected but not those in an environment.[17]

[16]Otherwise, the use of `&` in the command `\Block` will raise a TeX error :
`! Extra alignment tab has been changed to \cr.`

[17]It's not possible to write `\Block[ampersand-in-blocks]{}{\begin{array}{cc}1&2\end{array}}`. Of course, it's possible without the key `ampersand-in-blocks`.

With the ampserand `&`, it's possible to divide horizontally a block in sub-blocks of *the same size*.

```
\begin{NiceTabular}{ll}%
   [hvlines,ampersand-in-blocks]
  & the five first naturels numbers \\
3 & \Block{}{one & two & three} \\
4 & \Block{}{one& two & three & four} \\
5 & \Block{}{one & two & three & four & five} \\
\end{NiceTabular}
```

	the five first naturels numbers				
3	one	two	three		
4	one	two	three	four	
5	one	two	three	four	five

As we can see, the blocks (which was are in fact mono-cell blocks) are divided into sub-blocks of the *same size*. However, maybe the following code would be prefered.

```
\begin{NiceTabular}{lccccc}%
   [hvlines,ampersand-in-blocks]
  & \Block{1-5}{the five first
                natural numbers} \\
3 & \Block{1-5}{one & two & three} \\
4 & \Block{1-5}{one& two & three & four} \\
5 & one & two & three & four & five \\
\end{NiceTabular}
```

	the five first natural numbers				
3	one	two	three		
4	one	two	three	four	
5	one	two	three	four	five

In this code, we have blocks of size `1-5` which are divided into three or four sub-blocks.

5 The rules

The usual techniques for the rules may be used in the environments of nicematrix (except `\vline`). However, there is some small differences with the classical environments.

5.1 Some differences with the classical environments

5.1.1 The vertical rules

In the environments of nicematrix, the vertical rules specified by | in the preambles of the environments are never broken, even by an incomplete row or by a double horizontal rule specified by `\hline\hline` (there is no need to use the package hhline).

```
\begin{NiceTabular}{|c|c|} \hline
First & Second \\ \hline\hline
Peter \\ \hline
Mary & George\\ \hline
\end{NiceTabular}
```

First	Second
Peter	
Mary	George

However, the vertical rules are not drawn in the blocks (created by `\Block`: cf. p. 4) nor in the corners (created by the key `corner`: cf. p. 14) nor in the potential exterior rows (created by the keys `first-row` and `last-row`: cf. p. 29).

If you use booktabs (which provides `\toprule`, `\midrule`, `\bottomrule`, etc.) and if you actually want to add vertical rules (which is not in the spirit of booktabs), you should notice that the vertical rules drawn by nicematrix are compatible with booktabs. Remark that nicematrix does *not* load booktabs.

```
$\begin{NiceArray}{c|ccc} \toprule
a & b & c & d \\ \midrule
1 & 2 & 3 & 4 \\
1 & 2 & 3 & 4 \\ \bottomrule
\end{NiceArray}$
```

a	b	c	d
1	2	3	4
1	2	3	4

However, it's still possible to define a specifier (named, for instance, `I`) to draw vertical rules with the standard behaviour of array.

```
\newcolumntype{I}{!{\vrule}}
```

5.1.2 The command \cline

The horizontal and vertical rules drawn by `\hline` and the specifier "`|`" make the array larger or wider by a quantity equal to the width of the rule (with array and also with nicematrix).

For historical reasons, this is not the case with the command `\cline`, as shown by the following example.

```
\setlength{\arrayrulewidth}{2pt}
\begin{tabular}{cccc} \hline
A&B&C&D \\ \cline{2-2}
A&B&C&D \\ \hline
\end{tabular}
```

A	B	C	D
A	B	C	D

In the environments of nicematrix, this situation is corrected (it's still possible to go to the standard behaviour of `\cline` with the key `standard-cline`).

```
\setlength{\arrayrulewidth}{2pt}
\begin{NiceTabular}{cccc} \hline
A&B&C&D \\ \cline{2}
A&B&C&D \\ \hline
\end{NiceTabular}
```

A	B	C	D
A	B	C	D

In the environments of nicematrix, an instruction `\cline{i}` is equivalent to `\cline{i-i}`.

5.2 The thickness and the color of the rules

The environments of nicematrix provide a key `rules/width` to set the width (in fact the thickness) of the rules in the current environment. In fact, this key merely sets the value of the standard LaTeX length `\arrayrulewidth`.

It's well known that colortbl provides the command `\arrayrulecolor` in order to specify the color of the rules.

With nicematrix, it's possible to specify the color of the rules even when colortbl is not loaded. For sake of compatibility, the command is also named `\arrayrulecolor`. However, nicematrix also provides a key `rules/color`, available in `\NiceMatrixOptions` or in an individual environment, to fix the color of the rules. This key sets the value locally (whereas `\arrayrulecolor` acts globally!) and should be preferred.

```
\begin{NiceTabular}{|ccc|}[rules/color=[gray]{0.9},rules/width=1pt]
\hline
rose & tulipe & lys \\
arum & iris & violette \\
muguet & dahlia & souci \\
\hline
\end{NiceTabular}
```

rose	tulipe	lys
arum	iris	violette
muguet	dahlia	souci

In fact, in that example, instead of `\hline`, it would have a better choice to use `\Hline`, provided by nicematrix and described just below, because it ensures a better output in the PDF viewers at the low levels of zoom.

5.3 The tools of nicematrix for the rules

Here are the tools provided by nicematrix for the rules.

- the keys `hlines`, `vlines`, `hvlines` and `hvlines-except-borders`;
- the specifier "`|`" in the preamble (for the environments with preamble);
- the command `\Hline`.

All these tools don't draw the rules in the blocks nor in the empty corners (when the key `corners` is used), nor in the exterior rows and columns.

- These blocks are:
  - the blocks created by the command `\Block`[18] presented p. 4;
  - the blocks implicitly delimited by the continuous dotted lines created by `\Cdots`, `\Vdots`, etc. (cf. p. 31).
- The corners are created by the key `corners` explained below (see p. 14).
- For the exterior rows and columns, see p. 29.

In particular, this remark exhibits a first difference between the standard command `\hline` and the command `\Hline` provided by nicematrix.
Moreover, the key `\Hline` takes in an optional argument (between square brackets) which is a list of *key=value* pairs. For the description of those keys, see `custom-line` on p. 16.[19]

As well as the command `\Hline`, the specifier "`|`" supports an optional argument between square brackets for the characteristics of the rule.

```
\begin{NiceTabular}{ | c |[color=blue] c |}
\Hline
a & b \\
\Hline[color=red]
c & d \\
\Hline
\end{NiceTabular}
```

a	b
c	d

5.3.1 The keys hlines and vlines

The keys `hlines` and `vlines` (which draw, of course, horizontal and vertical rules) take in as value a list of numbers which are the numbers of the rules to draw.[20]
In fact, for the environments with delimiters (such as `{pNiceMatrix}` or `{bNiceArray}`), the key `vlines` don't draw the exterior rules (this is certainly the expected behaviour).

```
$\begin{pNiceMatrix}[vlines,rules/width=0.2pt]
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 2 & 3 & 4 & 5 & 6
\end{pNiceMatrix}$
```

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

When the key `hlines` is in force, it's still possible to use `\Hline\Hline` to put a double horizontal rule. As well, it's possible to put `||` in the preamble (of an environment with preamble) to put a double vertical rule, even when the key `vlines` is in force.

[18]And also the command `\multicolumn` but it's recommended to use instead `\Block` in the environments of nicematrix.
[19]Technical remark: If the user wants to write a command over the command `\Hline`, it shall be ensured that this new command is expandable in the TeX sens (by using, for instance, `\NewExpandableDocumentCommand` of LaTeX3, `\newcommand` of LaTeX or `\def` of TeX). Example: `\NewExpandableDocumentCommand}{}{\Hline[color=red]}`
[20]It's possible to put in that list some intervals of integers with the syntax i-j.

```
$\begin{NiceArray}{c||ccccc}[hlines,vlines]
  & a & b & c & d & e \\ \Hline\Hline
x & 0 & 0 & 0 & 0 & 0 \\
y & 0 & 0 & 0 & 0 & 0 \\
z & 0 & 0 & 0 & 0 & 0 \\
\end{NiceArray}$
```

	a	b	c	d	e
x	0	0	0	0	0
y	0	0	0	0	0
z	0	0	0	0	0

5.3.2 The keys hvlines and hvlines-except-borders

The key `hvlines` (no value) is the conjunction of the keys `hlines` and `vlines`.

```
\setlength{\arrayrulewidth}{1pt}
\begin{NiceTabular}{cccc}[hvlines,rules/color=blue]
rose      & tulip & sunflower & lily \\
violet    & \Block[draw=red]{2-2}{\LARGE flowers} & & orchid \\
carnation & & & lys \\
daisy     & peony & jasmine & magnolia
\end{NiceTabular}
```

rose	tulip	sunflower	lily
violet	flowers		orchid
carnation			lys
daisy	peony	jasmine	magnolia

It's worth noting that, when the key `rounded-corners` is used for the environment `{NiceTabular}`, the key `hvlines` draws rounded corners for the exterior frame of the tabular: cf. part 14.1, p. 49.

The key `hvlines-except-borders` is similar to the key `hvlines` but does not draw the rules on the horizontal and vertical borders of the array. For an example of use of that key, see the part "Use with tcolorbox", p. 64.

5.3.3 The (empty) corners

The four `corners` of an array will be designed by `NW`, `SW`, `NE` and `SE` (*north west*, *south west*, *north east* and *south east*).
For each of these corners, we will call *empty corner* (or simply *corner*) the reunion of all the empty rectangles starting from the cell actually in the corner of the array.[21]
However, it's possible, for a cell without content, to require nicematrix to consider that cell as not empty with the command `\NotEmpty`.

In the example on the right (where B is in the center of a block of size 2×2), we have colored in blue the four (empty) corners of the array.

				A	
		A	A	A	
			A		
		A	A	A	A
A	A	A	A	A	A
A	A	A	A	A	A
	A	A	A		
	B		A		
			A		

When the key `corners`[22] is used, nicematrix computes the (empty) corners and these corners will be taken into account by the tools for drawing the rules (the rules won't be drawn in the corners).

[21] For sake of completeness, we should also say that a cell contained in a block (even an empty cell) is not taken into account for the determination of the corners. That behaviour is natural. The precise definition of a "non-empty cell" is given below (cf. p. 61).

[22] The key `corners` that we describe now has no direct link with the key `rounded-corners` described in the part 14.1, p. 49

```
\NiceMatrixOptions{cell-space-top-limit=3pt}
\begin{NiceTabular}{*{6}{c}}[corners,hvlines]
  &   &   &   & A \\
  &   & A & A & A \\
  &   &   & A \\
  &   & A & A & A & A \\
A & A & A & A & A & A \\
A & A & A & A & A & A \\
  & A & A & A \\
  & \Block[C]{2-2}{B} & & A \\
  &   &   & A \\
\end{NiceTabular}
```

				A	
		A	A	A	
			A		
		A	A	A	A
A	A	A	A	A	A
A	A	A	A	A	A
	A	A	A		
	B		A		
			A		

It's also possible to provide to the key `corners` a (comma-separated) list of corners (designed by `NW`, `SW`, `NE` and `SE`).

```
\NiceMatrixOptions{cell-space-top-limit=3pt}
\begin{NiceTabular}{*{6}{c}}[corners=NE,hvlines]
1\\
1&1\\
1&2&1\\
1&3&3&1\\
1&4&6&4&1\\
 & & & & &1
\end{NiceTabular}
```

1					
1	1				
1	2	1			
1	3	3	1		
1	4	6	4	1	
					1

▷ The corners are also taken into account by the tools provided by nicematrix to color cells, rows and columns. These tools don't color the cells which are in the corners (cf. p. 19). The command `\TikzEveryCell` available in the `\CodeBefore` and the `\CodeAfter` (cf. p. 44) takes also into account the empty corners.

5.3.4 The command \diagbox

The command `\diagbox` (inspired by the package diagbox), allows, when it is used in a cell, to slash that cell diagonally downwards.

```
$\begin{NiceArray}{*{5}{c}}[hvlines]
\diagbox{x}{y}  & e & a & b & c \\
e & e & a & b & c \\
a & a & e & c & b \\
b & b & c & e & a \\
c & c & b & a & e
\end{NiceArray}$
```

$x \backslash y$	e	a	b	c
e	e	a	b	c
a	a	e	c	b
b	b	c	e	a
c	c	b	a	e

It's possible to use the command `\diagbox` in a `\Block`.

```
$\begin{NiceArray}{*{5}{c}}[hvlines]
\Block{2-2}{\diagbox{x}{y}} &  & a & b & c \\
  &   & a & b & c \\
a & a & e & c & b \\
b & b & c & e & a \\
c & c & b & a & e
\end{NiceArray}$
```

$x \backslash y$		a	b	c
		a	b	c
a	a	e	c	b
b	b	c	e	a
c	c	b	a	e

But it's also possible to use `\diagbox` for the diagonal rule only (and the labels are put but the usual way, that is to say in cells of the tabular).

```
$\begin{NiceArray}{*{5}{c}}[hvlines]
\Block{2-2}{\diagbox{}{}} & y & a & b & c \\
x &   & a & b & c \\
a & a & e & c & b \\
b & b & c & e & a \\
c & c & b & a & e
\end{NiceArray}$
```

	y	a	b	c
x		a	b	c
a	a	e	c	b
b	b	c	e	a
c	c	b	a	e

Nevertheless, it's always possible to draw whatever rule we wish with TikZ in the `\CodeAfter` (or the `\CodeBefore`) by using the PGF/TikZ nodes created by nicematrix: cf. p. 54.

5.3.5 Commands for customized rules

It's also possible to define commands and letters for customized rules with the key `custom-line` available in `\NiceMatrixOptions` and in the options of individual environments. That key takes in as argument a list of *key=value* pairs. First, there is three keys to define the tools which will be used to use that new type of rule.

- the key `command` is the name (without the backslash) of a command that will be created by nicematrix and that will be available for the final user in order to draw horizontal rules (similarly to `\hline`);
- the key `ccommand` is the name (without the backslash) of a command that will be created by nicematrix and that will be available for the final user to order to draw partial horizontal rules (similarly to `\cline`, hence the name `ccommand`): the argument of that command is a list of intervals of columns specified by the syntax i or i-j.[23]
- the key `letter` takes in as argument a letter[24] that the user will use in the preamble of an environment with preamble (such as `{NiceTabular}` in order to specify a vertical rule.

We will now speak of the keys which describe the rule itself. Those keys may also be used in the (optional) argument of an individual command `\Hline` or in the (optional) argument of a specifier "`|`" in the preamble of an environment.

There is three possibilities.

- *First possibility*

  It's possible to specify composite rules, with a color and a color for the inter-rule space (as possible with colortbl for instance).

  - the key `multiplicity` is the number of consecutive rules that will be drawn: for instance, a value of 2 will create double rules such those created by `\hline\hline` or `||` in the preamble of an environment;
  - the key `color` sets the color of the rules ;
  - the key `sep-color` sets the color between two successive rules (should be used only in conjunction with `multiplicity`). The name of that key is inspired by the command `\doublerulesepcolor` of colortbl.

  That system may be used, in particular, for the definition of commands and letters to draw rules with a specific color (and those rules will respect the blocks and corners as do all the rules of nicematrix).

[23]It's recommended to use such commands only once in a row because each use will create space between the rows corresponding to the total width of the rule.
[24]The following letters are forbidden: `lcrpmbVX|()[]!@<>`

```
\begin{NiceTabular}{lcIcIc}[custom-line = {letter=I, color=blue}]
\hline
          & \Block{1-3}{dimensions} \\
          & L & l & h \\
\hline
Product A & 3 & 1 & 2 \\
Product B & 1 & 3 & 4 \\
Product C & 5 & 4 & 1 \\
\hline
\end{NiceTabular}
```

	dimensions		
	L	l	H
Product A	3	1	2
Product B	1	3	4
Product C	5	4	1

The key `sep-color` with the value `white` may also be used in case of an horizontal double-rule on the top of a colored cell (if we want the space between both rules above the cell not colored by the color of the cell).

```
\NiceMatrixOptions
  {
    custom-line =
     {
       command = DoubleRule ,
       multiplicity = 2 ,
       sep-color = white
     }
  }

\begin{NiceTabular}{ccc}
one & two & three \\
\DoubleRule
four & \cellcolor{yellow} five & six \\
\end{NiceTabular}
```

one	two	three
four	five	six

- *Second possibility*

  It's possible to use the key `tikz` (if TikZ is loaded). In that case, the rule is drawn directly with TikZ by using as parameters the value of the key `tikz` which must be a list of *key=value* pairs which may be applied to a TikZ path.

  By default, no space is reserved for the rule that will be drawn with TikZ. It is possible to specify a reservation (horizontal for a vertical rule and vertical for an horizontal one) with the key `total-width`. That value of that key, is, in some ways, the width of the rule that will be drawn (nicematrix does not compute that width from the characteristics of the rule specified in `tikz`).

  Here is an example with the key `dotted` of TikZ.

```
\NiceMatrixOptions
  {
    custom-line =
     {
       letter = I ,
       tikz = dotted ,
       total-width = \pgflinewidth
     }
  }

\begin{NiceTabular}{cIcIc}
one & two & three \\
four & five & six \\
seven & eight & nine
\end{NiceTabular}
```

one	two	three
four	five	six
seven	eight	nine

- *Third possibility* : the key `dotted`

  As one can see, the dots of a dotted line of TikZ have the shape of a square, and not a circle. That's why the extension nicematrix provides in the key `custom-line` a key `dotted` which will draw rounded dots. The initial value of the key `total-width` is, in this case, equal to the diameter of the dots (but the user may change the value with the key `total-width` if needed). Those dotted rules are also used by nicematrix to draw continuous dotted rules between cells of the matrix with `\Cdots`, `\Vdots`, etc. (cf. p. 31).

  In fact, nicematrix defines by default the commands `\hdottedline` and `\cdottedline` and the letter "`:`" for those dotted rules.[25]

```
\NiceMatrixOptions % present in nicematrix.sty
  {
    custom-line =
     {
       letter = : ,
       command = hdottedline ,
       ccommand = cdottedline ,
       dotted
     }
  }
```

  Thus, it's possible to use the commands `\hdottedline` and `\cdottedline` to draw horizontal dotted rules.

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

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

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

[25] However, it's possible to overwrite those definitions with a `custom-line` (in order, for example, to switch to dashed lines).

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

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

6 The color of the background of the rows and columns

6.1 Use of colortbl

We recall that the package colortbl can be loaded directly with `\usepackage{colortbl}` or by loading xcolor with the key `table`: `\usepackage[table]{xcolor}`.

However, there is two drawbacks:

- The package colortbl patches array, leading to some incompatibilities (for instance with the command `\hdotsfor`).
- The package colortbl constructs the array row by row, alterning colored rectangles, rules and contents of the cells. The resulting PDF is difficult to interpret by some PDF viewers and may lead to artefacts on the screen.
  - Some rules seem to disappear. This is because many PDF viewers give priority to graphical element drawn posteriorly (which is in the spirit of the "painting model" of PostScript and PDF). Concerning this problem, MuPDF (which is used, for instance, by SumatraPDF) gives better results than Adobe Reader.
  - A thin white line may appear between two cells of the same color. This phenomenon occurs when each cell is colored with its own instruction `fill` (the PostScript operator `fill` noted `f` in PDF). This is the case with colortbl: each cell is colored on its own, even when `\columncolor` or `\rowcolor` is used.

    As for this phenomenon, Adobe Reader gives better results than MuPDF (however, it seems, that, in recent versions of MuPDF, the problem has been solved...).

  The package nicematrix provides tools to avoid both problems.

6.2 The tools of nicematrix in the \CodeBefore

The package nicematrix provides some tools to draw the colored panels first, and, then, the content of the cells and the rules. This strategy is more conform to the "painting model" of the formats PostScript and PDF and is more suitable for the PDF viewers. However, it requires several compilations.[26]

The extension nicematrix provides a key `code-before` for some code that will be executed before the drawing of the tabular.

An alternative syntax is provided: it's possible to put the content of that `code-before` between the keywords `\CodeBefore` and `\Body` at the beginning of the environment.

```
\begin{pNiceArray}{preamble}
\CodeBefore [options]
  instructions of the code-before
\Body
  contents of the environment
\end{pNiceArray}
```

The optional argument between square brackets is a list of *key=value* pairs which will be presented progressively in this documentation.[27]

[26] If you use Overleaf, Overleaf will do automatically a sufficient number of compilations.

[27] The available keys are `create-cell-nodes`, `sub-matrix` (and its subkeys) and `delimiters-color`.

New commands are available in that \CodeBefore: \cellcolor, \rectanglecolor, \rowcolor, \columncolor, \rowcolors, \rowlistcolors, \chessboardcolors and \arraycolor.[28]
The names of these commands are inspired by the names of the commands provided by colortbl.

These commands don't color the cells which are in the "corners" if the key corners is used. That key has been described p. 14.

These commands respect the rounded corners if the key rounded-corners (described in the part 14.1 at the page 49) has been used.

All these commands accept an optional argument, between square brackets and in first position. That optional argument may contain two elements (separated by a comma)

- the colorimetric space (RGB, rgb, HTML, etc) as specified by the the extension xcolor;
- a specification of opacity of the form opacity = *value*.[29]

We describe now in detail those commands.

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

  This command takes in as mandatory arguments a color and a list of cells, each of which with the format i-j where i is the number of the row and j the number of the column of the cell. In fact, despite its name, this command may be used to color a whole row (with the syntax i-) or a whole column (with the syntax -j).

```
\begin{NiceTabular}{ccc}[hvlines]
\CodeBefore
  \cellcolor[HTML]{FFFF88}{3-1,2-2,-3}
\Body
a & b & c \\
e & f & g \\
h & i & j \\
\end{NiceTabular}
```

a	b	c
e	f	g
h	i	j

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

```
\begin{NiceTabular}{ccc}[hvlines]
\CodeBefore
  \rectanglecolor{blue!15}{2-2}{3-3}
\Body
a & b & c \\
e & f & g \\
h & i & j \\
\end{NiceTabular}
```

a	b	c
e	f	g
h	i	j

- The command \arraycolor takes in as mandatory argument a color and color the whole tabular with that color (except the potential exterior rows and columns: cf. p. 29). It's only a particular case of \rectanglecolor.

28Remark that, in the \CodeBefore, PGF/TikZ nodes of the form "(i-|j)" are also available to indicate the position to the potential rules: cf. p. 58.

29Caution: that feature creates instructions of transparency in the PDF and some readers of PDF don't support such instructions.

- The command `\chessboardcolors` takes in as mandatory arguments two colors and it colors the cells of the tabular in quincunx with these colors.

```
$\begin{pNiceMatrix}[r,margin]
\CodeBefore
  \chessboardcolors{red!15}{blue!15}
\Body
1 & -1 & 1 \\
-1 & 1 & -1 \\
1 & -1 & 1
\end{pNiceMatrix}$
```

$$\begin{pmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{pmatrix}$$

We have used the key `r` which aligns all the columns rightwards (cf. p. 51).

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

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

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

- The command `\columncolor` takes its name from the command `\columncolor` of colortbl. Its syntax is similar to the syntax of `\rowcolor`.

- The command `\rowcolors` (with a *s*) takes its name from the command `\rowcolors` of colortbl. The *s* emphasizes the fact that there is *two* colors. This command colors alternately the rows of the tabular with the two colors (provided in second and third argument), beginning with the row whose number is given in first (mandatory) argument. One of the arguments of color may be empty (no color is applied in the corresponding rows).

  In fact, the first (mandatory) argument is, more generally, a comma separated list of intervals describing the rows involved in the action of `\rowcolors` (an interval of the form *i*- describes in fact the interval of all the rows of the tabular, beginning with the row *i*).

  The last argument of `\rowcolors` is an optional list of pairs *key=value* (the optional argument in the first position corresponds to the colorimetric space). The available keys are `cols`, `restart` and `respect-blocks`.

  – The key `cols` describes a set of columns. The command `\rowcolors` will color only the cells of these columns. The value is a comma-separated list of intervals of the form *i*-*j* (where *i* or *j* may be replaced by `*`).

– With the key `restart`, each interval of rows (specified by the first mandatory argument) begins with the same color.[30]

– With the key `respect-blocks` the “rows” alternately colored may extend over several rows if they have to incorporate blocks (created with the command `\Block`: cf. p. 4).

```
\begin{NiceTabular}{clr}[hvlines]
\CodeBefore
  \rowcolors[gray]{2}{0.8}{}[cols=2-3,restart]
\Body
\Block{1-*}{Results} \\
John & 12 \\
Stephen & 8 \\
Sarah & 18 \\
Ashley & 20 \\
Henry & 14 \\
Madison & 15
\end{NiceTabular}
```

Results		
A	John	12
	Stephen	8
B	Sarah	18
	Ashley	20
	Henry	14
	Madison	15

```
\begin{NiceTabular}{lr}[hvlines]
\CodeBefore
  \rowcolors{1}{blue!10}{}[respect-blocks]
\Body
\Block{2-1}{John}     & 12 \\
                      & 13 \\
Steph                 &  8 \\
\Block{3-1}{Sarah}    & 18 \\
                      & 17 \\
                      & 15 \\
Ashley                & 20 \\
Henry                 & 14 \\
\Block{2-1}{Madison}  & 15 \\
                      & 19
\end{NiceTabular}
```

John	12
	13
Steph	8
Sarah	18
	17
	15
Ashley	20
Henry	14
Madison	15
	19

- The extension nicematrix provides also a command `\rowlistcolors`. This command generalises the command `\rowcolors`: instead of two successive arguments for the colors, this command takes in an argument which is a (comma-separated) list of colors. In that list, the symbol `=` represent a color identical to the previous one.

```
\begin{NiceTabular}{c}
\CodeBefore
  \rowlistcolors{1}{red!15,blue!15,green!15}
\Body
Peter \\
James \\
Abigail \\
Elisabeth \\
Claudius \\
Jane \\
Alexandra \\
\end{NiceTabular}
```

Peter
James
Abigail
Elisabeth
Claudius
Jane
Alexandra

[30] Otherwise, the color of a given row relies only upon the parity of its absolute number.

It's also possible to use in the command `\rowlistcolors` a color series defined by the command `\definecolorseries` of xcolor (and initialized with the command `\resetcolorseries`[31]).

```
\begin{NiceTabular}{c}
\CodeBefore
   \definecolorseries{BlueWhite}{rgb}{last}{blue}{white}
   \resetcolorseries[\value{iRow}]{BlueWhite}
   \rowlistcolors{1}{BlueWhite!!+}
\Body
Peter \\
James \\
Abigail \\
Elisabeth \\
Claudius \\
Jane \\
Alexandra \\
\end{NiceTabular}
```

Peter
James
Abigail
Elisabeth
Claudius
Jane
Alexandra

We recall that all the color commands we have described don't color the cells which are in the "corners". In the following example, we use the key `corners` to require the determination of the corner *north east* (NE).

```
\begin{NiceTabular}{ccccccc}[corners=NE,margin,hvlines,first-row,first-col]
\CodeBefore
  \rowlistcolors{1}{blue!15, }
\Body
  & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
0 & 1 \\
1 & 1 & 1 \\
2 & 1 & 2 & 1 \\
3 & 1 & 3 & 3 & 1 \\
4 & 1 & 4 & 6 & 4 & 1 \\
5 & 1 & 5 & 10 & 10 & 5 & 1 \\
6 & 1 & 6 & 15 & 20 & 15 & 6 & 1 \\
\end{NiceTabular}
```

	0	1	2	3	4	5	6
0	1						
1	1	1					
2	1	2	1				
3	1	3	3	1			
4	1	4	6	4	1		
5	1	5	10	10	5	1	
6	1	6	15	20	15	6	1

The previous example uses the keys `first-row` and `first-col` which are described in the chapter concerning the «exterior» rows and columns (cf. p. 29).

As one can see, *by default*, the coloring commands that we have described don't apply in those exterior rows and columns.

However, it may still be possible to color in those rows and columns by providing explicitly the numbers of those rows and columns.

In the following example, we require a color in the column 0 (which is the «first column» and which exists because the key `first-col` has been used).

[31] For the initialization, in the following example, you have used the counter `iRow` (which corresponds to the internal TeX counter `\c@iRow`) which, when used in the `\CodeBefore` (and in the `\CodeAfter`) corresponds to the number of rows of the array: cf. p 52. That leads to an adjustment of the gradation of the colors to the size of the tabular.

```
\begin{NiceTabular}{ccccccc}[corners=NE,margin,hvlines,first-row,first-col]
\CodeBefore
  \rowlistcolors{1}{blue!15, }
  \columncolor{red!15}{0}
\Body
  & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
0 & 1 \\
1 & 1 & 1 \\
2 & 1 & 2 & 1 \\
3 & 1 & 3 & 3 & 1 \\
4 & 1 & 4 & 6 & 4 & 1 \\
5 & 1 & 5 & 10 & 10 & 5 & 1 \\
6 & 1 & 6 & 15 & 20 & 15 & 6 & 1 \\
\end{NiceTabular}
```

	0	1	2	3	4	5	6
0	1						
1	1	1					
2	1	2	1				
3	1	3	3	1			
4	1	4	6	4	1		
5	1	5	10	10	5	1	
6	1	6	15	20	15	6	1

One should remark that all the previous commands are compatible with the commands of booktabs (`\toprule`, `\midrule`, `\bottomrule`, etc). However, booktabs is *not* loaded by nicematrix.

```
\begin{NiceTabular}[c]{lSSSS}
\CodeBefore
  \rowcolor{red!15}{1-2}
  \rowcolors{3}{blue!15}{}
\Body
\toprule
\Block{2-1}{Product} &
\Block{1-3}{dimensions (cm)} & & &
\Block{2-1}{\rotate Price} \\
\cmidrule(rl){2-4}
 & L & l & h \\
\midrule
small    & 3   & 5.5  & 1   & 30    \\
standard & 5.5 & 8    & 1.5 & 50.5  \\
premium  & 8.5 & 10.5 & 2   & 80    \\
extra    & 8.5 & 10   & 1.5 & 85.5  \\
special  & 12  & 12   & 0.5 & 70    \\
\bottomrule
\end{NiceTabular}
```

Product	dimensions (cm)			Price
	L	l	h	
small	3	5.5	1	30
standard	5.5	8	1.5	50.5
premium	8.5	10.5	2	80
extra	8.5	10	1.5	85.5
special	12	12	0.5	70

We have used the type of column `S` of siunitx (which should be loaded by the user).

New 7.1

It's also possible, in the `\CodeBefore`, to use the commands `\EmptyColumn` and `\EmptyRow`. The command `\EmptyColumn` takes in as argument a (comma-separated) list of numbers of columns and requires that no rules nor background colors will be drawn in the corresponding columns. The command `\EmptyRow` is similar for the rows.

```
\begin{NiceTabular}{ccccc}[hvlines,no-cell-nodes]
\CodeBefore
  \rowcolor{blue!15}{1}
  \EmptyColumn{3}
\Body
   one & two && three & four \\
   one & \Block{}{two\\ rows} && three & four \\
\end{NiceTabular}
```

one	two		three	four
one	two rows		three	four

6.3 Color tools to be used inside the tabular

The extension nicematrix also provides commands to be used directly in the tabular (as does colortbl).

These commands are the following several commands available (the first three ones are inspired by the similar commands of colortbl).

- `\cellcolor` which colorizes a cell;[32]
- `\rowcolor` which must be used in a cell and which colorizes the end of the row;[33]
- `\columncolor` which must be used in the preamble of the environment with the same syntax as the corresponding command of colortbl (however, unlike the command `\columncolor` of colortbl, this command `\columncolor` can appear within another command, itself used in the preamble of the array; on the other hand, the two final optional arguments for fixing offset are not supported);
- `\rowcolors` which takes in as arguments two colors and color the rest of the tabular with those colors;
- `\rowlistcolors` which takes in as argument a color and color the rest of the tabular with the colors of that list of colors.[34]

These commands are compatible with the commands for the overlays of Beamer (`\only`, etc.)

```
\NewDocumentCommand { \Blue } { } { \columncolor{blue!15} }
\begin{NiceTabular}{>{\Blue}c>{\Blue}cc}
\toprule
\rowcolor{red!15}
Last name & First name & Birth day \\
\midrule
Achard  & Jacques & 5 juin 1962 \\
Lefebvre & Mathilde & 23 mai 1988 \\
Vanesse & Stephany & 30 octobre 1994 \\
Dupont & Chantal & 15 janvier 1998 \\
\bottomrule
\end{NiceTabular}
```

Last name	First name	Birth day
Achard	Jacques	5 juin 1962
Lefebvre	Mathilde	23 mai 1988
Vanesse	Stephany	30 octobre 1994
Dupont	Chantal	15 janvier 1998

Each use of the `\rowlistcolors` (or `\rowcolors`, which is, in fact, a special case of `\rowlistcolors`) stops the potential coloring schemes[35] specified by a previous command `\rowlistcolors`. In particular, it's possible to start coloring the rows with `\rowlistcolors{...}` and stop coloring by a command `\rowlistcolors` with an empty argument.

[32]That command `\cellcolor` will delete the following spaces (which does not the command `\cellcolor` of colortbl). Moreover, if one wishes to define a command above that command `\cellcolor`, it must be protected in the TeX sens (whereas, if it were the command `\cellcolor` of colortbl, one should, no the contrary, write a *fully expandable* command).

[33]If you want a commande to color the following n rows, consider the command `\RowStyle` and its key `fillcolor`, p. 26

[34]When the command `\rowlistcolors` (or the command `\rowcolors`) is used in a cell of the column j of the array, the command applies only on the columns above j (by design).

[35]We say *schemes* in plural form because it's possible to start simultaneously several coloring schemes if they apply on different columns.

```
\begin{NiceTabular}{c}[hvlines]
one \\
two \\
\rowlistcolors{red!15}
three \\
four \\
five \\
\rowlistcolors{}
six \\
seven \\
\end{NiceTabular}
```

one
two
three
four
five
six
seven

6.4 The special color “nocolor”

The extension nicematrix provides the special color `nocolor` which may be used in all the coloring commands provided by nicematrix (in the `\CodeBefore` or in the array itself).
The cells marked by this color won’t be colored, whatever the other instructions of coloring which may apply to these cells.
Therefore, the color `nocolor` is an easy way to add an exception to a more general coloring command.

7 The command \RowStyle

The command `\RowStyle` takes in as argument some formatting instructions that will be applied to each cell on the rest of the current row.

That command also takes in as optional argument (between square brackets) a list of *key=value* pairs.

- The key `nb-rows` sets the number of rows to which the specifications of the current command will apply (with the special value `*`, it will apply to all the following rows).
- The keys `cell-space-top-limit`, `cell-space-bottom-limit` and `cell-space-limits` are available with the same meaning that the corresponding global keys (cf. p. 2).
- The key `fill` (alias: `rowcolor`) sets the color of the background and `opacity`[36] sets its opacity.
- If the key `rounded-corners` is used, that background will have rounded-corners.
- The key `color` sets the color of the text.[37]
- The key `bold` enforces bold characters for the cells of the row, both in math and text mode.

```
\begin{NiceTabular}{cccc}
\hline
\RowStyle[cell-space-limits=3pt]{\rotate}
first & second & third & fourth \\
\RowStyle[nb-rows=2,fill=blue!50,color=white]{\sffamily}
1 & 2 & 3 & 4 \\
I & II & III & IV
\end{NiceTabular}
```

first	second	third	fourth
1	2	3	4
I	II	III	IV

The command `\rotate` is described p. 51.

[36]Caution: that feature creates instructions of transparency in the PDF and some readers of PDF don’t support such instructions.
[37]The key `color` uses the command `\color` but also inserts an instruction `\leavevmode` before. This instruction prevents an extra vertical space in the cells which belong to columns of type `p`, `b`, `m` and `X` (which start in vertical mode of LaTeX). For the columns `V` (of varwidth), that’s not enough, except when LuaLaTeX is used with the package luacolor (see question 460489 on TeX StackExchange).

8 The width of the columns

8.1 Basic tools

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

```
\begin{NiceTabular}{W{c}{2cm}cc}[hvlines]
Paris  & New York & Madrid \\
Berlin & London   & Roma   \\
Rio    & Tokyo    & Oslo
\end{NiceTabular}
```

Paris	New York	Madrid
Berlin	London	Roma
Rio	Tokyo	Oslo

In the environments of `nicematrix`, it's also possible to fix the *minimal* width of all the columns (except the potential exterior columns: cf. p. 29) directly with the key `columns-width`.

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

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

Note that the space inserted between two columns (equal to 2 `\tabcolsep` in `{NiceTabular}` and to 2 `\arraycolsep` in the other environments) is not suppressed (of course, it's possible to suppress this space by setting `\tabcolsep` or `\arraycolsep` equal to 0 pt before the environment).

It's possible to give the special value `auto` to the option `columns-width`: all the columns of the array will have a width equal to the widest cell of the array.[38]

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

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

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

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

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

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`[39]. The environment `{NiceMatrixBlock}` has no direct link with the command `\Block` presented previously in this document (cf. p. 4).

[38] The result is achieved with only one compilation (but PGF/TikZ will have written information in the `aux` file and a message requiring a second compilation will appear).

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

```
\begin{NiceMatrixBlock}[auto-columns-width]
$\begin{array}{c}
\begin{bNiceMatrix}
 9 & 17 \\ -2 & 5
 \end{bNiceMatrix} \\ \\
\begin{bNiceMatrix}
 1   & 1245345 \\  345 & 2
\end{bNiceMatrix} \\
\end{array}$
\end{NiceMatrixBlock}
```

$$\begin{array}{c}\begin{bmatrix}9 & 17\\ -2 & 5\end{bmatrix}\\ \\ \begin{bmatrix}1 & 1245345\\ 345 & 2\end{bmatrix}\end{array}$$

8.2 The columns X

The environment `{NiceTabular}` provides `X` columns similar to those provided by the environment `{tabularx}` of the eponymous package.
The required width of the tabular may be specified with the key `width` (in `{NiceTabular}` or in `\NiceMatrixOptions`). The initial value of this parameter is `\linewidth` (and not `\textwidth`).
For sake of similarity with the environment `{tabularx}`, nicematrix also provides an environment `{NiceTabularX}` with a syntax similar to the syntax of `{tabularx}`, that is to say with a first mandatory argument which is the width of the tabular.
As with the packages tabu[40] and tabularray, the specifier `X` takes in an optional argument (between square brackets) which is a list of keys.

- It's possible to give a weight for the column by providing a positive number directly as argument of the specifier `X`. For example, a column `X[2]` will have a width double of the width of a column `X` (which has a weight equal to 1).[41]
- It's possible to specify an horizontal alignment with one of the letters `l`, `c` and `r` (which insert respectively `\raggedright`, `\centering` and `\raggedleft` followed by `\arraybackslash`).[42]
- It's possible to specify a vertical alignment with one of the keys `t` (alias `p`), `m` and `b` (which construct respectively columns of type `p`, `m` and `b`). The initial value is `t`.

```
\begin{NiceTabular}[width=9cm]{X[c,m]X[0.5,c,m]}[hvlines]
a rather long text which fits on several lines
& a rather long text which fits on several lines \\
a shorter text & a shorter text
\end{NiceTabular}
```

a rather long text which fits on several lines	a rather long text which fits on several lines
a shorter text	a shorter text

8.3 The columns V of varwidth

Let's recall first the behaviour of the environment `{varwidth}` of the eponymous package varwidth. That environment is similar to the classical environment `{minipage}` but the width provided in the argument is only the *maximal* width of the created box. In the general case, the width of the box constructed by an environment `{varwidth}` is the natural width of its contents.

That point is illustrated on the following examples.

[40] The extension tabu is now considered as deprecated.
[41] The negative values of the weight, as provided by tabu (which is now obsolete), are *not* supported by nicematrix. If such a value is used, an error will be raised.
[42] In fact, when ragged2e is loaded, nicematrix uses the commands `\RaggedRight`, `\Centering` and `\RaggedLeft` of ragged2e instead of the commands `\raggedright`, `\centering` and `\raggedleft`. That ensures a better output.

```
\fbox{%
\begin{varwidth}{8cm}
\begin{itemize}
\item first item
\item second item
\end{itemize}
\end{varwidth}}
```

- first item
- second item

```
\fbox{%
\begin{minipage}{8cm}
\begin{itemize}
\item first item
\item second item
\end{itemize}
\end{minipage}}
```

- first item
- second item

The package varwidth provides also the column type `V`. A column of type `V{`⟨*dim*⟩`}` encapsulates all its cells in a `{varwidth}` with the argument ⟨*dim*⟩ (and does also some tuning).

When the package varwidth is loaded, the columns `V` of varwidth are supported by nicematrix.

```
\begin{NiceTabular}[corners=NW,hvlines]{V{3cm}V{3cm}V{3cm}}
& some text & some very very very long text \\
some very very very long text \\
some very very very long text
\end{NiceTabular}
```

	some text	some very very very long text
some very very very long text		
some very very very long text		

Concerning nicematrix, one of the interests of this type of columns is that, for a cell of a column of type `V`, the PGF/TikZ node created by nicematrix for the content of that cell has a width adjusted to the content of the cell : cf. p. 56.

The columns `V` of nicematrix supports the keys `t`, `p`, `m`, `b`, `l`, `c` and `r` also supported by the columns `X`: see their description in the section 8.2, p. 28.

One should remark that the extension varwidth (at least in its version 0.92) has some problems: for instance, with LuaLaTeX, it does not work when the content begins with `\color`.

9 The exterior rows and columns

The options `first-row`, `last-row`, `first-col` and `last-col` allow the composition of exterior rows and columns in the environments of nicematrix. It's particularly interesting for the (mathematical) matrices.

A potential "first row" (exterior) has the number 0 (and not 1). Idem for the potential "first column".

```
$\begin{pNiceMatrix}[first-row,last-row,first-col,last-col,nullify-dots]
       & C_1    & \Cdots &        & C_4    &        \\
L_1    & a_{11} & a_{12} & a_{13} & a_{14} & L_1    \\
\Vdots & a_{21} & a_{22} & a_{23} & a_{24} & \Vdots \\
       & a_{31} & a_{32} & a_{33} & a_{34} &        \\
L_4    & a_{41} & a_{42} & a_{43} & a_{44} & L_4    \\
       & C_1    & \Cdots &        & C_4    &
\end{pNiceMatrix}$
```

$$\begin{array}{cccccc} & C_1 & \cdots & \cdots & C_4 & \\ L_1 & & & & & L_1 \\ \vdots & & \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} & & & \vdots \\ L_4 & & & & & L_4 \\ & C_1 & \cdots & \cdots & C_4 & \end{array}$$

The dotted lines have been drawn with the tools presented p. 31.

We have several remarks to do.

- For the environments with an explicit preamble (i.e. `{NiceTabular}`, `{NiceArray}` and its variants), no letter must be given in that preamble for the potential first column and the potential last column: they will automatically (and necessarily) be of type `r` for the first column and `l` for the last one.[43]
- One may wonder how nicematrix determines the number of rows and columns which are needed for the composition of the "last row" and "last column".
  - For the environments with explicit preamble, like `{NiceTabular}` and `{pNiceArray}`, the number of columns can obviously be computed from the preamble.
  - When the option `light-syntax` (cf. p. 53) is used, nicematrix has, in any case, to load the whole body of the environment (and that's why it's not possible to put verbatim material in the array with the option `light-syntax`). The analysis of this whole body gives the number of rows and the number of columns.
  - In the other cases, nicematrix compute the number of rows and columns during the first compilation and write the result in the `aux` file for the next run.

    However, it's possible to provide the number of the last row and the number of the last column as values of the options `last-row` *and* `last-col`, *tending to an acceleration of the whole compilation of the document.* That's what we will do throughout the rest of the document.

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

```
\NiceMatrixOptions{code-for-first-row = \color{red},
                   code-for-first-col = \color{blue},
                   code-for-last-row = \color{green},
                   code-for-last-col = \color{magenta}}
\begin{displaymath}
\begin{pNiceArray}{cc|cc}[first-row,last-row=5,first-col,last-col,nullify-dots]
       & C_1    & \multicolumn1c{\Cdots} &        & C_4    &        \\
L_1    & a_{11} & a_{12} & a_{13} & a_{14} & L_1    \\
\Vdots & a_{21} & a_{22} & a_{23} & a_{24} & \Vdots \\
```

[43]The users wishing exterior columns with another type of alignment should consider the command `\SubMatrix` available in the `\CodeAfter` and in the `\CodeBefore` (cf. p. 40) or try to put delimiter directly in the preamble of the array (cf. p. 38).

```
\hline
       & a_{31} & a_{32} & a_{33} & a_{34} &        \\
L_4    & a_{41} & a_{42} & a_{43} & a_{44} & L_4    \\
       & C_1    & \multicolumn1c{\Cdots} &        & C_4    &
\end{pNiceArray}
\end{displaymath}
```

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

Remarks

- As shown in the previous example, the horizontal and vertical rules don't extend in the exterior rows and columns. This remark also applies to the customized rules created by the key `custom-line` (cf. p. 16).
- A specification of color present in `code-for-first-row` also applies to a dotted line drawn in that exterior "first row" (except if a value has been given to `xdots/color`). Idem for the other exterior rows and columns.
- Logically, the potential option `columns-width` (described p. 27) doesn't apply to the "first column" and "last column".
- For technical reasons, it's not possible to use the option of the command `\\` after the "first row" or before the "last row". The placement of the delimiters would be wrong. If you are looking for a workaround, consider the command `\SubMatrix` in the `\CodeAfter` (and the `\CodeBefore`) described p. 40.

10 The continuous dotted lines

Inside the environments of the package nicematrix, new commands are defined: `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots` and `\Iddots`. These commands are intended to be used in place of `\dots`, `\cdots`, `\vdots`, `\ddots` and `\iddots`.[44]

Each of them must be used alone in the cell of the array and it draws a dotted line between the first non-empty cells[45] on both sides of the current cell. Of course, for `\Ldots` and `\Cdots`, it's an horizontal line; for `\Vdots`, it's a vertical line and for `\Ddots` and `\Iddots` diagonal ones. It's possible to change the color of these lines with the option `color`.[46]

```
\begin{bNiceMatrix}
a_1      & \Cdots &          & & a_1    \\
\Vdots   & a_2    & \Cdots & & a_2    \\
         & \Vdots & \Ddots[color=red] \\
\\
a_1      & a_2    &          & & a_n
\end{bNiceMatrix}
```

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

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

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

[45] The precise definition of a "non-empty cell" is given below (cf. p. 61).

[46] It's also possible to change the color of all these dotted lines with the option `xdots/color` (*xdots* to remind that it works for `\Cdots`, `\Ldots`, `\Vdots`, etc.): cf. p. 35.

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

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

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

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

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

In the first column of this example, there are two instructions `\Vdots` but, of course, only one dotted line is drawn.

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

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

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

There are also other means to change the size of the matrix. Someone might want to use the optional argument of the command `\\` for the vertical dimension and a command `\hspace*` in a cell for the horizontal dimension.[47]
However, a command `\hspace*` might interfere with the construction of the dotted lines. That's why the package nicematrix provides a command `\Hspace` which is a variant of `\hspace` transparent for the dotted lines of nicematrix.

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

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

10.1 The option nullify-dots

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

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

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

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

[47] In nicematrix, one should use `\hspace*` and not `\hspace` for such an usage because nicematrix loads array. One may also remark that it's possible to fix the width of a column by using the environment `{NiceArray}` (or one of its variants) with a column of type `w` or `W`: see p. 27

```
$B = \begin{pmatrix}
h & i & j & k & l & m \\
x & \ldots & \ldots  & \ldots & \ldots & x
\end{pmatrix}$
```

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

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

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

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

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 row. It's possible by using the option `nullify-dots` (and only one instruction `\Ldots` is necessary).

```
$D = \begin{pNiceMatrix}[nullify-dots]
h & i & j & k & l & m \\
x & \Ldots & &  &  & x
\end{pNiceMatrix}$
```

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

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

Caution: the key `nullify-dots` has a name that may be confusing; that key does not imply that the dotted rules won't be drawn!

10.2 The commands \Hdotsfor and \Vdotsfor

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.

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

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

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

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

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

Remark: Unlike the command `\hdotsfor` of amsmath, the command `\Hdotsfor` may be used even when the package colortbl[48] is loaded (however, with nicematrix, loading colortbl is rather point-less since nicematrix provides its own coloring tools: cf. p. 19).

[48]We recall that when xcolor is loaded with the option `table`, the package colortbl is loaded.

The package nicematrix also provides a command `\Vdotsfor` similar to `\Hdotsfor` but for the vertical dotted lines. The following example uses both `\Hdotsfor` and `\Vdotsfor`:

```
\begin{bNiceMatrix}
C[a_1,a_1] & \Cdots & C[a_1,a_n]
  & \hspace*{20mm} & C[a_1,a_1^{(p)}] & \Cdots & C[a_1,a_n^{(p)}] \\
\Vdots & \Ddots & \Vdots
  & \Hdotsfor{1} & \Vdots & \Ddots & \Vdots \\
C[a_n,a_1] & \Cdots & C[a_n,a_n]
  & & C[a_n,a_1^{(p)}] & \Cdots & C[a_n,a_n^{(p)}] \\
\rule{0pt}{15mm}\NotEmpty  & \Vdotsfor{1} & & \Ddots & & \Vdotsfor{1} \\
C[a_1^{(p)},a_1] & \Cdots & C[a_1^{(p)},a_n]
  & & C[a_1^{(p)},a_1^{(p)}] & \Cdots & C[a_1^{(p)},a_n^{(p)}] \\
\Vdots & \Ddots & \Vdots
  & \Hdotsfor{1} & \Vdots & \Ddots & \Vdots \\
C[a_n^{(p)},a_1] & \Cdots & C[a_n^{(p)},a_n]
  & & C[a_n^{(p)},a_1^{(p)}] & \Cdots & C[a_n^{(p)},a_n^{(p)}]
\end{bNiceMatrix}
```

$$\begin{bmatrix}
C[a_1,a_1] & \cdots & C[a_1,a_n] & & C[a_1,a_1^{(p)}] & \cdots & C[a_1,a_n^{(p)}] \\
\vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\
C[a_n,a_1] & \cdots & C[a_n,a_n] & & C[a_n,a_1^{(p)}] & \cdots & C[a_n,a_n^{(p)}] \\
& \vdots & & \ddots & & \vdots & \\
C[a_1^{(p)},a_1] & \cdots & C[a_1^{(p)},a_n] & & C[a_1^{(p)},a_1^{(p)}] & \cdots & C[a_1^{(p)},a_n^{(p)}] \\
\vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\
C[a_n^{(p)},a_1] & \cdots & C[a_n^{(p)},a_n] & & C[a_n^{(p)},a_1^{(p)}] & \cdots & C[a_n^{(p)},a_n^{(p)}]
\end{bmatrix}$$

10.3 How to generate the continuous dotted lines transparently

Imagine you have a document with a great number of mathematical matrices with ellipsis. You may wish to use the dotted lines of nicematrix without having to modify the code of each matrix. It's possible with the keys. `renew-dots` and `renew-matrix`.[49]

- The option `renew-dots`

  With this option, the commands `\ldots`, `\cdots`, `\vdots`, `\ddots`, `\iddots`[44] 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`.

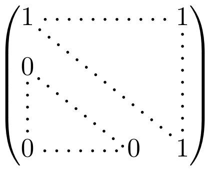
- The option `renew-matrix`

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

Therefore, with the keys `renew-dots` and `renew-matrix`, a classical code gives directly the output of nicematrix.

[49]The options `renew-dots`, `renew-matrix` can be fixed with the command `\NiceMatrixOptions` like the other options. However, they can also be fixed as options of the command `\usepackage`.

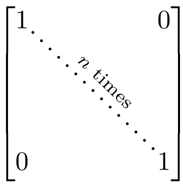
```
\NiceMatrixOptions{renew-dots,renew-matrix}
\begin{pmatrix}
1      & \cdots & \cdots & 1      \\
0      & \ddots &        & \vdots \\
\vdots & \ddots & \ddots & \vdots \\
0      & \cdots & 0      & 1
\end{pmatrix}
```



10.4 The labels of the dotted lines

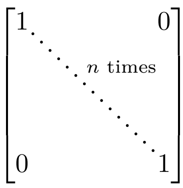
The commands `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots`, `\Hdotsfor` and `\Vdotsfor` (and the command `\line` in the `\CodeAfter` which is described p. 39) accept two optional arguments specified by the tokens _ and ^ for labels positioned below and above the line. The arguments are composed in math mode with `\scriptstyle`.
The version 6.22 of nicematrix has introduced a new label, specified by the token "`:`" for a label placed on the line. The label is composed on a white background which is put on the line previously drawn (see example on p. 68).

```
$\begin{bNiceMatrix}
1 & \hspace*{1cm}            & 0 \\[8mm]
  & \Ddots^{n \text{ times}} &   \\
0 &                          & 1
\end{bNiceMatrix}$
```



With the key `horizontal-label`, the labels stay horizontal.

```
$\begin{bNiceMatrix}
1 & \hspace*{1cm}            & 0 \\[8mm]
  & \Ddots[horizontal-label]^{n \text{ times}} &   \\
0 &                          & 1
\end{bNiceMatrix}$
```



10.5 Customisation of the dotted lines

The dotted lines drawn by `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots`, `\Hdotsfor` and `\Vdotsfor` (and by the command `\line` in the `\CodeAfter` which is described p. 39) may be customized by the following options (specified between square brackets after the command):

- `horizontal-label(s)`;
- `color`;
- `radius`;
- `shorten-start`, `shorten-end` and `shorten`;
- `inter`;
- `line-style`.

For an individual command, it's possible to write `horizontal-label` (in the singular).

These options may also be fixed with `\NiceMatrixOptions`, as options of `\CodeAfter` or at the level of a given environment but, in those cases, they must be prefixed by `xdots` (*xdots* to remind that it works for `\Cdots`, `\Ldots`, `\Vdots`, etc.), and, thus have for names:

- `xdots/horizontal-labels`;
- `xdots/color`;
- `xdots/radius`;

- `xdots/shorten-start`, `xdots/shorten-end` and `xdots/shorten`;
- `xdots/inter`;
- `xdots/line-style`.

For the clarity of the explanations, we will use those names.

With the key `xdots/horizontal-labels`, the labels (introduced by `_`, `^` and `:`) stay horizontal.

The option `xdots/color` fixes the color or the dotted line. It's possible to use a color definied "on the fly" (e.g.: `xdots/color = { [RGB]{204,204,255} }`). One should remark that the dotted lines drawn in the exterior rows and columns have a special treatment: cf. p. 29.

The option `radius` fixes the radius of the dots. The initial value is 0.53 pt.

The keys `xdots/shorten-start` and `xdots/shorten-end` fix the margin at the extremities of the line. The key `xdots/shorten` fixes both parameters. The initial value is 0.3 em (it is recommended to use a unit of length dependent of the current font).[50]

The option `xdots/inter` fixes the length between the dots. The initial value is 0.45 em (it is recommended to use a unit of length dependent of the current font).

The option `xdots/line-style`

It should be pointed that, by default, the lines drawn by TikZ with the parameter `dotted` are composed of square dots (and not rounded ones).[51]

```
\tikz \draw [dotted] (0,0) -- (5,0) ;
```

In order to provide lines with rounded dots in the style of those provided by `\ldots` (at least with the *Computer Modern* fonts), the package `nicematrix` embeds its own system to draw a dotted line (and this system uses PGF and not TikZ). This style is called `standard` and that's the initial value of the parameter `xdots/line-style`.
However (when TikZ is loaded) it's possible to use for `xdots/line-style` any style provided by TikZ, that is to say any sequence of options provided by TikZ for the TikZ paths (with the exception of "`color`", "`shorten >`" and "`shorten <`").

Here is for example a tridiagonal matrix with the style `loosely dotted`:

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

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

[50] In fact, when `xdots/shorten`, `xdots/shorten-start` and `xdots/shorten-end` are used in `\NiceMatrixOptions` or at the level of an environment (such as `{pNiceMatrix}`), those keys only apply to the extremeties of dotted lines corresponding to a non-empty content of a cell. When they are used for a command such as `\Cdots` (and, in that case, their names are `shorten`, `shorten-start` and `shorten-end`), they apply to all the extremities.

[51] The first reason of this behaviour is that the PDF format includes a description for dashed lines. The lines specified with this descriptor are displayed very efficiently by the PDF readers. It's easy, starting from these dashed lines, to create a line composed by square dots whereas a line of rounded dots needs a specification of each dot in the PDF file. Nevertheless, you can have a look at the following page to see how to have dotted rules with rounded dots in TikZ: `https://tex.stackexchange.com/questions/52848/tikz-line-with-large-dots`

10.6 The commands \Hbrace and \Vbrace

New 7.1

Since, as said in the previous part, it's possible to use, with the commands `\Cdots`, `\Ldots`, `\Vdots`, etc. all the styles of lines provided by TikZ, one may wish to use those commands to draw braces with the decoration `brace` provided by the library decorations.pathreplacing of TikZ.

In order to facilitate that use, nicematrix provides two commands `\Hbrace` and `\Vbrace`. Those commands are available only if TikZ has been loaded with its library decorations.pathreplacing (before or after the loading of nicematrix). Otherwise, a (non-fatal) error will be raised.

```
\usepackage{tikz}
\usetikzlibrary{decorations.pathreplacing}
```

The commands `\Hbrace` and `\Vbrace` have the same syntax. Both commands take in three arguments:

- an optional argument, between square brackets, which contains a list of *key=value* pairs: the keys allowed are `color`, `horizontal-labels`, `shorten`, `shorten-start` and `shorten-end`.
- a mandatory argument which is the number of columns (for `\Hbrace`) or the number of rows (for `\Vbrace`) over which the command applies;
- an mandatory argument which is the label of the curly brace.

As regards the ampersands (`&`), `\Hbrace` has the same behavior as the commands `\multicolumn`, `\hdotsfor`, `\Hdotsfor`, etc.: only one ampersand must be inserted, even if the brace that will be drawn will extend on several columns.

```
$\begin{NiceArray}{ccccc}%
  [ hvlines ,
    first-row ,
    last-row = 6,
    first-col ,
    last-col ,
    xdots/horizontal-labels ]
& \Hbrace{3}{p} & \Hbrace{2}{q} \\
\Vbrace{3}{p} & 1 & 1 & 134 & 1 & 1 & \Vbrace{3}{p} \\
& 1 & 1 & 134 & 1 & 1 \\
& 1 & 1 & 13456 & 1 & 1 \\
\Vbrace{2}{q} & 1 & 1 & 134 & 1 & 1 & \Vbrace{2}{q}\\
& 1 & 1 & 134 & 1 & 1 \\
& \Hbrace{3}{p} & \Hbrace[color=blue]{2}{q} \\
\end{NiceArray}$
```

$$\begin{array}{c|c|c|c|c|c|c}
 & \multicolumn{3}{c}{\overbrace{\hphantom{1\;1\;13456}}^{p}} & \multicolumn{2}{c}{\overbrace{\hphantom{1\;1}}^{q}} & \\
\hline
 & 1 & 1 & 134 & 1 & 1 & \\
\hline
p\left\{\right. & 1 & 1 & 134 & 1 & 1 & \left.\right\}p \\
\hline
 & 1 & 1 & 13456 & 1 & 1 & \\
\hline
q\left\{\right. & 1 & 1 & 134 & 1 & 1 & \left.\right\}q \\
\hline
 & 1 & 1 & 134 & 1 & 1 & \\
\hline
 & \multicolumn{3}{c}{\underbrace{\hphantom{1\;1\;13456}}_{p}} & \multicolumn{2}{c}{\color{blue}\underbrace{\hphantom{1\;1}}_{q}} &
\end{array}$$

The TikZ style used by nicematrix to draw those braces is called `nicematrix/brace`. Here is the initial value of that style:

```
decoration = { brace , raise = -0.15 em } ,
decorate
```

The final user can change that style TikZ as he wishes (respecting the structure with the keys `decorate` and `decoration`).

For another example of use of `\Hbrace` and `\Vbrace`, see the section "Dotted lines that are no longer dotted", p. 67.

10.7 The dotted lines and the rules

The dotted lines determine virtual blocks which have the same behaviour regarding the rules (the rules specified by the specifier `|` in the preamble, by the command `\Hline`, by the keys `hlines`, vlines, hvlines and hvlines-except-borders and by the tools created by custom-line are not drawn within the blocks).[52]

```
$\begin{bNiceMatrix}[margin,hvlines]
\Block{3-3}<\LARGE>{A} & & & 0 \\
& \hspace*{1cm} & & \Vdots \\
& & & 0 \\
0 & \Cdots& 0 & 0
\end{bNiceMatrix}$
```

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

11 Delimiters in the preamble of the environment

In the environments with preamble ({NiceArray}, {pNiceArray}, etc.), it's possible to put vertical delimiters directly in the preamble of the environment.[53]

The opening delimiters should be prefixed by the keyword \left and the closing delimiters by the keyword \right. It's not mandatory to use \left and \right pair-wise.

All the vertical extensible delimiters of LaTeX are allowed.

Here is a example which uses the delimiters \lgroup and \rgroup.

```
$\begin{NiceArray}{\left\lgroup ccc\right\rgroup l}
1 & 2 & 3 &
4 & 1 & 6 &
7 & 8 & 9 & \scriptstyle L_3 \gets L_3 + L_1 + L_2
\end{NiceArray}$
```

$$\begin{array}{l} \left\lgroup\begin{array}{ccc} 1 & 2 & 3 \\ 4 & 1 & 6 \\ 7 & 8 & 9 \end{array}\right\rgroup_{\scriptstyle L_3 \gets L_3 + L_1 + L_2} \end{array}$$

For this example, it would also have been possible to use the environment {NiceArrayWithDelims} (cf. the section 14.10, p. 54) and the key last-col (cf. p. 29).

There is a particular case: for the delimiters (, [and \{[54], and the corresponding closing delimiters, the prefixes \left et \right are optional.[55]

Here is an example with a left delimiter \{ in a {NiceTabular} (remark the compatibility with the key t).

```
We define $f$ with\quad
\begin{NiceTabular}[t]{\{ll}
$f(x) = 0$ & if $x$ is non positive \\
$f(x) = 1-e^x$ & if $x$ is positive
```

On définit f par $\begin{cases} f(x) = 0 & \text{if } x \text{ is non positive} \\ f(x) = 1 - e^x & \text{if } x \text{ is positive} \end{cases}$

When there are two successive delimiters (necessarily a closing one following by an opening one for another submatrix), a space equal to \enskip is automatically inserted.

[52]On the other side, the command \line in the \CodeAfter (cf. p. 39) does *not* create block.

[53]This syntax is inspired by the extension blkarray.

[54]For the braces, the protection by a backslash is mandatory (that's why we have written \{).

[55]For the delimiters [and], the prefixes remain mandatory when there is a conflict of notation with the square brackets for the options of some descriptors of columns.

```
$\begin{pNiceArray}{(c)(c)(c)}
a_{11} & a_{12}                                  & a_{13} \\
a_{21} & \displaystyle \int_0^1\dfrac{1}{x^2+1}\,dx & a_{23} \\
a_{31} & a_{32}                                  & a_{33}
\end{pNiceArray}$
```

$$\left(\begin{pmatrix} a_{11} \\ a_{21} \\ a_{31} \end{pmatrix}\begin{pmatrix} a_{12} \\ \displaystyle \int_0^1\dfrac{1}{x^2+1}\,dx \\ a_{32} \end{pmatrix}\begin{pmatrix} a_{13} \\ a_{23} \\ a_{33} \end{pmatrix}\right)$$

For more complex constructions, in particular with delimiters spanning only a *subset* of the rows of the array, one should consider the command `\SubMatrix` available in the `\CodeAfter` (and the `\CodeBefore`). See the section 12.2, p. 40.

12 The \CodeAfter

The option `code-after` may be used to give some code that will be executed *after* the construction of the matrix.[56]

For the legibility of the code, an alternative syntax is provided: it's possible to give the instructions of the `code-after` at the end of the environment, after the keyword `\CodeAfter` (the key `code-after` is of course mandatory in the `\AutoNiceMatrix` and the similar commands). Although `\CodeAfter` is a keyword, it takes in an optional argument (between square brackets).[57]

The experienced users may, for instance, use the PGF/TikZ nodes created by nicematrix in the `\CodeAfter`. These nodes are described further beginning on p. 54.

The `\CodeAfter` provides several special commands: `\line`, `\SubMatrix`, `\OverBrace`, `\UnderBrace` and `\TikzEveryCell`. We will now present these commands.

One should avoid spurious spaces in that `\CodeAfter`.[58]

12.1 The command \line in the \CodeAfter

The command `\line` draws directly dotted lines between cells or blocks. It takes in two arguments for the cells or blocks to link. Both argument may be:

- a specification of cell of the form *i*-*j* where is the number of the row and *j* is the number of the column;
- the name of a block (created by the command `\Block` with the key `name` of that command).

The options available for the customisation of the dotted lines created by `\Cdots`, `\Vdots`, etc. are also available for this command (cf. p. 35).

This command may be used, for example, to draw a dotted line between two adjacent cells.

[56]There is also a key `code-before` described p. 19.
[57]Here are the keys accepted in that argument: `delimiters/color`, `rules` and its sub-keys, `sub-matrix` (linked to the command `\SubMatrix`) and its sub-keys and `xdots` (for the command `\line`) and its sub-keys.
[58]See: `https://tex.stackexchange.com/questions/689336`

```
\NiceMatrixOptions{xdots/shorten = 0.6 em}
\begin{pNiceMatrix}
I       & 0       & \Cdots  &0     \\
0       & I       & \Ddots  &\Vdots\\
\Vdots  &\Ddots   & I       &0     \\
0       &\Cdots   & 0       &I
\CodeAfter \line{2-2}{3-3}
\end{pNiceMatrix}
```

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

It can also be used to draw a diagonal line not parallel to the other diagonal lines (by default, the dotted lines drawn by `\Ddots` are "parallelized": cf. p. 60).

```
\begin{bNiceMatrix}
1      & \Cdots &   & 1      & 2      & \Cdots          & 2      \\
0      & \Ddots &   & \Vdots & \Vdots & \hspace*{2.5cm} & \Vdots \\
\Vdots & \Ddots &   &        &        &                 &        \\
0      & \Cdots & 0 & 1      & 2      & \Cdots          & 2
\CodeAfter \line[shorten=6pt]{1-5}{4-7}
\end{bNiceMatrix}
```

$$\begin{bmatrix} 1 & \cdots & & 1 & 2 & \cdots & 2 \\ 0 & \ddots & & \vdots & \vdots & \ddots & \vdots \\ \vdots & \ddots & & & & & \\ 0 & \cdots & 0 & 1 & 2 & \cdots & 2 \end{bmatrix}$$

12.2 The command \SubMatrix in the \CodeAfter (and the \CodeBefore)

The command `\SubMatrix` provides a way to put delimiters on a portion of the array considered as a submatrix. The command `\SubMatrix` takes in five arguments:

- the first argument is the left delimiter, which may be any extensible delimiter provided by LaTeX : `(`, `[`, `\{`, `\langle`, `\lgroup`, `\lfloor`, etc. but also the null delimiter `.`;
- the second argument is the upper-left corner of the submatrix with the syntax i-j where i the number of row and j the number of column (the special word `last` is allowed);
- the third argument is the lower-right corner with the same syntax;
- the fourth argument is the right delimiter;
- the last argument, which is optional, is a list of *key=value* pairs.[59]

One should remark that the command `\SubMatrix` draws the delimiters *after* the construction of the array: no space is inserted by the command `\SubMatrix` itself. That's why, in the following example, we have used the key `margin` and you have added by hand some space between the third and fourth column with `@{\hspace{1.5em}}` in the preamble of the array.

```
\[\begin{NiceArray}{ccc@{\hspace{1.5em}}c}[cell-space-limits=2pt,margin]
 1           & 1           & 1           & x \\
\dfrac{1}{4} & \dfrac{1}{2} & \dfrac{1}{4} & y \\
 1           & 2           & 3           & z
\CodeAfter
  \SubMatrix({1-1}{3-3})
  \SubMatrix({1-4}{3-4})
\end{NiceArray}\]
```

$$\begin{pmatrix} 1 & 1 & 1 \\ \dfrac{1}{4} & \dfrac{1}{2} & \dfrac{1}{4} \\ 1 & 2 & 3 \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

[59]There is no optional argument between square brackets in first position because a square bracket just after `\SubMatrix` must be interpreted as the first (mandatory) argument of the command `\SubMatrix`: that bracket is the left delimiter of the sub-matrix to construct (eg.: `\SubMatrix[{2-2}{4-7}]`).

Eventually, in this example, it would probably have been easier to put the delimiters directly in the preamble of `{NiceArray}` (see section 11, p. 38) with the following construction.

```
$\begin{NiceArray}{(ccc)(c)}[cell-space-limits=2pt]
 1          & 1           & 1           & x \\
\dfrac{1}{4} & \dfrac{1}{2} & \dfrac{1}{4} & y \\
 1          & 2           & 3           & z
\end{NiceArray}$
```

$$\begin{pmatrix} 1 & 1 & 1 \\ \dfrac{1}{4} & \dfrac{1}{2} & \dfrac{1}{4} \\ 1 & 2 & 3 \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

In fact, the command `\SubMatrix` also takes in two optional arguments specified by the traditional symbols `^` and `_` for material in superscript and subscript (but no space is reserved for that material).

```
$\begin{bNiceMatrix}[right-margin=1em]
1 & 1 & 1 \\
1 & a & b \\
1 & c & d
\CodeAfter
  \SubMatrix[{2-2}{3-3}]^{T}
\end{bNiceMatrix}$
```

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & \left[\begin{matrix} a \\ c \end{matrix}\right. & \left.\begin{matrix} b \\ d \end{matrix}\right]^T \end{bmatrix}$$

The options of the command `\SubMatrix` are as follows:

- `left-xshift` and `right-xshift` shift horizontally the delimiters (there exists also the key `xshift` which fixes both parameters);
- `extra-height` adds a quantity to the total height of the delimiters (height `\ht` + depth `\dp`);
- `delimiters/color` fixes the color of the delimiters (also available in `\NiceMatrixOptions`, in the environments with delimiters and as option of the keyword `\CodeAfter`);
- `slim` is a boolean key: when that key is in force, the horizontal position of the delimiters is computed by using only the contents of the cells of the submatrix whereas, in the general case, the position is computed by taking into account the cells of the whole columns implied in the submatrix (see example below). ;
- `vlines` contents a list of numbers of vertical rules that will be drawn in the sub-matrix (if this key is used without value, all the vertical rules of the sub-matrix are drawn);
- `hlines` is similar to `vlines` but for the horizontal rules;
- `hvlines`, which must be used without value, draws all the vertical and horizontal rules;
- `code` insert code, especially TikZ code, after the construction of the submatrix. That key is detailed below.

One should remark that the keys add their rules after the construction of the main matrix: no space is added between the rows and the columns of the array for these rules.

All these keys are also available in `\NiceMatrixOptions`, at the level of the environments of nicematrix or as option of the command `\CodeAfter` with the prefix `sub-matrix` which means that their names are therefore `sub-matrix/left-xshift`, `sub-matrix/right-xshift`, `sub-matrix/xshift`, etc.

```
$\begin{NiceArray}{cc@{\hspace{5mm}}l}[cell-space-limits=2pt]
  &   & \frac{1}{2} \\
  &   & \frac{1}{4} \\[1mm]
a & b & \frac{1}{2}a+\frac{1}{4}b \\
c & d & \frac{1}{2}c+\frac{1}{4}d \\
\CodeAfter
  \SubMatrix({1-3}{2-3})
  \SubMatrix({3-1}{4-2})
  \SubMatrix({3-3}{4-3})
\end{NiceArray}$
```

$$\begin{array}{ccl} & & \begin{pmatrix} \frac{1}{2} & \\ \frac{1}{4} & \end{pmatrix} \\ \begin{pmatrix} a & b \\ c & d \end{pmatrix} & & \begin{pmatrix} \frac{1}{2}a+\frac{1}{4}b \\ \frac{1}{2}c+\frac{1}{4}d \end{pmatrix} \end{array}$$

Here is the same example with the key `slim` used for one of the submatrices.

```
$\begin{NiceArray}{cc@{\hspace{5mm}}l}[cell-space-limits=2pt]
  &   & \frac12 \\
  &   & \frac14 \\[1mm]
a & b & \frac12a+\frac14b \\
c & d & \frac12c+\frac14d \\
\CodeAfter
  \SubMatrix({1-3}{2-3})[slim]
  \SubMatrix({3-1}{4-2})
  \SubMatrix({3-3}{4-3})
\end{NiceArray}$
```

$$\begin{array}{cc} & \begin{pmatrix} \frac12 \\ \frac14 \end{pmatrix} \\ \begin{pmatrix} a & b \\ c & d \end{pmatrix} & \begin{pmatrix} \frac12 a + \frac14 b \\ \frac12 c + \frac14 d \end{pmatrix} \end{array}$$

There is also a key `name` which gives a name to the submatrix created by `\SubMatrix`. That name is used to create PGF/TikZ nodes: cf p. 59.

Despite its name, the command `\SubMatrix` may also be used within a `{NiceTabular}`. Here is an example (which uses `\bottomrule` and `\toprule` of `booktabs`).

```
\begin{NiceTabular}{@{}ll@{}}
\toprule
Part A               & the first part \\
\Block{2-1}{Part B} & a first sub-part \\
                     & a second sub-part \\
\bottomrule
\CodeAfter
  \SubMatrix{\lbrace}{2-2}{3-2}{.}
\end{NiceTabular}
```

Part A	the first part
Part B	{ a first sub-part
	a second sub-part

The command `\SubMatrix` is, in fact, also available in the `\CodeBefore`. By using `\SubMatrix` in the `\CodeBefore`, the delimiters drawn by those commands `\SubMatrix` are taken into account to limit the continuous dotted lines (drawn by `\Cdots`, `\Vdots`, etc.) which have an open extremity. For an example, see voir 18.9 p. 75.

Caution : The following functionality is fragile and does not work with `latex–dvips–ps2pdf`.

The key `code` of the command `\SubMatrix` allows the insertion of code after the construction of the submatrix. It's meant to be used to insert TikZ instructions because, in the TikZ instructions inserted by that code, the nodes of the form `i-j.`*`anchor`* or `i-|j` are interpreted with `i` and `j` as numbers of row and column *relative to the submatrix*.[60]

```
$\begin{NiceArray}{ccc@{}w{c}{5mm}@{}ccc}
    &   &   && -1 & 1  & 2  \\
    &   &   &&  0 & 3  & 4  \\
    &   &   &&  0 & 0  & 5  \\
  1 & 2 & 3 && -1 & 7  & 25 \\
  0 & 4 & 5 && 0  & 12 & 41 \\
  0 & 0 & 6 && 0  & 0  & 30
\CodeAfter
  \NewDocumentCommand{\MyDraw}{}{\tikz \draw [blue] (2-|1) -| (3-|2) -|
      (4-|3) ;}
  \SubMatrix({1-5}{3-7})[code = \MyDraw ]
  \SubMatrix({4-1}{6-3})[code = \MyDraw ]
  \SubMatrix({4-5}{6-7})[code = \MyDraw ]
\end{NiceArray}$
```

[60] Be careful: the syntax `j|-i` is *not* allowed.

$$\begin{array}{cc} & \begin{pmatrix} -1 & 1 & 2 \\ 0 & 3 & 4 \\ 0 & 0 & 5 \end{pmatrix} \\ \begin{pmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{pmatrix} & \begin{pmatrix} -1 & 7 & 25 \\ 0 & 12 & 41 \\ 0 & 0 & 30 \end{pmatrix} \end{array}$$

As we see, the drawing done by our command `\MyDraw` is *relative* to the submatrix to which it is applied.

12.3 The commands \OverBrace and \UnderBrace in the \CodeAfter

The commands `\OverBrace` and `\UnderBrace` provide a way to put horizontal braces on a part of the array. These commands take in three arguments:

- the first argument is the upper-left corner of the submatrix with the syntax i-j where i the number of row and j the number of column;
- the second argument is the lower-right corner with the same syntax;
- the third argument is the label of the brace that will be put by nicematrix (with PGF) above the brace (for the command `\OverBrace`) or under the brace (for `\UnderBrace`).

  It's possible to use the command `\\` in this third argument in order to format the label on several lines.

```
\begin{pNiceMatrix}
1  & 2  & 3  & 4  & 5  & 6  \\
11 & 12 & 13 & 14 & 15 & 16 \\
\CodeAfter
  \OverBrace{1-1}{2-3}{A}
  \OverBrace{1-4}{2-6}{B}
\end{pNiceMatrix}
```

$$\begin{pmatrix} \overbrace{\begin{matrix} 1 & 2 & 3 \\ 11 & 12 & 13 \end{matrix}}^{A} & \overbrace{\begin{matrix} 4 & 5 & 6 \\ 14 & 15 & 16 \end{matrix}}^{B} \end{pmatrix}$$

Caution: There is no vertical space reserved for those braces and their labels.[61]

In fact, the commands `\OverBrace` and `\UnderBrace` take in an optional argument (in first position and between square brackets) for a list of *key=value* pairs. The available keys are:

- `left-shorten` and `right-shorten` which do not take in value; when the key `left-shorten` is used, the abscissa of the left extremity of the brace is computed with the contents of the cells of the involved sub-array, otherwise, the position of the potential vertical rule is used (idem for `right-shorten`).
- `shorten`, which is the conjunction of the keys `left-shorten` and `right-shorten`;
- `yshift`, which shifts vertically the brace (and its label);
- `color`, which sets the color of the brace (and its label).

```
\begin{pNiceMatrix}
1  & 2  & 3  & 4  & 5  & 6  \\
11 & 12 & 13 & 14 & 15 & 16 \\
\CodeAfter
  \OverBrace[shorten,yshift=3pt]{1-1}{2-3}{A}
  \OverBrace[shorten,yshift=3pt]{1-4}{2-6}{B}
\end{pNiceMatrix}
```

$$\begin{pmatrix} \overbrace{\begin{matrix} 1 & 2 & 3 \\ 11 & 12 & 13 \end{matrix}}^{A} & \overbrace{\begin{matrix} 4 & 5 & 6 \\ 14 & 15 & 16 \end{matrix}}^{B} \end{pmatrix}$$

[61] See: https://tex.stackexchange.com/questions/685755

12.4 The command \TikzEveryCell in the \CodeAfter

The command `\TikzEveryCell` executes with TikZ the rectangular path corresponding to each cell of the tabular with parameters of TikZ the argument of `\TikzEveryCell`. That argument must be a list of *key=value* pairs which may be applied to a TikZ path. In fact, the command applies to each of the tabular, except those in the exterior rows and columns (cf. p. 29) and those in the empty corners (when the key `corners` is used: cf. p. 14). It applies in fact to each block (excepted those with the key `transparent`) and does not apply to the individual cells located within these blocks.

In fact, in the list of keys provided as argument of `\TikzEveryCell`, it's possible to put a key `offset`. That key is not provided by TikZ but by nicematrix. It will narrow the rectangular frame corresponding to the block by a margin (horizontally and vertically) equal to the value (of that key `offset`). That new frame, a bit narrower, will be executed by TikZ with options which are the other keys in the argument of `\TikzEveryCell`.

```
\renewcommand{\arraystretch}{1.3}
\begin{NiceTabular}{ccc}[corners]
  & \Block{1-2}{columns} \\
  \Block{2-1}{rows}
  & cell 1 1 & cell 1 2 \\
  & cell 2 1 & cell 2 2
\CodeAfter
  \TikzEveryCell{offset=1pt,draw}
\end{NiceTabular}
```

	columns	
rows	cell 1 1	cell 1 2
	cell 2 1	cell 2 2

The command `\TikzEveryCell` has two optional keys, available between square brackets.

- with the key `empty`, the command only acts on the empty cells (the exact definition of an "empty cell" is given in part 17.2, p. 61);
- with key `non-empty`, the command only acts on the non-empty cells.

```
\renewcommand{\arraystretch}{1.4}
\begin{NiceTabular}{cccccc}[hvlines]
  P & O & U & R & V & U \\
  O &   &   & E & I &   \\
  M & O & R & F & A & L \\
  E & T & A & L &   & E \\
  L & A & S & E & R & S \\
  O &   & E & X & I & T
\CodeAfter
  \TikzEveryCell[empty]{fill=gray,draw}
\end{NiceTabular}
```

P	O	U	R	V	U
O			E	I	
M	O	R	F	A	L
E	T	A	L		E
L	A	S	E	R	S
O		E	X	I	T

The command `\TikzEveryCell` is, in fact, also available in the `\CodeBefore`.

13 Captions and notes in the tabulars

13.1 Caption of a tabular

The environment `{NiceTabular}` provides the keys `caption`, `short-caption` and `label` which may be used when the tabular is inserted in a floating environment (typically the environment `{table}`).

With the key `caption`, the caption, when it is long, is wrapped at the width of the tabular (except the potential exterior columns specified by `first-col` and `last-col`: cf. 9, p. 29), without the use of the package threeparttable or the package floatrow.

By default, the caption is composed below the tabular. With the key `caption-above`, available in `\NiceMatrixOptions`, the caption will be composed above the tabular.

The key `short-caption` corresponds to the optional argument of the classical command `\caption` and the key `label` corresponds, of course, to the command `\label`.

See table 1, p. 47, for an example of use the keys `caption` and `label`.

These functionalities are compatible with the extension caption.

13.2 The footnotes

The package nicematrix allows, by using footnote or footnotehyper, the extraction of the notes inserted by `\footnote` in the environments of nicematrix and their composition in the foot of the page with the other notes of the document.

If nicematrix is loaded with the option `footnote` (with `\usepackage[footnote]{nicematrix}` or with `\PassOptionsToPackage`), the package footnote is loaded (if it is not yet loaded) and it is used to extract the footnotes.

If nicematrix is loaded with the option `footnotehyper`, the package footnotehyper is loaded (if it is not yet loaded) ant it is used to extract footnotes.

Caution: The packages footnote and footnotehyper are incompatible. The package footnotehyper is the successor of the package footnote and should be used preferently. The package footnote has some drawbacks, in particular: it must be loaded after the package xcolor and it is not perfectly compatible with hyperref.

13.3 The notes of tabular

The package nicematrix also provides a command `\tabularnote` which gives the ability to specify notes that will be composed at the end of the array with a width of text equal to the width of the array (except the potential exterior columns specified by `first-col` and `last-col`: cf. 9, p. 29). With no surprise, that command is available only in the environments `{NiceTabular}`, `{NiceTabular*}` and `{NiceTabularX}`.
In fact, this command is available only if the extension enumitem has been loaded (before or after nicematrix). Indeed, the notes are composed at the end of the array with a type of list provided by the package enumitem.

```
\begin{NiceTabular}{@{}llr@{}}
\toprule \RowStyle{\bfseries}
Last name & First name & Birth day \\
\midrule
Achard\tabularnote{Achard is an old family of the Poitou.}
& Jacques & June 5, 2005 \\
Lefebvre\tabularnote{The name Lefebvre is an alteration of the name Lefebure.}
& Mathilde & January 23, 1975 \\
Vanesse & Stephany & October 30, 1994 \\
Dupont & Chantal & January 15, 1998 \\
\bottomrule
\end{NiceTabular}
```

Last name	**First name**	**Birth day**
Achard[a]	Jacques	June 5, 2005
Lefebvre[b]	Mathilde	January 23, 1975
Vanesse	Stephany	October 30, 1994
Dupont	Chantal	January 15, 1998

[a] Achard is an old family of the Poitou.
[b] The name Lefebvre is an alteration of the name Lefebure.

- If you have several successive commands `\tabularnote{...}` *with no space at all between them*, the labels of the corresponding notes are composed together, separated by commas (this is similar to the option `multiple` of footmisc for the footnotes).
- If a command `\tabularnote{...}` is exactly at the end of a cell (with no space at all after) and if the alignment mode of the column is `c` or `r`, the label of the note is composed in an overlapping position (towards the right). This structure may provide a better alignment of the cells of a given column.
- If the key `notes/para` is used, the notes are composed at the end of the array in a single paragraph (as with the key `para` of threeparttable).
- There is a key `tabularnote` which provides a way to insert some text in the zone of the notes before the numbered tabular notes.

  An alternative syntax is available with the environment `{TabularNote}`. That environment should be used at the end of the environment `{NiceTabular}` (but *before* a potential instruction `\CodeAfter`).
- If the package booktabs has been loaded (before or after nicematrix), the key `notes/bottomrule` draws a `\bottomrule` of booktabs *after* the notes.
- The command `\tabularnote` may be used *before* the environment of nicematrix. Thus, it's possible to use it on the title inserted by `\caption` in an environment `{table}` of LaTeX (or in a command `\captionof` of the package caption). It's also possible, as expected, to use the command `\tabularnote` in the caption provided by the *key* `caption` of the environment `{NiceTabaular}`.

  If several commands `\tabularnote` are used in a tabular with the same argument, only one note is inserted at the end of the tabular (but all the labels are composed, of course). It's possible to control that feature with the key `notes/detect-duplicates`.[62]
- It's possible to create a reference to a tabular note created by `\tabularnote` (with the usual command `\label` used after the `\tabularnote`).
- The command `\tabularnote` has an optional argument (between square brackets) to change the symbol of the reference of the note.

  Example: `\tabularnote[$\star$]{A footnote...}`

For an illustration of some of those remarks, see table 1, p. 47. This table has been composed with the following code (the package caption has been loaded in this document).

```
\begin{table}
\centering
\NiceMatrixOptions{caption-above}
\begin{NiceTabular}{@{}llc@{}
  [
    caption = A tabular whose caption has been specified by the key
      \texttt{caption}\tabularnote[$\star$]{It's possible to put a tabular
      note in the caption.} ,
    label = t:tabularnote ,
    tabularnote = Some text before the notes. ,
    notes/bottomrule
  ]
\toprule
Last name & First name & Length of life \\
\midrule
Churchill & Wiston &  91\\
Nightingale\tabularnote{Considered as the first nurse of history}%
```

[62]For technical reasons, the final user is not allowed to put several commands `\tabularnote` with exactly the same argument in the caption of the tabular. It's not a real restriction...

```
\tabularnote{Nicknamed ``the Lady with the Lamp''.}
& Florence\tabularnote{This note is shared by two references.} & 90 \\
Schoelcher & Victor & 89\tabularnote{The label of the note is overlapping.}\\
Touchet & Marie\tabularnote{This note is shared by two references.} & 89 \\
Wallis & John & 87 \\
\bottomrule
\end{NiceTabular}
\end{table}
```

Table 1: A tabular whose caption has been specified by the key `caption`*

Last name	First name	Length of life
Churchill	Wiston	91
Nightingale^{a,b}	Florence^{c}	90
Schoelcher	Victor	89^{d}
Touchet	Marie^{c}	89
Wallis	John	87

Some text before the notes.

* It's possible to put a tabular note in the caption
^{a} Considered as the first nurse of history.
^{b} Nicknamed "the Lady with the Lamp".
^{c} This note is shared by two references.
^{d} The label of the note is overlapping.

13.4 Customisation of the tabular notes

The tabular notes can be customized with a set of keys available in `\NiceMatrixOptions`. The name of these keys is prefixed by `notes`.

- `notes/para`
- `notes/bottomrule`
- `notes/style`
- `notes/label-in-tabular`
- `notes/label-in-list`
- `notes/enumitem-keys`
- `notes/enumitem-keys-para`
- `notes/code-before`

For sake of commodity, it is also possible to set these keys in `\NiceMatrixOptions` via a key `notes` which takes in as value a list of pairs *key=value* where the name of the keys need no longer be prefixed by `notes`:

```
\NiceMatrixOptions
  {
    notes =
     {
       bottomrule ,
       style = ... ,
       label-in-tabular = ... ,
       enumitem-keys =
        {
          labelsep = ... ,
          align = ... ,
          ...
        }
     }
  }
```

We detail these keys.

- The key `notes/para` requires the composition of the notes (at the end of the tabular) in a single paragraph.

  Initial value: `false`

  That key is also available within a given environment.

- The key `notes/bottomrule` adds a `\bottomrule` of booktabs *after* the notes. Of course, that rule is drawn only if there is really notes in the tabular. The package booktabs must have been loaded (before or after the package nicematrix). If it is not, an error is raised.

  Initial value: `false`

  That key is also available within a given environment.

- The key `notes/style` is a command whose argument is specified by `#1` and which gives the style of numerotation of the notes. That style will be used by `\ref` when referencing a tabular note marked with a command `\label`. The labels formatted by that style are used, separated by commas, when the user puts several consecutive commands `\tabularnote`. The marker `#1` is meant to be the name of a LaTeX counter.

  Initial value: `\textit{\alph{#1}}`

  Another possible value should be a mere `\arabic{#1}`

- The key `notes/label-in-tabular` is a command whose argument is specified by `#1` which is used when formatting the label of a note in the tabular. Internally, this number of note has already been formatted by `notes/style` before sent to that command.

  Initial value: `\textsuperscript{#1}`

  In French, it's a tradition of putting a small space before the label of note. That tuning could be achieved by the following code:

  ```
  \NiceMatrixOptions{notes/label-in-tabular = \,\textsuperscript{~#1}}
  ```

- The key `notes/label-in-list` is a command whose argument is specified by `#1` which is used when formatting the label in the list of notes at the end of the tabular. Internally, this number of note has already been formatted by `notes/style` before sent to that command.

  Initial value: `\textsuperscript{#1}`

  In French, the labels of notes are not composed in upper position when composing the notes. Such behaviour could be achieved by:

  ```
  \NiceMatrixOptions{notes/label-in-list = ~#1.\nobreak\hspace{0.25em}}
  ```

  The command `\nobreak` is for the event that the option `para` is used.

- The notes are composed at the end of the tabular by using internally a style of list of enumitem. This style of list is defined as follows (with, of course, keys of enumitem):

  `noitemsep , leftmargin = * , align = left , labelsep = 0pt`

  The specification `align = left` in that style requires a composition of the label leftwards in the box affected to that label. With that tuning, the notes are composed flush left, which is pleasant when composing tabulars in the spirit of booktabs (see for example the table 1, p. 47).

  The key `notes/enumitem-keys` specifies a list of pairs *key=value* (following the specifications of enumitem) to customize that style of list (it uses internally the command `\setlist*` of enumitem).

- The key `notes/enumitem-keys-para` is similar to the previous one but corresponds to the type of list used when the option `para` is in force. Of course, when the option `para` is used, a list of type `inline` (as called by enumitem) is used and the pairs *key=value* should correspond to such a list of type `inline`.

  Initially, the style of list is defined by: `afterlabel = \nobreak, itemjoin = \quad`

- The key `notes/code-before` is a token list inserted by nicematrix just before the composition of the notes at the end of the tabular.

  Initial value: *empty*

  For example, if one wishes to compose all the notes in gray and `\footnotesize`, he should use that key:

  ```
  \NiceMatrixOptions{notes/code-before = \footnotesize \color{gray}}
  ```

  It's also possible to add `\raggedright` or `\RaggedRight` in that key (`\RaggedRight` is a command of ragged2e).

- The key `notes/detect-duplicates` activates the detection of the commands `\tabularnotes` with the same argument.

  Initial value : `true`

For an example of customisation of the tabular notes, see p. 65.

13.5 Use of {NiceTabular} with threeparttable

If you wish to use an environment `{NiceTabular}`, `{NiceTabular*}` or `{NiceTabularX}` in an environment `{threeparttable}` of the eponymous package, you have to patch `{threeparttable}` with the following code.

```
\makeatletter
\AddToHook{env/threeparttable/begin}
  {\TPT@hookin{NiceTabular}\TPT@hookin{NiceTabular*}\TPT@hookin{NiceTabularX}}
\makeatother
```

Nevertheless, the use of threeparttable in conjunction with nicematrix seems rather point-less because of the functionalities provided by nicematrix (see the key `caption` in the section 13.1, p. 44).

14 Other features

14.1 The key rounded-corners

The key `rounded-corners` that we will describe now has no direct link with the key `corners` (which is used to specify "empty corners") described in the part 5.3.3, p. 14.

The key `rounded-corners` specifies that the tabular should have rounded corners with a radius equal to the value of that key (the default value is 4 pt[63]). More precisely, that key has two effects that we describe now.

- All the commands for coloring the cells, columns and rows (in the `\CodeBefore` but also directly in the array will respect those rounded corners.
- When the key `hvlines` is used, the exterior rules will be drawn with rounded corners.[64]

That key is available in all the environments and commands (e.g. `\pAutoNiceMatrix`) of nicematrix and also in the command `\NiceMatrixOptions`.

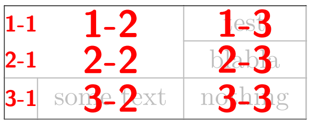
```
\begin{NiceTabular}
  [hvlines,rounded-corners]
  {ccc}
\CodeBefore
  \rowcolor{red!15}{1}
\Body
Last name & First name & Profession \\
Arvy & Jacques & Physicist \\
Jalon & Amandine & Physicist
\end{NiceTabular}
```

Last name	First name	Profession
Arvy	Jacques	Physicist
Jalon	Amandine	Physicist

14.2 Command \ShowCellNames

The command `\ShowCellNames`, which may be used in the `\CodeBefore` and in the `\CodeAfter` displays the name (with the form *i*-*j*) of each cell. When used in the `\CodeAfter`, that command applies a semi-transparent white rectangle to fade the array (caution: some PDF readers don't support transparency).

```
\begin{NiceTabular}{ccc}[hvlines,cell-space-limits=3pt]
  \Block{2-2}{} &            & test \\
                &            & blabla \\
                & some text & nothing
\CodeAfter \ShowCellNames
\end{NiceTabular}
```



14.3 Use of the column type S of siunitx

If the package siunitx is loaded (before or after nicematrix), it's possible to use the `S` column type of siunitx in the environments of nicematrix.

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

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

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

[63]This value is the initial value of the *rounded corners* of Tikz.

[64]Of course, when used in an environment with delimiters (`{pNiceMatrix}`, `{bNiceArray}`, etc.), the key `hvlines` does not draw the exterior rules.

14.4 Default column type in {NiceMatrix}

The environments without preamble (`{NiceMatrix}`, `{pNiceMatrix}`, `{bNiceMatrix}`, etc.) and the commande `\pAutoNiceMatrix` (and its variants) provide an option `columns-type` to specify the type of column which will be used (the initial value is, of course, `c`).
The keys `l` and `r` are shortcuts for `columns-type=l` and `columns-type=r`.

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

$$\begin{bmatrix} \cos x & -\sin x \\ \sin x & \cos x \end{bmatrix}$$

The key `columns-type` is available in `\NiceMatrixOptions` but with the prefix `matrix`, which means that its name is, within `\NiceMatrixOptions` : `matrix/columns-type`.

14.5 The command \rotate

The package `nicematrix` provides a command `\rotate`. When used in the beginning of a cell, this command composes the contents of the cell after a rotation of 90° in the direct sens.
In the following command, we use that command in the `code-for-first-row`.[65]

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

$$A = \begin{array}{c} \begin{array}{ccc} \scriptstyle\text{image of } e_1 & \scriptstyle\text{image of } e_2 & \scriptstyle\text{image of } e_3 \end{array} \\ \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \begin{array}{c} \scriptstyle e_1 \\ \scriptstyle e_2 \\ \scriptstyle e_3 \end{array} \end{array}$$

If the command `\rotate` is used in the "last row" (exterior to the matrix), the corresponding elements are aligned upwards as shown below.

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

$$A = \begin{array}{c} \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \begin{array}{c} \scriptstyle e_1 \\ \scriptstyle e_2 \\ \scriptstyle e_3 \end{array} \\ \begin{array}{ccc} \scriptstyle\text{image of } e_1 & \scriptstyle e_2 & \scriptstyle e_3 \end{array} \end{array}$$

The command `\rotate` has a key `c`: `\rotate[c]` (spaces are deleted after `\rotate[c]`). When that key is used, the content is composed in a `\vcenter` and, therefore, in most cases, we will have a vertical alignment.

Caution: the command `\rotate` is designed to be used in a `\Block` or in columns of type `l`, `r`, `c`, `w` ou `W`; if it is used in other column types (such as `p{...}`), the result will maybe differ from what is expected.

[65] It can also be used in `\RowStyle` (cf. p. 26.)

14.6 The option small

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

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

$$\left[\begin{smallmatrix} 1 & -2 & 3 & 4 & 5 \\ 0 & 3 & 2 & 1 & 2 \\ 0 & 1 & 1 & 2 & 3 \end{smallmatrix}\right] \begin{smallmatrix} \\ L_2 \gets 2L_1 - L_2 \\ L_3 \gets L_1 + L_3 \end{smallmatrix}$$

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

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

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

When the key `small` is in force, some functionalities of nicematrix are no longer available: for example, it's no longer possible to put vertical delimiters directly in the preamble of an environment with preamble (cf. section 11, p. 38).

14.7 The counters iRow and jCol

In the cells of the array, it's possible to use the LaTeX counters `iRow` and `jCol`[66] which represent the number of the current row and the number of the current column. We recall that the exterior "first row" (if it exists) has the number 0 and that the exterior "first column" (if it exists) has also the number 0. Of course, the user must not change the value of these counters `iRow` and `jCol` which are used internally by nicematrix.
In the `\CodeBefore` (cf. p. 19) and in the `\CodeAfter` (cf. p. 39), `iRow` represents the total number of rows (except the potential exterior rows: cf. p. 29) and `jCol` represents the total number of columns (except the potential exterior columns).

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

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

[66] There counters are actual LaTeX counters: the underlying TeX counters are `\c@iRow` and `\c@jCol`

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

The package nicematrix also provides commands in order to compose automatically matrices from a general pattern. These commands are `\AutoNiceMatrix`, `\pAutoNiceMatrix`, `\bAutoNiceMatrix`, `\vAutoNiceMatrix`, `\VAutoNiceMatrix` et `\BAutoNiceMatrix`.
These commands take in two mandatory arguments. The first is the format of the matrix, with the syntax n-p where n is the number of rows and p the number of columns. The second argument is the pattern (it's a list of tokens which are inserted in each cell of the constructed matrix).

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

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

There exists also `\AutoNiceArrayWithDelims` similar to `{NiceArrayWithDelims}`.

14.8 The key light-syntax

The option `light-syntax` (inpired by the package spalign) allows the user to compose the arrays with a lighter syntax, which gives a better legibility of the TeX source.
When this option is used, one should use the semicolon for the end of a row and spaces or tabulations to separate the columns. However, as usual in the TeX world, the spaces after a control sequence are discarded and the elements between curly braces are considered as a whole.

```
$\begin{bNiceMatrix}[light-syntax,first-row,first-col]
{} a             b                 ;
a  2\cos a       {\cos a + \cos b} ;
b \cos a+\cos b  { 2 \cos b }
\end{bNiceMatrix}$
```

$$\begin{array}{c} \\ a \\ b \end{array}\begin{array}{c} \begin{array}{cc} \quad a \quad & \quad b \quad \end{array} \\ \begin{bmatrix} 2\cos a & \cos a + \cos b \\ \cos a + \cos b & 2\cos b \end{bmatrix} \end{array}$$

It's possible to change the character used to mark the end of rows with the option `end-of-row`. As said before, the initial value is a semicolon.

When the option `light-syntax` is used, it is not possible to put verbatim material (for example with the command `\verb`) in the cells of the array.[67]

The key `light-syntax-expanded` has the same behaviour as the key `light-syntax` but the body of the environment is expanded (in the TeX sense[68]) before being split in rows (but after the extraction of a potential `\CodeAfter`).

14.9 Color of the delimiters

For the environments with delimiters (`{pNiceArray}`, `{pNiceMatrix}`, etc.), it's possible to change the color of the delimiters with the key `delimiters/color`.

```
$\begin{bNiceMatrix}[delimiters/color=red]
1 & 2 \\
3 & 4
\end{bNiceMatrix}$
```

$$\color{red}\left[\color{black}\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix}\color{red}\right]$$

This color also applies to the delimiters drawn by the command `\SubMatrix` (cf. p. 40) and for the delimiters directly specified in the preamble of the environments with preamble (cf. p. 38).

[67]The reason is that, when the option `light-syntax` is used, the whole content of the environment is loaded as a TeX argument to be analyzed. The environment doesn't behave in that case as a standard environment of LaTeX which only put TeX commands before and after the content.
[68]More precisely, it's an expansion of type `e` of L3.

14.10 The environment {NiceArrayWithDelims}

In fact, the environment `{pNiceArray}` and its variants are based upon a more general environment, called `{NiceArrayWithDelims}`. The first two mandatory arguments of this environment are the left and right delimiters used in the construction of the matrix.
It's possible to use `{NiceArrayWithDelims}` if we want to use atypical or asymmetrical delimiters.

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

$$\left\downarrow\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}\right\uparrow$$

14.11 The command \OnlyMainNiceMatrix

The command `\OnlyMainNiceMatrix` executes its argument only when it is in the main part of the array, that is to say when it is not in one of the exterior rows or one of the exterior columns. If it is used outside an environment of nicematrix, that command is no-op.
For an example of utilisation, see `tex.stackexchange.com/questions/488566`

15 Use of TikZ with nicematrix

15.1 The nodes corresponding to the contents of the cells

The package nicematrix creates a PGF/TikZ node[69] for each (non-empty) cell of the considered array. These nodes are used to draw the dotted lines between the cells of the matrix (inter alia).

Caution : By default, no node is created in a empty cell.

However, it's possible to impose the creation of a node with the command `\NotEmpty`. [70]

The creation of those nodes needs time and memory. It's possible to desactive ponctually the creation of those nodes with the key `no-cell-nodes` in order to speed up the compilations. But be careful: those nodes are used internally but nicematrix for several features. One should use the key `no-cell-nodes` only when those feateures are not used. Among these are the key `corners` and the commands for the continuous dotted lines (`\Cdots`, etc.).

The nodes of a document must have distinct names. That's why the names of the nodes created by nicematrix contains the number of the current environment. Indeed, the environments of nicematrix are numbered by a internal global counter.

In the environment with the number n, the node of the row i and column j has for name `nm-`n`-`i`-`j.

The command `\NiceMatrixLastEnv` provides the number of the last environment of nicematrix (for LaTeX, it's a "fully expandable" command and not a counter).

However, it's advisable to use instead the key `name`[71]. This key gives a name to the current environment. When the environment has a name, the nodes are accessible with the name "*name-i-j*" where *name* is the name given to the array and i and j the numbers of row and column. It's possible to use these nodes with PGF but the final user will probably prefer to use TikZ (which is a convenient layer upon PGF). However, one should remind that nicematrix doesn't load TikZ by default. In the following examples, we assume that TikZ has been loaded.

[69]We recall that TikZ is a layer over PGF. The extension nicematrix loads PGF but does not load TikZ. We speak of PGF/TikZ nodes to emphase the fact that the PGF nodes created by nicematrix may be used with PGF but also with TikZ. The final user will probably prefer to use TikZ rather than PGF.

[70]One should note that, with that command, the cell is considered as non-empty, which has consequencies for the continuous dotted lines (cf. p. 31) and the computation of the "corners" (cf. p. 14).

[71]The value of the key `name` is *expanded* (in the TeX sens).

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

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

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

In the `\CodeAfter`, the things are easier : one must refer to the nodes with the form i-j (we don't have to indicate the environment which is of course the current environment).

```
$\begin{pNiceMatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\CodeAfter
  \tikz \draw (2-2) circle (2mm) ;
\end{pNiceMatrix}$
```

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

The nodes of the last column (except the potential "last column" specified by `last-col`[72]) may also be indicated by i-`last`. Similarly, the nodes of the last row may be indicated by `last-`j.

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

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

Since those nodes are PGF nodes, one won't be surprised to learn that they are drawn by using a specific PGF style. That style is called `nicematrix/cell-node` and its definition in the source file `nicematrix.sty` is as follows:

```
\pgfset
  {
    nicematrix / cell-node /.style =
     {
       inner sep = 0 pt ,
       minimum width = 0 pt
     }
  }
```

The final user may modify that style by changing the values of the keys `text/rotate`, `inner xsep`, `inner ysep`, `inner sep`, `outer xsep`, `outer ysep`, `outer sep`, `minimum width`, `minimum height` and `minimum size`.

For an example of utilisation, see part 18.10, p. 76.

15.1.1 The key pgf-node-code

For the experienced users, nicematrix provides the key `pgf-node-code` which corresponds to some PGF node that will be executed at the creation, by PGF, of the nodes corresponding to the cells of the array. More pricisely, the value given to the key `pgf-node-code` will be passed in the fifth argument of the command `\pgfnode`. That value should contain at least an instruction such as `\pgfusepath`, `\pgfusepathqstroke`, `\pgfusepathqfill`, etc.

[72]For the exterior columns, cf. part 9, p. 29.

15.1.2 The columns V of varwidth

When the extension varwidth is loaded, the columns of the type `V` defined by varwidth are supported by nicematrix. It may be interessant to notice that, for a cell of a column of type `V`, the PGF/TikZ node created by nicematrix for the content of that cell has a width adjusted to the content of the cell. This is in contrast to the case of the columns of type `p`, `m` or `b` for which the nodes have always a width equal to the width of the column. In the following example, the command `\lipsum` is provided by the eponymous package.

```
\begin{NiceTabular}{V{10cm}}
\bfseries \large
Titre \\
\lipsum[1][1-4]
\CodeAfter
  \tikz \draw [rounded corners] (1-1) -| (last-|2) -- (last-|1) |- (1-1) ;
\end{NiceTabular}
```

Titre

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna.

We have used the nodes corresponding to the position of the potential rules, which are described below (cf. p. 58).

15.2 The "medium nodes" and the "large nodes"

In fact, the package nicematrix can create "extra nodes": the "medium nodes" and the "large nodes". The first ones are created with the option `create-medium-nodes` and the second ones with the option `create-large-nodes`.[73]

These nodes are not used by nicematrix by default, and that's why they are not created by default.

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

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

The names of the "large nodes" are constructed by adding the suffix "`-large`" to the names of the "normal nodes". In the following example, we have underlined the "large nodes". We consider that this example is self-explanatory.[74]

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

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

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

[74]There is no "large nodes" created in the exterior rows and columns (for these rows and columns, cf. p. 29).

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`.[75]

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

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

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

Be careful : These nodes are reconstructed from the contents of the contents cells of the array. Usually, they do not correspond to the cells delimited by the rules (if we consider that these rules are drawn).

Here is an array composed with the following code:

```
\large
\begin{NiceTabular}{wl{2cm}ll}[hvlines]
fraise & amande & abricot \\
prune & pêche & poire  \\[1ex]
noix & noisette & brugnon
\end{NiceTabular}
```

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

Here, we have colored all the cells of the array with `\chessboardcolors`.

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

Here are the "large nodes" of this array (without use of `margin` nor `extra-margin`).

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

The nodes we have described are not available by default in the `\CodeBefore` (described p. 19).
It's possible to have these nodes available in the `\CodeBefore` by using the key `create-cell-nodes` of the keyword `\CodeBefore` (in that case, the nodes are created first before the construction of the array by using information written on the `aux` file and created a second time during the construction of the array itself; this mechanism is not activated by default for efficiency reasons).

Here is an example which uses these nodes in the `\CodeAfter`.

```
\begin{NiceArray}{c@{\;}c@{\;}c@{\;}c@{\;}c}[create-medium-nodes]
    u_1 &-& u_0 &=& r     \\
    u_2 &-& u_1 &=& r     \\
    u_3 &-& u_2 &=& r     \\
```

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

```
    u_4 &-& u_3 &=& r     \\
    \phantom{u_5} & &  \phantom{u_4}    &\smash{\vdots} &       \\
    u_n &-& u_{n-1} &=& r \\[3pt]
    \hline
    u_n &-& u_0 &=& nr \\
\CodeAfter
    \tikz[very thick, red, opacity=0.4, name suffix = -medium]
    \draw (1-1.north west) -- (2-3.south east)
    (2-1.north west) -- (3-3.south east)
    (3-1.north west) -- (4-3.south east)
    (4-1.north west) -- (5-3.south east)
    (5-1.north west) -- (6-3.south east) ;
\end{NiceArray}
```

$$\begin{array}{ccccc}
u_1 & - & u_0 & = & r \\
u_2 & - & u_1 & = & r \\
u_3 & - & u_2 & = & r \\
u_4 & - & u_3 & = & r \\
 & & & \vdots & \\
u_n & - & u_{n-1} & = & r \\
\hline
u_n & - & u_0 & = & nr
\end{array}$$

15.3 The nodes which indicate the position of the rules

The package nicematrix creates a PGF/TikZ node merely called i (with the classical prefix) at the intersection of the horizontal rule of number i and the vertical rule of number i (more specifically the potential position of those rules because maybe there are not actually drawn). The last node has also an alias called `last`.

There are also nodes called i`.1`, i`.2`, i`.25`, i`.3`, i`.4`, i`.5`, i`.6`, i`.7`, i`.75`, i`.8` and i`.9` between the node i and the node $i+1$.

These nodes are available in the `\CodeBefore` and the `\CodeAfter`.

	tulipe	lys
arum		violette mauve
muguet	dahlia	

If we use TikZ (we remind that nicematrix does not load TikZ by default, by only PGF, which is a sub-layer of TikZ), we can access, in the `\CodeAfter` but also in the `\CodeBefore`, to the intersection of the (potential) horizontal rule i and the (potential) vertical rule j with the syntax `(`i`-|`j`)`.

```
\begin{NiceMatrix}
\CodeBefore
  \tikz \draw [fill=red!15] (7-|4) |- (8-|5) |- (9-|6) |- cycle ;
\Body
1 \\
1 & 1 \\
1 & 2 &  1 \\
1 & 3 &  3 &  1 \\
1 & 4 &  6 &  4 &  1 \\
1 & 5 & 10 & 10 &  5 &  1 \\
1 & 6 & 15 & 20 & 15 &  6 &  1 \\
1 & 7 & 21 & 35 & 35 & 21 &  7 & 1 \\
1 & 8 & 28 & 56 & 70 & 56 & 28 & 8 & 1
\end{NiceMatrix}
```

1								
1	1							
1	2	1						
1	3	3	1					
1	4	6	4	1				
1	5	10	10	5	1			
1	6	15	20	15	6	1		
1	7	21	35	35	21	7	1	
1	8	28	56	70	56	28	8	1

The "decimal" nodes (like i.4) may be used, for example to cross a row of a matrix (if TikZ is loaded).

```
$\begin{pNiceArray}{ccc|c}
2 & 1 & 3 & 0 \\
3 & 3 & 1 & 0 \\
3 & 3 & 1 & 0
\CodeAfter
  \tikz \draw [red] (3.4-|1) -- (3.4-|last) ;
\end{pNiceArray}$
```

$$\left(\begin{array}{ccc|c} 2 & 1 & 3 & 0 \\ 3 & 3 & 1 & 0 \\ 3 & 3 & 1 & 0 \end{array}\right)$$

15.4 The nodes corresponding to the command \SubMatrix

The command `\SubMatrix` available in the `\CodeAfter` has been described p. 40.

If a command `\SubMatrix` has been used with the key `name` with an expression such as `name=`*`MyName`* three PGF/TikZ nodes are created with the names *`MyName`*`-left`, *`MyName`* and *`MyName`*`-right`.

The nodes *`MyName`*`-left` and *`MyName`*`-right` correspond to the delimiters left and right and the node *`MyName`* correspond to the submatrix itself.

In the following example, we have highlighted these nodes (the submatrix itself has been created with `\SubMatrix\{{2-2}{3-3}\}`).

$$\begin{pmatrix} 121 & 23 & 345 & 345 \\ 45 & \left\{\begin{matrix} 346 & 863 \\ 38458 & 34 \end{matrix}\right\} & & 444 \\ 3462 & & & 294 \\ 34 & 7 & 78 & 309 \end{pmatrix}$$

16 API for the developers

The package nicematrix provides two variables which are internal but public[76]:

- `\g_nicematrix_code_before_tl` ;
- `\g_nicematrix_code_after_tl`.

These variables contain the code of what we have called the "`code-before`" (usually specified at the beginning of the environment with the syntax using the keywords `\CodeBefore` and `\Body`) and the "`code-after`" (usually specified at the end of the environment after the keyword `\CodeAfter`). The developer can use them to add code from a cell of the array (the affectation must be global, allowing to exit the cell, which is a TeX group).

One should remark that the use of `\g_nicematrix_code_before_tl` needs one compilation more (because the instructions are written on the `aux` file to be used during the next run).

[76]According to the LaTeX3 conventions, each variable with name beginning with `\g_nicematrix` ou `\l_nicematrix` is public and each variable with name beginning with `\g__nicematrix` or `\l__nicematrix` is private.

Example : We want to write a command `\crossbox` to draw a cross in the current cell. This command will take in an optional argument between square brackets for a list of pairs *key-value* which will be given to TikZ before the drawing.
It's possible to program such command `\crossbox` as follows, explicitly using the public variable `\g_nicematrix_code_before_tl`.

```
\ExplSyntaxOn
\cs_new_protected:Nn \__pantigny_crossbox:nnn
  {
    \tikz \draw [ #3 ]
          ( #1 -| \int_eval:n { #2 + 1 } ) -- ( \int_eval:n { #1 + 1 } -| #2 )
          ( #1 -| #2 ) -- ( \int_eval:n { #1 + 1 } -| \int_eval:n { #2 + 1 } ) ;
  }

\NewDocumentCommand \crossbox { ! O { } }
  {
    \tl_gput_right:Nx \g_nicematrix_code_before_tl
      {
        \__pantigny_crossbox:nnn
          { \int_use:c { c@iRow } }
          { \int_use:c { c@jCol } }
          { \exp_not:n { #1 } }
      }
    \ignorespaces
  }
\ExplSyntaxOff
```

We have used the LaTeX `counters iRow` and `jCol` provided by nicematrix (cf. p. 52).

Here is an example of utilisation:

```
\begin{NiceTabular}{ccc}[hvlines]
\CodeBefore
  \arraycolor{gray!10}
\Body
merlan & requin & cabillaud \\
baleine & \crossbox[red] & morue \\
mante & raie & poule
\end{NiceTabular}
```

merlan	requin	cabillaud
baleine		morue
mante	raie	poule

17 Technical remarks

First remark: the package underscore must be loaded before nicematrix. If it is loaded after, an error will be raised.

17.1 Diagonal lines

By default, all the diagonal lines[77] 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.

[77]We speak of the lines created by `\Ddots` and not the lines created by a command `\line` in the `\CodeAfter`.

Example with parallelization (default):

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

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

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

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

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

The same example without parallelization:

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

It's possible to specify the instruction `\Ddots` which will be drawn first (and which will be used to draw the other diagonal dotted lines when the parallelization is in force) with the key `draw-first`: `\Ddots[draw-first]`.

17.2 The "empty" cells

The package nicematrix uses, in several circumstancies, the concept of "empty cell'. For example, an instruction like `\Ldots`, `\Cdots`, etc. (cf. p. 31) tries to determine the first non-empty cell on both sides. In the same way, when the key `corners` is used (cf. p. 14), nicematrix computes corners consisting of empty cells.
However, an "empty cell" is not necessarily a cell with no TeX content (that is to say a cell with no token between the two ampersands `&`). 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.

- For the columns of type `p`, `m`, `b`, `V`[78] and `X`[79], the cell is empty if (and only if) its content in the TeX code is empty (there is only spaces between the ampersands `&`).

- For the columns of type `c`, `l`, `r` and `w{...}{...}`, the cell is empty if (and only if) its TeX output has a width equal to zero.

- A cell containing the command `\NotEmpty` is not empty (and a PGF/TikZ node is created in that cell).

---

[78] The columns of type `V` are provided by varwidth, which must be loaded: cf. p. 28.
[79] See p. 28

- A cell with only 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.

17.3 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[80]. The environment `{matrix}` of amsmath and its variants (`{pmatrix}`, `{vmatrix}`, etc.) of amsmath prefer to delete these spaces with explicit instructions `\hskip -\arraycolsep`[81]. The package nicematrix does the same in all its environments, `{NiceArray}` included. However, if the user wants the environment `{NiceArray}` behaving by default like the environment `{array}` of array (for example, when adapting an existing document) it's possible to control this behaviour with the option `exterior-arraycolsep`, set by the command `\NiceMatrixOptions`. With this option, exterior spaces of length `\arraycolsep` will be inserted in the environments `{NiceArray}` (the other environments of nicematrix are not affected).

17.4 Incompatibilities

There might be incompatibilities between nicematrix and babel with the language which activate (in the TeX sens) some characters, in particular the character `<`.
Par exemple, for Spanish, it's recommended to deactivate the abreviations at load-time with the instruction:

```
\usepackage[spanish,es-noshorthands]{babel}
```

The extension nicematrix is usually not compatible with the classes and packages that redefine the environments `{tabular}` and `{array}`. In particular, it's the case of the class `socg-lipics-v2021`. However, in that case, it's possible to load the class with the key `notab` which requires that the environment `{tabular}` is not redefined.

The package nicematrix is not compatible with the class ieeeaccess because that class is not compatible with PGF/TikZ. However, there is a simple workaround by writing:[82]

```
\let\TeXyear\year
\documentclass{IEEEaccess}
\let\year\TeXyear
```

In order to use nicematrix with the class aastex631 (of the *American Astronomical Society*), you have to add the following instructions in the preamble of your document :

```
\BeforeBegin{NiceTabular}{\let\begin\BeginEnvironment\let\end\EndEnvironment}
\BeforeBegin{NiceArray}{\let\begin\BeginEnvironment}
\BeforeBegin{NiceMatrix}{\let\begin\BeginEnvironment}
```

The package nicematrix is not fully compatible with the packages and classes of LuaTeX-ja: the detection of the empty corners (cf. p. 14) may be wrong in some circonstances.

The package nicematrix is not fully compatible with the package arydshln (because this package redefines many internals of array) and does not support the columns `V` of boldline (because the letter `V` is reserved for the columns `V` of varwidth). By any means, nicematrix provides, with the key `custom-line` (cf. part 5.3.5, p. 16), tools to draw dashed rules and rules of different widths.

The columns `d` of dcolumn are not supported (but it's possible to use the columns `S` of siunitx).

[80] In 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).*
[81] And not by inserting `@{}` on both sides of the preamble of the array. As a consequence, the length of the `\hline` is not modified and may appear too long, in particular when using square brackets.
[82] See `https://tex.stackexchange.com/questions/528975`

17.5 Compatibility with the Tagging Project of LaTeX

New 7.0

Since the version 7.0, the extension nicematrix is compatible with the *Tagging Project* of LaTeX (whose aim is to create tagged PDF). At this time, only simple standard tabulars are correctly tagged.
Here is a example of code which will produce a generate tagged PDF.

```
\DocumentMetadata{
  lang        = en,
  pdfversion  = 2.0,
  pdfstandard = ua-2,
  pdfstandard = a-4f,
  testphase   = {phase-III, table, math}
}
\documentclass{article}
\usepackage{lmodern}
\usepackage{nicematrix}

\begin{document}

\begin{center}
\tagpdfsetup{table/header-rows=1}
\begin{NiceTabular}{ccc}[hvlines]
First name & Last name & Age \\
Paul       & Imbert    & $66$ \\
John       & Sarrus    & $23$ \\
Liz        & Taylor    & $100$ \\
George     & Adams     & $34$
\end{NiceTabular}
\end{center}

\end{document}
```

18 Examples

18.1 Utilisation of the key "tikz" of the command \Block

The key `TikZ` of the command `\Block` is available only when TikZ is loaded.[83]
For the following example, we also need the TikZ library `patterns`.

```
\usetikzlibrary{patterns}

\ttfamily \small
\begin{NiceTabular}{X[m]X[m]X[m]}[hvlines,cell-space-limits=3pt,rounded-corners]
  \Block[tikz={pattern=grid,pattern color=lightgray}]{}
    {pattern = grid,\\ pattern color = lightgray}
& \Block[tikz={pattern = north west lines,pattern color=blue}]{}
    {pattern = north west lines,\\ pattern color = blue}
& \Block[tikz={outer color = red!50, inner color=white }]{2-1}
    {outer color = red!50,\\ inner color = white} \\
  \Block[tikz={pattern = sixpointed stars, pattern color = blue!15}]{}
    {pattern = sixpointed stars,\\ pattern color = blue!15}
& \Block[tikz={left color = blue!50}]{}
    {left color = blue!50} \\
\end{NiceTabular}
```

[83]By default, nicematrix only loads PGF, which is a sub-layer of TikZ.

pattern = grid, pattern color = lightgray	pattern = north west lines, pattern color = blue	outer color = red!50, inner color = white
pattern = sixpointed stars, pattern color = blue!15	left color = blue!50	

In the following example, we use the key `tikz` to hatch a row of the tabular. Remark that you use the key `transparent` of the command `\Block` in order to have the rules drawn in the block.[84]

```
\begin{NiceTabular}{ccc}[hvlines]
\CodeBefore
  \columncolor[RGB]{169,208,142}{2}
\Body
one & two & three \\
\Block[transparent, tikz={pattern = north west lines, pattern color = gray}]{1-*}{}
four & five & six \\
seven & eight & nine
\end{NiceTabular}
```

one	two	three
four	five	six
seven	eight	nine

18.2 Use with tcolorbox

Here is an example of use of `{NiceTabular}` within a command `\tcbox` of tcolorbox. We have used the key `hvlines-except-borders` in order to draw all the rules, except on the borders (which are, of course, added by tcolorbox).

```
\tcbset
  {
    colframe = blue!50!black ,
    colback = white ,
    fonttitle = \bfseries ,
    nobeforeafter ,
    center title
  }

\tcbox
  [
    left = 0mm ,
    right = 0mm ,
    top = 0mm ,
    bottom = 0mm ,
    boxsep = 0mm ,
    toptitle = 0.5mm ,
    bottomtitle = 0.5mm ,
    title = My table
  ]
  {
    \renewcommand{\arraystretch}{1.2}% <-- the % is mandatory here
    \begin{NiceTabular}{rcl}[hvlines-except-borders,rules/color=blue!50!black]
    \CodeBefore
      \rowcolor{red!15}{1}
    \Body
```

[84]By default, the rules are not drawn in the blocks created by the command `\Block` (cf. section 5 p. 11): with the key `transparent`, they are drawn (the block becomes *transparent* to the exterior rules).

```
    One & Two & Three \\
    Men & Mice & Lions \\
    Upper & Middle & Lower
  \end{NiceTabular}
}
```

My table		
One	Two	Three
Men	Mice	Lions
Upper	Middle	Lower

18.3 Notes in the tabulars

The tools provided by nicematrix for the composition of the tabular notes have been presented in the section 13 p. 44.

Let's consider that we wish to number the notes of a tabular with stars.[85]

First, we write a command `\stars` similar to the well-known commands `\arabic`, `\alph`, `\Alph`, etc. which produces a number of stars equal to its argument[86].

```
\ExplSyntaxOn
\NewDocumentCommand { \stars } { m }
  { \prg_replicate:nn { \value { #1 } } { \( \star \) } }
\ExplSyntaxOff
```

Of course, we change the style of the labels with the key `notes/style`. However, it would be interesting to change also some parameters in the type of list used to compose the notes at the end of the tabular. First, we required a composition flush right for the labels with the setting `align=right`. Moreover, we want the labels to be composed on a width equal to the width of the widest label. The widest label is, of course, the label with the greatest number of stars. We know that number: it is equal to `\value{tabularnote}` (because `tabularnote` is the LaTeX counter used by `\tabularnote` and, therefore, at the end of the tabular, its value is equal to the total number of tabular notes). We use the key `widest*` of enumitem in order to require a width equal to that value: `widest*=\value{tabularnote}`.

```
\NiceMatrixOptions
  {
    notes =
     {
       style = \stars{#1} ,
       enumitem-keys =
        {
          widest* = \value{tabularnote} ,
          align = right
        }
     }
  }
\begin{NiceTabular}{@{}llr@{}}
\toprule \RowStyle{\bfseries}
Last name & First name & Birth day \\
\midrule
Achard\tabularnote{Achard is an old family of the Poitou.}
& Jacques & 5 juin 1962 \\
```

[85]Of course, it's realistic only when there is very few notes in the tabular.
[86]In fact: the value of its argument.

```
Lefebvre\tabularnote{The name Lefebvre is an alteration of the name Lefebure.}
& Mathilde & 23 mai 1988 \\
Vanesse & Stephany & 30 octobre 1994 \\
Dupont & Chantal & 15 janvier 1998 \\
\bottomrule
\end{NiceTabular}
```

Last name	**First name**	**Birth day**
Achard*	Jacques	June 5, 2005
Lefebvre**	Mathilde	January 23, 1975
Vanesse	Stephany	October 30, 1994
Dupont	Chantal	January 15, 1998

*Achard is an old family of the Poitou.
**The name Lefebvre is an alteration of the name Lefebure.

18.4 Dotted lines

An example with the resultant of two polynoms:

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

$$\begin{vmatrix}
a_0 & & & & b_0 & & \\
a_1 & \ddots & & & b_1 & \ddots & \\
\vdots & \ddots & & & \vdots & \ddots & b_0 \\
a_p & & & a_0 & & & b_1 \\
 & \ddots & & a_1 & b_q & & \vdots \\
 & & & \vdots & & \ddots & \\
 & & & a_p & & & b_q
\end{vmatrix}$$

An example for a linear system:

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

$$
\begin{array}{cl}
\begin{pmatrix}
1 & 1 & 1 & \cdots & 1 & \bigm| & 0 \\
0 & 1 & 0 & \cdots & 0 & \bigm| & \vdots \\
0 & 0 & 1 & \ddots & \vdots & \bigm| & \vdots \\
\vdots & \ddots & \ddots & \ddots & 0 & \bigm| & \vdots \\
0 & \cdots & 0 & & 1 & \bigm| & 0
\end{pmatrix}
&
\begin{array}{l}
\\ L_2 \gets L_2 - L_1 \\ L_3 \gets L_3 - L_1 \\ \vdots \\ L_n \gets L_n - L_1
\end{array}
\end{array}
$$

18.5 Dotted lines which are no longer dotted

The option `line-style` controls the style of the lines drawn by `\Ldots`, `\Cdots`, etc. Thus, it's possible with these commands to draw lines which are not longer dotted (TikZ should be loaded).

```
\setcounter{MaxMatrixCols}{12}
\newcommand{\blue}{\color{blue}}
\[\begin{pNiceMatrix}[last-row,last-col,nullify-dots,xdots/line-style={dashed,blue}]
1& & & \Vdots & & & & \Vdots \\
& \Ddots[line-style=standard] \\
& & 1 \\
\Cdots & & & \blue 0 & \Cdots & & & \blue 1 & & & \Cdots & \blue \leftarrow i \\
& & & & 1 \\
& & &\Vdots & & \Ddots[line-style=standard] & & \Vdots \\
& & & & & & 1 \\
\Cdots & & & \blue 1 & \Cdots & & \Cdots & \blue 0 & & & \Cdots & \blue \leftarrow j \\
& & & & & & & & 1 \\
& & & & & & & & \Ddots[line-style=standard] \\
& & & \Vdots & & & & \Vdots & & & 1 \\
& & & \blue \overset{\uparrow}{i} & & & & \blue \overset{\uparrow}{j} \\
\end{pNiceMatrix}\]
```

$$
\begin{array}{cl}
\begin{pmatrix}
1 & & & \vdots & & & \vdots & & \\
& \ddots & & \vdots & & & \vdots & & \\
& & 1 & \vdots & & & \vdots & & \\
\cdots & \cdots & \cdots & 0 & \cdots & \cdots & 1 & \cdots & \cdots \\
& & & \vdots & 1 & & \vdots & & \\
& & & \vdots & & \ddots & \vdots & & \\
& & & \vdots & & & 1 \\
\cdots & \cdots & \cdots & 1 & \cdots & \cdots & 0 & \cdots & \cdots \\
& & & \vdots & & & \vdots & 1 & \\
& & & \vdots & & & \vdots & & \ddots \\
& & & \vdots & & & \vdots & & & 1
\end{pmatrix}
&
\begin{array}{l}
\\ \\ \\ \leftarrow i \\ \\ \\ \\ \leftarrow j \\ \\ \\ \\
\end{array}
\\
\begin{array}{ccccccccc}
& & & \overset{\uparrow}{i} & & & \overset{\uparrow}{j} & &
\end{array}
&
\end{array}
$$

In fact, it's even possible to draw solid lines with the commands `\Cdots`, `\Vdots`, etc.[87]

```
\NiceMatrixOptions{xdots={horizontal-labels,line-style = <->}}
```

```
$\begin{pNiceArray}{ccc|cc}[first-row,last-col,margin]
\Hdotsfor{3}^{3} & \Hdotsfor{2}^{2} \\
2 & 1 & 1 & 1 & 1 & \Vdotsfor{3}^{3}\\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\Hline
1 & 1 & 1 & 1 & 1 & \Vdotsfor{2}^{2} \\
1 & 1 & 1 & 1 & 1 \\
\end{pNiceArray}$
```

$$\begin{array}{cc}
\begin{array}{ccc|cc}\multicolumn{3}{c}{\overset{3}{\longleftrightarrow}} & \multicolumn{2}{c}{\overset{2}{\longleftrightarrow}}\end{array} & \\
\left(\begin{array}{ccc|cc} 2&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\ \hline 1&1&1&1&1\\1&1&1&1&1\end{array}\right) & \begin{array}{l} \updownarrow 3 \\ \\ \updownarrow 2\end{array}
\end{array}$$

If you want the label *on the line*, you should use the special token "`:`" instead of "`^`":

```
\NiceMatrixOptions{xdots={horizontal-labels,line-style = <->}}
```

```
$\begin{pNiceArray}{ccc|cc}[first-row,last-col,margin]
\Hdotsfor{3}:{3} & \Hdotsfor{2}:{2} \\
2 & 1 & 1 & 1 & 1 & \Vdotsfor{3}:{3} \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\Hline
1 & 1 & 1 & 1 & 1 &
\Vdotsfor{2}:{2 \rlap{ \smash{rows}}} \\
1 & 1 & 1 & 1 & 1 \\
\end{pNiceArray}$
```

$$\begin{array}{cc}
\begin{array}{ccc|cc}\multicolumn{3}{c}{\longleftarrow 3 \longrightarrow} & \multicolumn{2}{c}{\leftarrow 2 \rightarrow}\end{array} & \\
\left(\begin{array}{ccc|cc} 2&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\ \hline 1&1&1&1&1\\1&1&1&1&1\end{array}\right) & \begin{array}{l} \updownarrow 3 \\ \\ \updownarrow 2 \text{ lignes}\end{array}
\end{array}$$

If one prefers the braces of the library decorations.pathreplacing of TikZ, the best way is to use the commands `\Hbrace` and `\Vbrace` provided by nicematrix (cf. p. 37).[88]

```
\NiceMatrixOptions{xdots/horizontal-labels}
$\begin{pNiceArray}{ccc|cc}[first-row,last-col,margin]
\Hbrace{3}{3} & \Hbrace{2}{2} \\
2 & 1 & 1 & 1 & 1 & \Vbrace{3}{3} \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\Hline
1 & 1 & 1 & 1 & 1 & \Vbrace{2}{2} \\
1 & 1 & 1 & 1 & 1 \\
\end{pNiceArray}$
```

$$\begin{array}{cc}
\begin{array}{ccc|cc}\multicolumn{3}{c}{\overbrace{\qquad\qquad}^{3}} & \multicolumn{2}{c}{\overbrace{\qquad}^{2}}\end{array} & \\
\left(\begin{array}{ccc|cc} 2&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\ \hline 1&1&1&1&1\\1&1&1&1&1\end{array}\right) & \begin{array}{l} \}\, 3 \\ \\ \}\, 2\end{array}
\end{array}$$

---

[87] In this document, the TikZ library arrows.meta has been loaded, which impacts the shape of the arrow tips.

[88] Remark that those commands are available only if TikZ has been loaded with the library decorations.pathreplacing: `\usepackage{TikZ} \usetikzlibrary{decorations.pathreplacing}`.

It's possible to change the TikZ style of those braces by changing the TikZ style `nicematrix/brace`. In the following example, we assume that the TikZ extension `calligraphy` has been loaded:

```
\usetikzlibrary{calligraphy}

\NiceMatrixOptions{xdots/horizontal-labels}
\tikzset
  {
    nicematrix/brace/.style =
      {
        decoration =
          {
            calligraphic brace ,
            amplitude = 0.4 em ,
            raise = -0.25 em
          } ,
        line width = 0.1 em ,
        decorate ,
      }
  }
```

```
$\begin{pNiceArray}{ccc|cc}[first-row,last-col,margin]
\Hbrace{3}{3} & \Hbrace{2}{2} \\
2 & 1 & 1 & 1 & 1 & \Vbrace{3}{3} \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\Hline
1 & 1 & 1 & 1 & 1 & \Vbrace{2}{2} \\
1 & 1 & 1 & 1 & 1 \\
\end{pNiceArray}$
```

$$\begin{array}{cc}
\begin{array}{cc} \overbrace{\hphantom{2\quad 1\quad 1}}^{3} & \overbrace{\hphantom{1\quad 1}}^{2} \end{array} & \\
\left(\begin{array}{ccc|cc} 2 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{array}\right) & \begin{array}{l} \left.\vphantom{\begin{array}{c}1\\1\\1\end{array}}\right\}3 \\ \left.\vphantom{\begin{array}{c}1\\1\end{array}}\right\}2 \end{array}
\end{array}$$

If one prefers the curly braces of the current mathematical font of LaTeX, one should use the commands `\SubMatrix`, `\OverBrace` and `\UnderBrace` in the `\CodeAfter`.

```
$\begin{pNiceArray}{ccc|cc}[margin,last-col]
2 & 1 & 1 & 1 & 1 & \Block{3-1}{\quad 3} \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\Hline
1 & 1 & 1 & 1 & 1 & \Block{2-1}{\quad 2} \\
1 & 1 & 1 & 1 & 1 \\
\CodeAfter
  \OverBrace[shorten,yshift=1.5mm]{1-1}{1-3}{3}
  \OverBrace[shorten,yshift=1.5mm]{1-4}{1-5}{2}
  \SubMatrix{.}{1-1}{3-5}{\rbrace}[xshift=3.5mm]
  \SubMatrix{.}{4-1}{5-5}{\rbrace}[xshift=3.5mm]
\end{pNiceArray}$
```

$$\begin{array}{cc}
\begin{array}{cc} \overbrace{\hphantom{2\quad 1\quad 1}}^{3} & \overbrace{\hphantom{1\quad 1}}^{2} \end{array} & \\
\left(\begin{array}{ccc|cc} 2 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{array}\right) & \begin{array}{l} \left.\vphantom{\begin{array}{c}1\\1\\1\end{array}}\right\}\ 3 \\ \left.\vphantom{\begin{array}{c}1\\1\end{array}}\right\}\ 2 \end{array}
\end{array}$$

Or course, the output may seem disappointing. That's why, for this type of use, we recommend the use of the commands `\Hbrace` and `\Vbrace` (provided by `nicematrix`), as shown in the previous examples.

18.6 Dashed rules

In the following example, we use the command `\Block` to draw dashed rules. For that example, TikZ should be loaded (by `\usepackage{tikz}`).

```
\begin{pNiceMatrix}
\Block[borders={bottom,right,tikz=dashed}]{2-2}{}
1 & 2 & 0  & 0 & 0 & 0 \\
4 & 5 & 0  & 0 & 0 & 0 \\
0 & 0 & \Block[borders={bottom,top,right,left,tikz=dashed}]{2-2}{}
        7  & 1 & 0 & 0 \\
0 & 0 & -1 & 2 & 0 & 0 \\
0 & 0 & 0  & 0 & \Block[borders={left,top,tikz=dashed}]{2-2}{}
                 3 & 4 \\
0 & 0 & 0  & 0 & 1 & 4
\end{pNiceMatrix}
```

$$\begin{pmatrix} 1 & 2 & 0 & 0 & 0 & 0 \\ 4 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 7 & 1 & 0 & 0 \\ 0 & 0 & -1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 4 \\ 0 & 0 & 0 & 0 & 1 & 4 \end{pmatrix}$$

18.7 Stacks of matrices

We often need to compose mathematical matrices on top on each other (for example for the resolution of linear systems).

In order to have the columns aligned one above the other, it's possible to fix a width for all the columns. That's what is done in the following example with the environment `{NiceMatrixBlock}` and its option `auto-columns-width`.

```
\begin{NiceMatrixBlock}[auto-columns-width]
\NiceMatrixOptions
  {
    light-syntax,
    last-col, code-for-last-col = \color{blue}\scriptstyle,
    vlines = 5 ,
    matrix/columns-type = r ,
    no-cell-nodes % only for speedup
  }
\setlength{\extrarowheight}{1mm}

\quad $\begin{pNiceMatrix}
12  -8  7  5   3 {} ;
 3 -18 12  1   4    ;
-3 -46 29 -2 -15    ;
 9  10 -5  4   7
\end{pNiceMatrix}$

\smallskip
\quad $\begin{pNiceMatrix}
12  -8   7  5   3 ;
0   64 -41  1  19  { L_2 \gets L_1-4L_2  } ;
0 -192 123 -3 -57  { L_3 \gets L_1+4L_3  } ;
0 -64   41 -1 -19  { L_4 \gets 3L_1-4L_4 } ;
\end{pNiceMatrix}$

\smallskip
\quad $\begin{pNiceMatrix}
12 -8   7 5  3 ;
0  64 -41 1 19 ;
0   0   0 0  0 { L_3 \gets 3 L_2 + L_3 }
\end{pNiceMatrix}$

\smallskip
```

```
\quad $\begin{pNiceMatrix}
12 -8   7 5   3 {} ;
0  64 -41 1 19    ;
\end{pNiceMatrix}$
\end{NiceMatrixBlock}
```

$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 3 & -18 & 12 & 1 & 4 \\ -3 & -46 & 29 & -2 & -15 \\ 9 & 10 & -5 & 4 & 7 \end{array}\right)$$

$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & -192 & 123 & -3 & -57 \\ 0 & -64 & 41 & -1 & -19 \end{array}\right)\begin{array}{l} \\ {\scriptstyle L_2 \gets L_1 - 4L_2} \\ {\scriptstyle L_3 \gets L_1 + 4L_3} \\ {\scriptstyle L_4 \gets 3L_1 - 4L_4} \end{array}$$

$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right)\begin{array}{l} \\ \\ {\scriptstyle L_3 \gets 3L_2 + L_3} \end{array}$$

$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \end{array}\right)$$

However, one can see that the last matrix is not perfectly aligned with others. That's why, in LaTeX, the parenthesis have not exactly the same width (smaller parenthesis are a bit slimer).

In order the solve that problem, it's possible to require the delimiters to be composed with the maximal width, thanks to the boolean key `delimiters/max-width`.

```
\begin{NiceMatrixBlock}[auto-columns-width]
\NiceMatrixOptions
  {
    delimiters/max-width,
    light-syntax,
    last-col, code-for-last-col = \color{blue}\scriptstyle,
    vlines = 5 ,
    matrix/columns-type = r ,
    no-cell-nodes % only for speedup
  }
\setlength{\extrarowheight}{1mm}

\quad $\begin{pNiceMatrix}
12  -8  7  5   3 {} ;
 3 -18 12  1   4    ;
-3 -46 29 -2 -15    ;
 9  10 -5  4   7
\end{pNiceMatrix}$
...
\end{NiceMatrixBlock}
```

$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 3 & -18 & 12 & 1 & 4 \\ -3 & -46 & 29 & -2 & -15 \\ 9 & 10 & -5 & 4 & 7 \end{array}\right)$$

$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & -192 & 123 & -3 & -57 \\ 0 & -64 & 41 & -1 & -19 \end{array}\right)\begin{array}{l} \\ {\scriptstyle L_2 \gets L_1 - 4L_2} \\ {\scriptstyle L_3 \gets L_1 + 4L_3} \\ {\scriptstyle L_4 \gets 3L_1 - 4L_4} \end{array}$$

$$
\begin{array}{l}
\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right) \begin{array}{l} \\ \\ \scriptstyle L_3 \gets 3L_2+L_3 \end{array} \\
\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \end{array}\right)
\end{array}
$$

If you wish an alignment of the different matrices without the same width for all the columns, you can construct a unique array and place the parenthesis with commands `\SubMatrix` in the `\CodeAfter`. Of course, that array can't be broken by a page break.

```
\setlength{\extrarowheight}{1mm}
\[\begin{NiceMatrix}[r, last-col=6, code-for-last-col = \scriptstyle \color{blue}]
12 &  -8  &  7 & 5 &  3 \\
 3 & -18  & 12 & 1 &  4 \\
-3 & -46  & 29 &-2 &-15 \\
 9 & 10   &-5  &4  & 7 \\[1mm]
12 & -8   & 7  &5  & 3 \\
0  & 64   &-41 & 1 & 19 & L_2 \gets L_1-4L_2  \\
0  & -192 &123 &-3 &-57 & L_3 \gets L_1+4L_3  \\
0  & -64  & 41 &-1 &-19 & L_4 \gets 3L_1-4L_4 \\[1mm]
12 & -8   &7   &5   & 3 \\
0  & 64   &-41 &1   &19 \\
0  &  0   &0   &0   & 0  & L_3 \gets 3L_2+L_3 \\[1mm]
12 & -8   &7   &5   & 3 \\
0  & 64   &-41 & 1 & 19 \\
\CodeAfter [sub-matrix/vlines=4]
   \SubMatrix({1-1}{4-5})
   \SubMatrix({5-1}{8-5})
   \SubMatrix({9-1}{11-5})
   \SubMatrix({12-1}{13-5})
\end{NiceMatrix}\]
```

$$
\begin{array}{l}
\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 3 & -18 & 12 & 1 & 4 \\ -3 & -46 & 29 & -2 & -15 \\ 9 & 10 & -5 & 4 & 7 \end{array}\right) \\
\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & -192 & 123 & -3 & -57 \\ 0 & -64 & 41 & -1 & -19 \end{array}\right) \begin{array}{l} \\ \scriptstyle L_2 \gets L_1-4L_2 \\ \scriptstyle L_3 \gets L_1+4L_3 \\ \scriptstyle L_4 \gets 3L_1-4L_4 \end{array} \\
\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right) \begin{array}{l} \\ \\ \scriptstyle L_3 \gets 3L_2+L_3 \end{array} \\
\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \end{array}\right)
\end{array}
$$

In this tabular, the instructions `\SubMatrix` are executed after the composition of the tabular and, thus, the vertical rules are drawn without adding space between the columns.

In fact, it's possible, with the key `vlines-in-sub-matrix`, to choice a letter in the preamble of the array to specify vertical rules which will be drawn in the `\SubMatrix` only (by adding space between the columns).

```
\setlength{\extrarowheight}{1mm}
\[\begin{NiceArray}
   [
     vlines-in-sub-matrix=I,
     last-col,
     code-for-last-col = \scriptstyle \color{blue}
   ]
  {rrrrIr}
12 &  -8  &  7 & 5 &  3 \\
 3 & -18  & 12 & 1 &  4 \\
-3 & -46  & 29 &-2 &-15 \\
 9 & 10   &-5  &4  & 7 \\[1mm]
12 & -8   & 7  &5  & 3 \\
0  & 64   &-41 & 1 & 19 & L_2 \gets L_1-4L_2  \\
0  & -192 &123 &-3 &-57 & L_3 \gets L_1+4L_3  \\
0  & -64  & 41 &-1 &-19 & L_4 \gets 3L_1-4L_4 \\[1mm]
12 & -8   &7   &5  & 3 \\
0  & 64   &-41 &1  &19 \\
0  &  0   &0   &0  & 0 & L_3 \gets 3L_2+L_3 \\[1mm]
12 & -8   &7   &5  & 3 \\
0  & 64   &-41 & 1 & 19 \\
\CodeAfter
   \SubMatrix({1-1}{4-5})
   \SubMatrix({5-1}{8-5})
   \SubMatrix({9-1}{11-5})
   \SubMatrix({12-1}{13-5})
\end{NiceArray}\]
```

$$\begin{array}{cl}
\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 3 & -18 & 12 & 1 & 4 \\ -3 & -46 & 29 & -2 & -15 \\ 9 & 10 & -5 & 4 & 7 \end{array}\right) & \\
\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & -192 & 123 & -3 & -57 \\ 0 & -64 & 41 & -1 & -19 \end{array}\right) & \begin{array}{l} \\ \scriptstyle L_2 \gets L_1-4L_2 \\ \scriptstyle L_3 \gets L_1+4L_3 \\ \scriptstyle L_4 \gets 3L_1-4L_4 \end{array} \\
\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right) & \begin{array}{l} \\ \\ \scriptstyle L_3 \gets 3L_2+L_3 \end{array} \\
\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \end{array}\right) &
\end{array}$$

18.8 How to highlight cells of a matrix

In order to highlight a cell of a matrix, it's possible to "draw" that cell with the key `draw` of the command `\Block` (this is one of the uses of a mono-cell block[89]).

[89] We recall that, if the first mandatory argument of the command `\Block` is left empty, that means that the block is a mono-cell block

```
$\begin{pNiceArray}{>{\strut}cccc}[margin,rules/color=blue,no-cell-nodes]
\Block[draw]{}{a_{11}} & a_{12} & a_{13} & a_{14} \\
a_{21} & \Block[draw]{}{a_{22}} & a_{23} & a_{24} \\
a_{31} & a_{32} & \Block[draw]{}{a_{33}} & a_{34} \\
a_{41} & a_{42} & a_{43} & \Block[draw]{}{a_{44}} \\
\end{pNiceArray}$
```

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

We should remark that the rules we have drawn are drawn *after* the construction of the array and thus, they don't spread the cells of the array. We recall that, on the other side, the commands `\hline` and `\Hline`, the specifier "`|`" and the options `hlines`, `vlines`, `hvlines` and `hvlines-except-borders` spread the cells.[90]

It's possible to color a row with `\rowcolor` in the `code-before` (or with `\rowcolor` in the first cell of the row.

```
\begin{pNiceArray}{>{\strut}cccc}% <-- % mandatory
   [margin, extra-margin=2pt,no-cell-nodes]
  \rowcolor{red!15}A_{11} & A_{12} & A_{13} & A_{14} \\
  A_{21} & \rowcolor{red!15}A_{22} & A_{23} & A_{24} \\
  A_{31} & A_{32} & \rowcolor{red!15}A_{33} & A_{34} \\
  A_{41} & A_{42} & A_{43} & \rowcolor{red!15}A_{44}
\end{pNiceArray}
```

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

However, it's not possible to do a fine tuning. That's why we describe now a method to highlight a row of the matrix.

That example and the following ones require TikZ (by default, nicematrix only loads PGF, which is a sub-layer of TikZ) and the TikZ library `fit`. The following instructions in the preamble of your document do the job:

```
\usepackage{tikz}
\usetikzlibrary{fit}
```

We create a rectangular TikZ node which encompasses the nodes of the second row by using the tools of the TikZ library fit. Those nodes are not available by default in the `\CodeBefore` (for efficiency). We have to require their creation with the key `create-cell-nodes` of the keyword `\CodeBefore`.

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

$\begin{bNiceMatrix}
\CodeBefore [create-cell-nodes]
  \tikz \node [highlight = (2-1) (2-last)] {} ;
```

[90]For the command `\cline`, see the remark p. 12.

```
\Body
0 & \Cdots & 0 \\
1 & \Cdots & 1 \\
0 & \Cdots & 0 \\
\end{bNiceMatrix}$
```

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

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

```
\begin{pNiceArray}{ccc}[last-col, margin = 2pt]
\CodeBefore [create-cell-nodes]
  \begin{tikzpicture}
  \node [highlight = (1-1) (1-last)] {} ;
  \node [highlight = (2-1) (2-last)] {} ;
  \node [highlight = (3-1) (3-last)] {} ;
  \end{tikzpicture}
\Body
a & a + b & a + b + c & L_1 \\
a & a     & a + b     & L_2 \\
a & a     & a         & L_3
\end{pNiceArray}
```

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

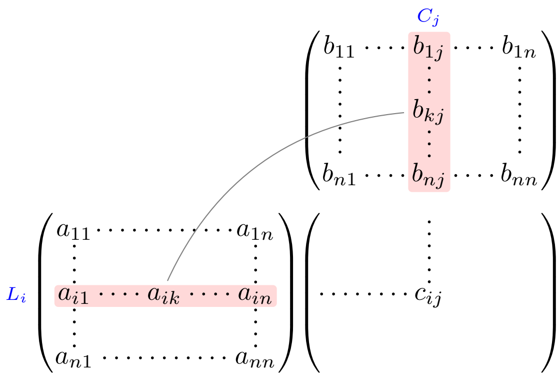
The result may seem disappointing. We can improve it by using the "medium nodes" instead of the "normal nodes".

```
\[\begin{pNiceArray}{ccc}[last-col, margin = 2pt, create-medium-nodes]
\CodeBefore [create-cell-nodes]
  \begin{tikzpicture} [name suffix = -medium]
  \node [highlight = (1-1) (1-last)] {} ;
  \node [highlight = (2-1) (2-last)] {} ;
  \node [highlight = (3-1) (3-last)] {} ;
  \end{tikzpicture}
\Body
a & a + b & a + b + c & L_1 \\
a & a     & a + b     & L_2 \\
a & a     & a         & L_3
\end{pNiceArray}\]
```

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

18.9 Utilisation of \SubMatrix in the \CodeBefore

In the following example, we illustrate the mathematical product of two matrices.
The whole figure is an environment `{NiceArray}` and the three pairs of parenthesis have been added with `\SubMatrix` in the `\CodeBefore`.




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

```
\[\begin{NiceArray}{*{6}{c}@{\hspace{6mm}}*{5}{c}}[nullify-dots]
\CodeBefore [create-cell-nodes]
  \SubMatrix({2-7}{6-last})
  \SubMatrix({7-2}{last-6})
  \SubMatrix({7-7}{last-last})
  \begin{tikzpicture}
    \node [highlight = (9-2) (9-6)] { } ;
    \node [highlight = (2-9) (6-9)] { } ;
  \end{tikzpicture}
\Body
    &        &        &        &        &        &        & \color{blue}\scriptstyle C_j \\
    &        &        &        &        & b_{11} & \Cdots & b_{1j} & \Cdots & b_{1n} \\
    &        &        &        &        & \Vdots &        & \Vdots &        & \Vdots \\
    &        &        &        &        &        &        & b_{kj} \\
    &        &        &        &        &        &        & \Vdots \\
    &        &        &        &        & b_{n1} & \Cdots & b_{nj} & \Cdots & b_{nn} \\[3mm]
    & a_{11} & \Cdots &        &        & a_{1n} \\
    & \Vdots &        &        &        & \Vdots &        &        & \Vdots \\
\color{blue}\scriptstyle L_i
    & a_{i1} & \Cdots & a_{ik} & \Cdots & a_{in} & \Cdots &        & c_{ij} \\
    & \Vdots &        &        &        & \Vdots \\
    & a_{n1} & \Cdots &        &        & a_{nn}  \\
\CodeAfter
\tikz \draw [gray,shorten > = 1mm, shorten < = 1mm] (9-4.north) to [bend left] (4-9.west) ;
\end{NiceArray}\]
```

18.10 A triangular tabular

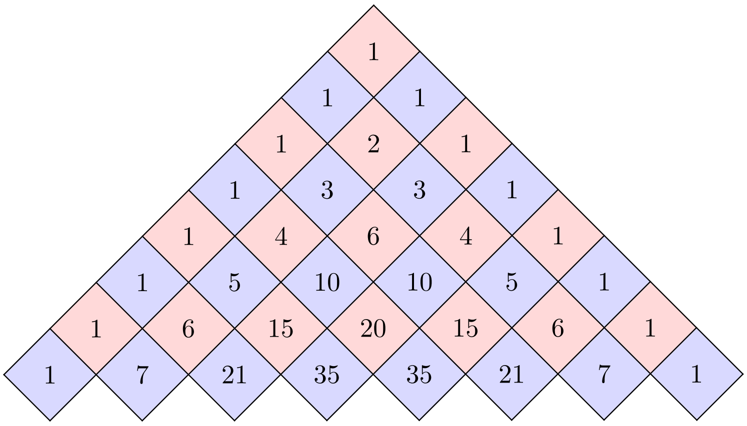
In the following example, we use the style PGF/TikZ `nicematrix/cell-node` to rotate the contents of the cells (and, then, we compensate that rotation by a rotation of the whole tabular with the command `\adjustbox` of the eponymous package, which must be loaded previously).

```
\pgfset
  {
    nicematrix/cell-node/.append style =
      { text/rotate = 45, minimum size = 6 mm }
  }

\setlength{\tabcolsep}{0pt}

\adjustbox{rotate = -45, set depth = 6mm + 1.414 \arrayrulewidth}
```

```
{\begin{NiceTabular} [ hvlines, corners=SE, baseline = line-9 ] { cccccccc }
 \CodeBefore
   \chessboardcolors{red!15}{blue!15}
 \Body
   1 & 1 & 1  & 1  & 1  & 1 & 1 & 1 \\
   1 & 2 & 3  & 4  & 5  & 6 & 7 \\
   1 & 3 & 6  & 10 & 15 & 21 \\
   1 & 4 & 10 & 20 & 35 \\
   1 & 5 & 15 & 35 \\
   1 & 6 & 21 \\
   1 & 7 \\
   1
 \end{NiceTabular}}
```



19 History

The development of the package nicematrix is done in the following GitHub depot:
https://github.com/fpantigny/nicematrix

The successive versions of the file nicematrix.sty provided by TeXLive are also available on the SVN server of TeXLive:

https:www.tug.org/svn/texlive/trunk/Master/texmf-dist/tex/latex/nicematrix/nicematrix.sty

Changes between version 7.0 and 7.1

New commands \Hbrace and \Vbrace.
New commands \EmptyColumn and \EmptyRow in the \CodeBefore.

Changes between version 6.29 and 7.0

The package nicematrix is now compatible (for the tabulars) with the Tagging Project.

Changes between version 6.28 and 6.29

Modification in order to be compatible with the next version of array (v. 2.6g).

Changes between version 6.27 and 6.28

Sub-blocks with the character & (when the key ampersand-in-block is in force).
Keys p and j for the command \Block.
PGF nodes i.1, i.2, i.3, etc.

Changes between version 6.26 and 6.27

New key light-syntax-expanded.

Changes between version 6.25 and 6.26

Special color nocolor.

Changes between version 6.24 and 6.25

Correction of several bugs.
An instruction \rowlistcolors in the tabular stops the effect of a previous \rowlistcolors.

Changes between version 6.23 and 6.24

New command \TikzEveryCell available in the \CodeAfter and the \CodeBefore to apply the same TikZ instruction to all the cells and blocks of the array.
New key offset in the key tikz of a command \Block.

Changes between version 6.22 and 6.23

The specifier "|" in the preambles of the environments has now an optional argument.
Correction of several bugs.

Changes between version 6.21 and 6.22

Key `opacity` for the command `\Block`.
It's now possible to put a label on a "continuous dotted line" with the specifier "`:`".

Changes between version 6.20a and 6.21

Key `c` for the command `\tabularnote`.
New commands `\rowcolors` and `\rowlistcolors` available in the array itself (previously, there were only two commands `\rowcolors` and `\rowlistcolors` available in the `\CodeBefore`.

Changes between version 6.20 and 6.20a

The value of the key `name` of an environment of `nicematrix` is expanded (in the TeX sens).
The labels of the tabular notes (created by the command `\tabularnote`) are now composed in an overlapping position *only* when the horizontal alignment mode of the column is `c` or `r`.

Changes between version 6.19 and 6.20

New key `xdots/horizontal-labels`.

Changes between version 6.18 and 6.19

The command `\tabularnote` now supports an optional argument (between square brackets) to change the symbol of the reference of the note.

Changes between version 6.17 and 6.18

New key `opacity` in the commands to color cells, rows and columns.
New key `rounded-corners` for a whole tabular.

Changes between version 6.16 and 6.17

New PGF/TikZ style `nicematrix/cell-node`.
New key `pgf-node-code`

Changes between version 6.15 and 6.16

It's now possible to put any LaTeX extensible delimiter (`\lgroup`, `\langle`, etc.) in the preamble of an environment with preamble (such as `{NiceArray}`) by prefixing them by `\left` and `\right`.
New key `code` for the command `\SubMatrix` in the `\CodeAfter`.

Changes between version 6.14 and 6.15

New key `transparent` for the command `\Block` (with that key, the rules are drawn within the block).

Changes between version 6.13 and 6.14

New keys for the command `\Block` for the vertical position of the content of that block.

Changes between version 6.12 and 6.13

New environment {TabularNote} in {NiceTabular} with the same semantic as the key tabularnote (for legibility).
The command \Hline nows accepts options (between square brackets).

Changes between version 6.11 and 6.12

New keys caption, short-caption and label in the environment {NiceTabular}.
In {NiceTabular}, a caption specified by the key caption is wrapped to the width of the tabular.
Correction of a bug: it's now possible to use \OverBrace and \UnderBrace with unicode-math (with XeLaTeX or LuaLaTeX).

Changes between version 6.10 and 6.11

New key matrix/columns-type to specify the type of columns of the matrices.
New key ccommand in custom-line and new command \cdottedline.

Changes between version 6.9 and 6.10

New keys xdots/shorten-start and xdots/shorten-end.
It's possible to use \line in the \CodeAfter between two blocks (and not only two cells).

Changes between version 6.8 and 6.9

New keys xdots/radius and xdots/inter for customisation of the continuous dotted lines.
New command \ShowCellNames available in the \CodeBefore and in the \CodeAfter.

Changes between version 6.7 and 6.8

In the notes of a tabular (with the command \tabularnote), the duplicates are now detected: when several commands \tabularnote are used with the same argument, only one note is created at the end of the tabular (but all the labels are present, of course).

Changes between version 6.6 and 6.7

Key color for \OverBrace and \UnderBrace in the \CodeAfter
Key tikz in the key borders of a command \Block

Changes between version 6.5 and 6.6

Keys tikz and width in custom-line.

Changes between versions 6.4 and 6.5

Key custom-line in \NiceMatrixOptions.
Key respect-arraystretch.

Changes between versions 6.3 and 6.4

New commands \UnderBrace and \OverBrace in the \CodeAfter.
Correction of a bug of the key baseline (cf. question 623258 on TeX StackExchange).
Correction of a bug with the columns V of varwidth.
Correction of a bug: the use of \hdottedline and : in the preamble of the array (or another letter specified by letter-for-dotted-lines) was incompatible with the key xdots/line-style.

Changes between versions 6.2 and 6.3

Keys `nb-rows`, `rowcolor` and `bold` for the command `\RowStyle`
Key `name` for the command `\Block`.
Support for the columns `V` of varwidth.

Changes between versions 6.1 and 6.2

Better compatibility with the classes revtex4-1 (deprecated) and revtex4-2.
Key `vlines-in-sub-matrix`.

Changes between versions 6.0 and 6.1

Better computation of the widths of the `X` columns.
Key `\color` for the command `\RowStyle`.

Index

Contents