Abstract

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

The package \texttt{nicematrix} is entirely contained in the file \texttt{nicematrix.sty}. This file may be put in the current directory or in a \texttt{texmf} tree. However, the best is to install \texttt{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 \texttt{nicematrix.sty} in the repository of your project in order to take full advantage of the latest version de \texttt{nicematrix}.

This package can be used with \texttt{xelatex}, \texttt{lualatex}, \texttt{pdflatex} but also by the classical workflow \texttt{latex-dvips-ps2pdf} (or Adobe Distiller). \textit{However, the file nicematrix.dtx of the present documentation should be compiled with XeLaTeX.}

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

The idea of \texttt{nicematrix} is to create PGF nodes under the cells and the positions of the rules of the tabular created by \texttt{array} and to use these nodes to develop new features. As usual with PGF, the coordinates of these nodes are written in the \texttt{aux} to be used on the next compilation and that’s why \texttt{nicematrix} may need \texttt{several compilations}.

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

A command \texttt{\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).

\begin{tabular}{ccc}
 \textbf{Product} & \textbf{dimensions (cm)} & \textbf{Price} \\
 \hline
 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 \\
\end{tabular}
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.\footnote{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. 24.}

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}

Inspired by the package cellspace which deals with that problem, the package nicematrix provides two keys \texttt{cell-space-top-limit} and \texttt{cell-space-bottom-limit} similar to the parameters of cellspace called \texttt{cellspacetoplimit} and \texttt{cellspacebottomlimit}.

There is also a key \texttt{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 \texttt{NiceMatrixOptions}.\footnote{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.}

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

\begin{pmatrix}
\frac{12}{3} & -\frac{12}{3} \\ \\
\frac{13}{4} & \frac{14}{4}
\end{pmatrix}
3 The vertical position of the arrays

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

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

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

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

In the following example, we use the option \texttt{t} (equivalent to \texttt{baseline=t}) immediately after an \texttt{item} of list. One should remark that the presence of a \texttt{\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 \texttt{array}, one must use \texttt{\firsthline}).

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

1. an item

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

1. an item

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

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

\[
A=\begin{pNiceMatrix}[baseline=line-3]
\dfrac{1}{A} & \dfrac{1}{B} & 0 & 0 \\
\dfrac{1}{A} & \dfrac{1}{B} & 0 & 0 \\
\dfrac{1}{C} & \dfrac{1}{D} & 0 & 0 \\
0 & 0 & A & B \\
0 & 0 & D & D
\end{pNiceMatrix}
\]

\[A=\begin{pNiceMatrix}
\dfrac{1}{A} & \dfrac{1}{B} & 0 & 0 \\
\dfrac{1}{A} & \dfrac{1}{B} & 0 & 0 \\
\dfrac{1}{C} & \dfrac{1}{D} & 0 & 0 \\
0 & 0 & A & B \\
0 & 0 & D & D
\end{pNiceMatrix}\]

\footnote{The extension \texttt{booktabs} is \textit{not} loaded by \texttt{nicematrix}.}
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.\footnote{The spaces after a command \Block are deleted.}
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. It’s possible to use \\ in that content to
  have a content on several lines. 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}

|  \Block{3-3}{A} & & & 0  |
| \hline |
| 0 & Vdots \| |
| k & k & 0 \|
| k & k & 0 \|
| hline |
| 0 & \|Cdots & 0 & 0 |
\end{bNiceArray}

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.\footnote{This argument
between angular brackets may also be used to insert a command of font such as \bfseries when
the command \A{} 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 \ref{rot}, p. \pageref{rot}).}

\begin{bNiceArray}{cw{c}{1cm}c|c}

|  A & 0  |
| \hline |
| 0 & \|dots \| |
| k & k & 0 \|
| k & k & 0 \|
| hline |
| 0 & \|Cdots & 0 & 0 |
\end{bNiceArray}

In fact, the command \Block accepts as first optional argument (between square brackets) a list of
couples key=value.

First, there are keys which are quick tools to control the appearance of the block.

- the key \texttt{fill} takes in as value a color and fills the block with that color;

- the key \texttt{opacity} sets the opacity of the filling color specified by \texttt{fill};

- the key \texttt{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);

\begin{bNiceArray}{cw{c}{1cm}c|c}

|  \Block{3-3}<\Large>{A} & & & 0  |
| \hline |
| 0 & Vdots \| |
| k & k & 0 \|
| k & k & 0 \|
| hline |
| 0 & \|Cdots & 0 & 0 |
\end{bNiceArray}
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;\(^8\)

- 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\(^9\)).

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 \texttt{\usepackage{tikz}}). By default, \texttt{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 \texttt{left}, \texttt{right}, \texttt{top} and \texttt{bottom}; it’s possible, in fact, in the list which is the value of the key **borders**, to add an entry of the form \texttt{tikz={list}} where \texttt{list} is a list of couples \texttt{key=value} of TikZ specifying the graphical characteristics of the lines that will be drawn (for an example, see p. 62).

- When the key **tikz** is used, the TikZ path corresponding of the rectangle which delimits the block is executed with TikZ\(^10\) by using as options the value of that key **tikz** (which must be a list of keys allowed for a TikZ path).

  **New 6.24** 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 \texttt{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 \texttt{\Block}.

  For examples, cf. p. 56.

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 \texttt{\CodeAfter} of the environment (cf. p. 24);

- the key **respect-arraystretch** prevents the setting of \texttt{\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. 9). However, if the key **transparent** is used, the rules are drawn. For an example, see section 18.1 on page 56. Caution: that key does not imply that the content of the block will be transparent!

There is also keys for the horizontal and vertical positions of the content of the block: cf. 4.3 p. 8.

**One must remark that, by default, the commands \texttt{\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 \texttt{w[c]{...}} of \texttt{array}).

---

\(^8\)However, the rules are not drawn in the sub-blocks of the block, as always with \texttt{nicematrix}: the rules are not drawn in the blocks, except when they have the key **transparent** (cf. section 5 p. 9).

\(^9\)This value is the initial value of the **rounded corners** of TikZ.

\(^10\)TikZ should be loaded (by default, \texttt{nicematrix} only loads PGF) and, if it’s not, an error will be raised.
\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 \&
iris & & \& lis \&
arum & periwinkle & forget-me-not & hyacinth
\end{NiceTabular}

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 \texttt{w(...)}\ldots, \texttt{W(...)}\ldots, \texttt{p(...)}, \texttt{b(...)}, \texttt{m(...)}, \texttt{V} and \texttt{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 (\texttt{c}, \texttt{r} or \texttt{l}) is taken into account for the blocks (but the \Block may have its own specification of alignment: cf. \texttt{4.5 p. 7}).

- The specifications of font specified for the column by a construction >\texttt{...} 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 \\
& 13 \hline
\Block{3-1}{Sarah} & 18 \\
& 17 \\
& 15 \hline
Ashley & 20 \\
& 19 \\
Henry & 14 \\
& 15 \\
\Block{2-1}{Madison} & 15 \\
& 19 \hline
\end{NiceTabular}

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 \texttt{\multicolumn} of \LaTeX{}), except when an option of vertical position has been used for the block (one of the keys \texttt{t}, \texttt{b}, \texttt{v-center}, \texttt{T} and \texttt{B} described in the part \texttt{4.6 p. 8}).

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`.
- 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}

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

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{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}

\begin{NiceTabular}{cc}
\toprule
Writer & year of birth \\
\midrule
Hugo & 1802 \\
Balzac & 1799 \\
\bottomrule
\end{NiceTabular}

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

By default, the horizontal position of the content of a block is computed by using the positions of the \textit{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`).

11If 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` (when the key `color-inside` is used).

12One may consider that the default value of the first mandatory argument of `\Block` is 1-1.
In order to have an horizontal positionning 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.

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}

4.6 Vertical position of the content of the block

For the vertical position, the command \Blocks accepts the keys v-center\textsuperscript{13}, t, b, T and B.

- With the key v-center, 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 contains several lines 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.

\textbf{Modification 6.18} No vertical margin is added. However, the contents of the block is (al-
ways) composed by \nicematrix in a \{minipage\}, a \{tabular\} or an \{array\} and, hence, there
will still remain a margin (in most cases).
- With the key B, the content of the block is set downwards.

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

\begin{NiceTabular}{ccc}
\Block[fill=red!10,t,1]{4-2}{two\{lines}
& & \Huge Un\\
& & deux \\
& & trois \\
& & \Huge quatre \\
text & text \\
\end{NiceTabular}

\textsuperscript{13}That key could not have been named c since the key c is used for the horizontal alignment.
\begin{NiceTabular}{ccc}
\Block[fill=red!10,b,r]{4-2}{two\ lines}
 & & \Huge Un\\
 & & deux \\
 & & trois \\
 & & \Huge quatre \\
text & text \\
\end{NiceTabular}

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

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

5 The rules

The usual techniques for the rules may be used in the environments of \texttt{nicematrix} (except \texttt{\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 \texttt{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 \texttt{\hline\hline} (there is no need to use the package \texttt{hhline}).

\begin{NiceTabular}{|c|c|c|c|} \toprule
a & b & c & d \\
\midrule
1 & 2 & 3 & 4 \\
\bottomrule
\end{NiceTabular}

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

If you use \texttt{booktabs} (which provides \texttt{\toprule}, \texttt{\midrule}, \texttt{\bottomrule}, etc.) and if you really want to add vertical rules (which is not in the spirit of \texttt{booktabs}), you should notice that the vertical rules drawn by \texttt{nicematrix} are compatible with \texttt{booktabs}.

$\begin{NiceArray}{|cccc|} \toprule
a & b & c & d \\
\midrule
1 & 2 & 3 & 4 \\
\bottomrule
\end{NiceArray}$
However, it’s still possible to define a specifier (named, for instance, I) to draw vertical rules with the standard behaviour of \texttt{array}.

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

\subsection{The command \texttt{\textbackslash cline}}

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

For historical reasons, this is not the case with the command \texttt{\textbackslash cline}, as shown by the following example.

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

In the environments of \texttt{nicematrix}, this situation is corrected (it’s still possible to go to the standard behaviour of \texttt{\textbackslash cline} with the key \texttt{standard-cline}).

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

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

\subsection{The thickness and the color of the rules}

The environments of \texttt{nicematrix} provide a key \texttt{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 \texttt{\arrayrulewidth}.

It’s well known that \texttt{colortbl} provides the command \texttt{\textbackslash arrayrulecolor} in order to specify the color of the rules.

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

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

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

\subsection{The tools of \texttt{nicematrix} for the rules}

Here are the tools provided by \texttt{nicematrix} for the rules.

- the keys \texttt{hlines}, \texttt{vlines}, \texttt{hvlines} and \texttt{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\footnote{And also the command \multicolumn but it’s recommended to use instead \Block in the environments of nicematrix.} presented p. 5.
  – the blocks implicitly delimited by the continuous dotted lines created by \Cdots, \Vdots, etc. (cf. p. 27).
• The corners are created by the key corners explained below (see p. 12).
• For the exterior rows and columns, see p. 26.

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. 13.\footnote{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: \newcommand{\RedLine}{\Hline[color=red]}\footnote{It’s possible to put in that list some intervals of integers with the syntax $i$-$j$.}}

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}

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.\footnote{It’s possible to put in that list some intervals of integers with the syntax $i$-$j$.}

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

5.3.2 The keys hvlines and hvlines-except-borders

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

\begin{NiceTabular}{cccc}[hvlines,\text{rules/color=blue}]
rose & tulipe & marguerite & dahlia \ \ \\
violette & \Block\[draw=red\]{2-2}{\LARGE fleurs} & & souci \ \\
pervenche & & lys \ \\
arum & iris & jacinthe & muguet
\end{NiceTabular}
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. 44.

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. 57.

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. However, it’s possible, for a cell without content, to require nicematrix to consider that cell as not empty with the key \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.

When the key `corners` 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).

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

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

\footnote{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. 53).}

\footnote{The key `corners` that we describe now has no direct link with the key `rounded-corners` described in the part 14.1, p. 44.}
\begin{NiceMatrixOptions}{cell-space-top-limit=3pt}
\begin{NiceTabular}{*{6}{c}}
\hline
1 \\
1 & 1 \\
1 & 2 & 1 \\
1 & 3 & 3 & 1 \\
1 & 4 & 6 & 4 & 1 \\
1 \\
\hline
\end{NiceTabular}
\end{NiceMatrixOptions}

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. 16). The command \TikzEveryCell available in the \CodeBefore and the \CodeAfter (cf. p. 38) takes also into account the empty corners.

5.3.4 The command \texttt{\diagbox}

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

\begin{NiceArray}{*{5}{c}}
\hline
\diagbox{x}{y} & e & a & b & c \\
e & e & a & b & c \\
a & a & c & b \\
b & b & c & e & a \\
c & c & b & a & e \\
\hline
\end{NiceArray}

It’s possible to use the command \texttt{\diagbox} in a \texttt{\Block}.

5.3.5 Commands for customized rules

It’s also possible to define commands and letters for customized rules with the key \texttt{custom-line} available in \NiceMatrixOptions and in the options of individual environments. That key takes in as argument a list of \texttt{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 \texttt{command} is the name (without the backslash) of a command that will be created by \texttt{nicematrix} and that will be available for the final user in order to draw horizontal rules (similarly to \texttt{\hline});
- the key \texttt{ccommand} is the name (without the backslash) of a command that will be created by \texttt{nicematrix} and that will be available for the final user to order to draw partial horizontal rules (similarly to \texttt{\cline}, hence the name \texttt{ccommand}): the argument of that command is a list of intervals of columns specified by the syntax \texttt{i} or \texttt{\textbackslash i-j}.
- the key \texttt{letter} takes in as argument a letter that the user will use in the preamble of an environment with preamble (such as \texttt{\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 \texttt{\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 \texttt{colortbl} for instance).

\footnote{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.}

\footnote{The following letters are forbidden: \texttt{icrmbVX!()[]@<>}}
– 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 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).

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).

\begin{NiceTabular}{lcIcIc}
\customline = {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}

• 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.

\begin{NiceMatrixOptions}
\customline =
\begin{NiceTabular}{cIcIc}
\customline = {letter=I, color=blue}
\hline
one & two & three \ 
four & five & six \ 
seven & eight & nine \ 
\end{NiceTabular}
• **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. \[27\]).

In fact, nicematrix defines by default the commands \hdottedline and \cdottedline and the letter “:” for those dotted rules.\[21\]

\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}

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

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

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

Since the package nicematrix is based on array, it’s possible to use colortbl with nicematrix. However, there is two drawbacks:

• The package colortbl patches array, leading to some incompatibilities (for instance with the command \hdotsfor).

\[21\] However, it’s possible to overwrite those definitions with a *custom-line* (in order, for example, to switch to dashed lines).
The package `colortbl` constructs the array row by row, alternating 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 \`\rowcolor` or \`\colortbl` is used.

As for this phenomenon, Adobe Reader gives better results than MuPDF.

The package `nicematrix` provides tools to avoid those problems.

### 6.2 The tools of `nicematrix` in the `\CodeBefore`

The package `nicematrix` provides some tools (independent of `colortbl`) 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.\(^{22}\)

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.

```latex
\begin{pNiceArray}{preamble}
\CodeBefore [options]
  \text{instructions of the code-before}
\Body
  \text{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.\(^{23}\)

New commands are available in that `\CodeBefore`: `\cellcolor`, `\rectanglecolor`, `\rowcolor`, `\columncolor`, `\rowcolors`, `\rowlistcolors`, `\chessboardcolors` and `\arraycolor`.\(^{24}\)

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. \(^{12}\).

These commands respect the rounded corners if the key `rounded-corners` (described in the part \(^{14.1}\) at the page \(^{44}\)) 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`.\(^{25}\)

\(^{22}\)If you use Overleaf, Overleaf will do automatically a sufficient number of compilations.

\(^{23}\)The available keys are `create-cell-nodes`, `sub-matrix` (and its subkeys) and `delimiters-color`.

\(^{24}\)Remark 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. \(^{14}\).

\(^{25}\)Caution: that feature creates instructions of transparency in the PDF and some PDF don’t support such instructions.
We describe now in detail those commands.

- The command \texttt{cellcolor} takes its name from the command \texttt{cellcolor} of \texttt{colortbl}.
  This command takes in as mandatory arguments a color and a list of cells, each of which with
  the format \texttt{i-j} where \texttt{i} is the number of the row and \texttt{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 \texttt{i-})
  or a whole column (with the syntax \texttt{-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}

- The command \texttt{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}

- The command \texttt{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. 26). It’s only a
  particular case of \texttt{rectanglecolor}.

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

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

  We have used the key \texttt{r} which aligns all the columns rightwards (cf. p. 45).

- The command \texttt{rowcolor} takes its name from the command \texttt{rowcolor} of \texttt{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 \texttt{a-b} (an interval of the form \texttt{a-} represent all the rows from the row \texttt{a} until
  the end).
The command \texttt{\rowcolors} takes its name from the command \texttt{\columncolor} of \texttt{colortbl}. Its syntax is similar to the syntax of \texttt{\rowcolor}. 

The command \texttt{\rowcolors} (with a \texttt{s}) takes its name from the command \texttt{\rowcolors} of \texttt{colortbl}. The \texttt{s} emphasizes the fact that there is \texttt{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.

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

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

- The key \texttt{cols} describes a set of columns. The command \texttt{\rowcolors} will color only the cells of these columns. The value is a comma-separated list of intervals of the form \texttt{i-j} (where \texttt{i} or \texttt{j} may be replaced by \texttt{*}).
- With the key \texttt{restart}, each interval of rows (specified by the first mandatory argument) begins with the same color.
- With the key \texttt{respect-blocks} the “rows” alternately colored may extend over several rows if they have to incorporate blocks (created with the command \texttt{\Block}: cf. p. \texttt{4}).
\begin{NiceTabular}{lr}[hvlines]
\CodeBefore
\rowcolors{1}{blue!10}\{[respect-blocks]
\Body
\Block{2-1}{John} & 12 \ \\
& 13 \ \\
& 12 \ \\
Steph & 8 \\
\Block{3-1}{Sarah} & 18 \ \\
& 17 \ \\
& 15 \ \\
& 18 \ \\
Ashley & 20 \\
Henry & 14 \\
\Block{2-1}{Madison} & 15 \ \\
& 19 \ \\
\end{NiceTabular}

- The extension nicematrix provides also a command \texttt{\rowlistcolors}. This command generalises the command \texttt{\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 \texttt{=} 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}

It’s also possible to use in the command \texttt{\rowlistcolors} a color series defined by the command \texttt{\definecolorseries} of \texttt{xcolor} (and initialized with the command \texttt{\resetcolorseries}\texttt{27}).

\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}

We recall that all the color commands we have described don’t color the cells which are in the

\footnote{For the initialization, in the following example, you have used the counter \texttt{iRow} which, when used in the \texttt{\CodeBefore} (and in the \texttt{\CodeAfter}) corresponds to the number of rows of the array: cf. p\textsuperscript{12}. That leads to an adjustment of the gradation of the colors to the size of the tabular.}
“corners”. In the following example, we use the key \texttt{corners} to require the determination of the corner \textit{north east} (NE).

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

The previous example uses the keys \texttt{first-row} and \texttt{first-col} which are described in the chapter concerning the «exterior» rows and columns (cf. p. 20).
As one can see, \textit{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 explicitely 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 \texttt{first-col} has been used).

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

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

20
6.3. Color tools to be used inside the tabular

It is possible to access the preceding tools with a syntax close to the syntax of colorbl. For that, one must use the key color-inside in the current environment.

There are several commands available (the first three ones are inspired by colortbl but are independent of colorbl):

- \cellcolor which colorizes a cell;
- \rowcolor which must be used in a cell and which colorizes the end of the row;
- \columncolor which must be used in the preamble of the environment with the same syntax as the corresponding command of colorbl. However, this command \columncolor can appear within another command, itself used in the preamble of the array;

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

\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}

We have used the type of column S of siunitx.
Vanessa & Stephany & 30 octobre 1994 \ 
Dupont & Chantal & 15 janvier 1998 \[bottomrule
\end{NiceTabular}

<table>
<thead>
<tr>
<th>Last name</th>
<th>First name</th>
<th>Birth day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Achard</td>
<td>Jacques</td>
<td>5 juin 1962</td>
</tr>
<tr>
<td>Lefebvre</td>
<td>Mathilde</td>
<td>23 mai 1988</td>
</tr>
<tr>
<td>Vanesse</td>
<td>Stephany</td>
<td>30 octobre 1994</td>
</tr>
<tr>
<td>Dupont</td>
<td>Chantal</td>
<td>15 janvier 1998</td>
</tr>
</tbody>
</table>

New 6.25 Each use of the command \rowlistcolors (or \rowcolors, which is, in fact, a special case of \rowlistcolors) stops the potential coloring schemes\(^{33}\) 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.

\begin{NiceTabular}{c}[hvlines,color-inside]
one \ \t\two \ \t\rowlistcolors{red!15}
three \ \t\four \ \t\five \ \t\rowlistcolors{}
six \ \t\seven \ \t\end{NiceTabular}

6.4 The special color “nocolor”

New 6.26 The extension nicematrix provides the special color nocolor which may be used in all the coloring commands provided by nicematrix (in the \CodeBefore or 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 \texttt{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 \texttt{cell-space-top-limit}, \texttt{cell-space-bottom-limit} and \texttt{cell-space-limits} are available with the same meaning that the corresponding global keys (cf. p. \[2]

\(^{33}\)We say schemes in plural form because it’s possible to start simultaneously several coloring schemes if they apply on different columns.

22
The key \texttt{rowcolor} sets the color of the background and the key \texttt{color} sets the color of the text.\textsuperscript{34}

The key \texttt{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,rowcolor=blue!50,color=white]{\sffamily}
1 & 2 & 3 & 4 \\
I & II & III & IV
\end{NiceTabular}

The command \texttt{rotate} is described p. \textsuperscript{45}

\section{The width of the columns}

\subsection{Basic tools}

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

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

In the environments of \texttt{nicematrix}, it’s also possible to fix the minimal width of all the columns (except the potential exterior columns: cf. p. \textsuperscript{26}) directly with the key \texttt{columns-width}.

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

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

It’s possible to give the special value \texttt{auto} to the option \texttt{columns-width}: all the columns of the array will have a width equal to the widest cell of the array.\textsuperscript{35}

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

\textsuperscript{34}The key \texttt{color} uses the command \texttt{\color} but inserts also an instruction \texttt{\leavevmode} before. This instruction prevents a extra vertical space in the cells which belong to columns of type \texttt{p}, \texttt{b}, \texttt{m}, \texttt{X} and \texttt{V} (which start in vertical mode of LaTeX).

\textsuperscript{35}The result is achieved with only one compilation (but PGF/TikZ will have written informations in the \texttt{aux} file and a message requiring a second compilation will appear).
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}$

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\cite{nicematrix-block}. The environment \NiceMatrixBlock has no direct link with the command \Block presented previously in this document (cf. p. 4).

\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}

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\cite{tabu} and tabularx, 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 integer 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).\footnote{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.}

- 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).

- 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 default value is t.

\begin{NiceTabular}[width=9cm]{X[2,1]X[1]}[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}

\cite{nicematrix-block}At this time, this is the only usage of the environment \NiceMatrixBlock but it may have other usages in the future.

\cite{tabu}The extension tabu is now considered as deprecated.
8.3 The columns V of \texttt{varwidth}

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

That point is illustrated on the following examples.

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

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

The package \texttt{varwidth} provides also the column type \texttt{V}. A column of type \texttt{V\{\langle\texttt{dim}\rangle\}} encapsulates all its cells in a \texttt{\{varwidth\}} with the argument \texttt{\langle\texttt{dim}\rangle} (and does also some tuning).

When the package \texttt{varwidth} is loaded, the columns \texttt{V} of \texttt{varwidth} are supported by \texttt{nicematrix}.

\begin{verbatim}
\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}
\end{verbatim}

Concerning \texttt{nicematrix}, one of the interests of this type of columns is that, for a cell of a column of type \texttt{V}, the PGF/TikZ node created by \texttt{nicematrix} for the content of that cell has a width adjusted to the content of the cell : cf. p. \pageref{p.24}.

The columns \texttt{V} of \texttt{nicematrix} supports the keys \texttt{t}, \texttt{p}, \texttt{m}, \texttt{b}, \texttt{l}, \texttt{c} and \texttt{r} also supported by the columns \texttt{X}: see their description in the section \texttt{8.2}, p. \pageref{p.24}.

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

The dotted lines have been drawn with the tools presented p. 27.

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.39

- 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. 17) 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.

\footnote{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. 17).}
\NiceMatrixOptions{code-for-first-row = \color{red},
   code-for-first-col = \color{blue},
   code-for-last-row = \color{green},
   code-for-last-col = \color{magenta}}
$\begin{pNiceArray}{cc|cc}[first-row,last-row=5,first-col,last-col=nullify-dots]
   & C_1 & \Cdots & & C_4 & \\
L_1 & a_{11} & a_{12} & a_{13} & a_{14} & L_1 \\
\Vdots & a_{21} & a_{22} & a_{23} & a_{24} & \Vdots \\
\hline \\
& a_{31} & a_{32} & a_{33} & a_{34} & \\
L_4 & a_{41} & a_{42} & a_{43} & a_{44} & L_4 \\
& C_1 & \Cdots & & C_4 & \\
\end{pNiceArray}$

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. 13).

- 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. 23) 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. 35.

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.\footnote{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.}

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\footnote{The precise definition of a “non-empty cell” is given below (cf. p. 53).} 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 \texttt{color}.\footnote{It's also possible to change the color of all these dotted lines with the option \texttt{xdots/color} (\texttt{xdots} to remind that it works for \texttt{\vdots}, \texttt{\ldots}, \texttt{\vdots}, etc.): cf. p. \pageref{xdots_color}.}

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

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

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

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 \texttt{nicematrix}:

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

In the first column of this example, there are two instructions \texttt{\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 & & \\
0 & \Cdots & 0
\end{bNiceMatrix}

There are also other means to change the size of the matrix. Someone might want to use the optional argument of the command \texttt{\vdots} for the vertical dimension and a command \texttt{\hspace*} in a cell for the horizontal dimension.\footnote{In \texttt{nicematrix}, one should use \texttt{\hspace*} and not \texttt{\hspace} for such an usage because \texttt{nicematrix} loads \texttt{array}. One may also remark that it's possible to fix the width of a column by using the environment \texttt{\NiceArray} (or one of its variants) with a column of type \texttt{w} or \texttt{W}: see p. \pageref{nicematrix_w}}

However, a command \texttt{\hspace*} might interfer with the construction of the dotted lines. That's why the package \texttt{nicematrix} provides a command \texttt{\hspace} which is a variant of \texttt{\hspace} transparent for the dotted lines of \texttt{nicematrix}.

\begin{bNiceMatrix}
0 & \Cdots & \hspace*{1cm} & 0 \\
\Vdots & \hspace*{1cm} & \& \hspace*{1cm} \\
0 & \Cdots & \hspace*{1cm} & 0
\end{bNiceMatrix}
10.1 The option nullify-dots

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

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

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

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

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

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

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 \texttt{nullify-dots} (and only one instruction \Ldots is necessary).

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

The option \texttt{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.

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

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

\[
\begin{pNiceMatrix}
1 & 2 & 3 & 4 & 5 \\
\& \Hdotsfor{3} \\
1 & 2 & 3 & 4 & 5 \\
1 & 2 & 3 & 4 & 5
\end{pNiceMatrix}
\]
Remark: Unlike the command \hdotsfor of amsmath, the command \Hdotsfor may be used even when the package colorbl\textsuperscript{44} is loaded (but you might have problem if you use \rowcolor on the same row as \Hdotsfor). However, when using nicematrix, it’s recommended to use the coloring tools provided by nicematrix instead of colorbl.

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}
\begin{array}{cccc}
C[a_1,a_1] & \Cdots & C[a_1,a_n]\\
& & \\[20mm] & C[a_1,a_1^{(p)}] & \Cdots & C[a_1,a_n^{(p)}]
\end{array}
\begin{array}{cccc}
C[a_n,a_1] & \Cdots & C[a_n,a_n]\\
& & \\
C[a_n,a_1^{(p)}] & \Cdots & C[a_n,a_n^{(p)}]
\end{array}
\end{bNiceMatrix}

\\[4pt]
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. \texttt{renew-dots} and \texttt{renew-matrix}.\textsuperscript{45}

- The option \texttt{renew-dots}
  
  With this option, the commands \ldots, \cdots, \vdots, \ddots and \iddots\textsuperscript{40} and \hdotsfor are redefined within the environments provided by nicematrix and behave like \Ldots, \Cdots, \Vdots, \Ddots and \Iddots; the command \ldots (“automatic dots” of amsmath) is also redefined to behave like \Ldots.

- The option \texttt{renew-matrix}
  
  With this option, the environment \texttt{matrix} is redefined and behave like \texttt{NiceMatrix}, and so on for the five variants.

\textsuperscript{44}We recall that when xcolor is loaded with the option table, the package colorbl is loaded.

\textsuperscript{45}The options \texttt{renew-dots}, \texttt{renew-matrix} can be fixed with the command \texttt{NiceMatrixOptions} like the other options. However, they can also be fixed as options of the command \texttt{\usepackage}.
Therefore, with the keys \texttt{renew-dots} and \texttt{renew-matrix}, a classical code gives directly the output of \texttt{nicematrix}.

\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 \texttt{\Ldots}, \texttt{\Cdots}, \texttt{\Vdots}, \texttt{\Ddots}, \texttt{\Iddots}, \texttt{\Hdots} for and \texttt{\Vdots} (and the command \texttt{\line} in the \texttt{\CodeAfter} which is described p. 34) 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 \texttt{\scriptsize}.

The version 6.22 of \texttt{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. 61).

\begin{bNiceMatrix}
1 & \hspace*{1cm} & 0 \\
& & \Ddots^{n \text{ times}} \\
0 & & 1
\end{bNiceMatrix}

With the key \texttt{xdots/horizontal-labels}, the labels stay horizontal.

\begin{bNiceMatrix}[xdots/horizontal-labels]
1 & \hspace*{1cm} & 0 \\
& & \Ddots^{n \text{ times}} \\
0 & & 1
\end{bNiceMatrix}

10.5 Customisation of the dotted lines

The dotted lines drawn by \texttt{\Ldots}, \texttt{\Cdots}, \texttt{\Vdots}, \texttt{\Ddots}, \texttt{\Iddots}, \texttt{\Hdots} for and \texttt{\Vdots} (and by the command \texttt{\line} in the \texttt{\CodeAfter} which is described p. 34) may be customized by the following options (specified between square brackets after the command):

- \texttt{horizontal-labels};
- \texttt{color};
- \texttt{radius};
- \texttt{shorten-start}, \texttt{shorten-end} and \texttt{shorten};
- \texttt{inter};
- \texttt{line-style}.

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

- \texttt{xdots/horizontal-labels};
- \texttt{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. However, one should remark that the dotted lines drawn in the exterior rows and columns have a special treatment: cf. p. 26.

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).

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).

\begin{tikzpicture}
\draw [dotted] (0,0) -- (5,0);
\end{tikzpicture}

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 pathes (with the exception of “color”, “shorten >” and “shorten <”).

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

\[
\begin{pNiceMatrix} [nullify-dots, xdots/line-style=loosely dotted]
\begin{NiceMatrix}
\begin{array}{cccccc}
\hline
a & b & 0 & \cdots & 0 & \vdotswithin{\text{b}} \\
\hline
b & a & b & \ddots & & \vdotswithin{\text{a}} \\
0 & b & a & \ddots & & \\
& \ddots & \ddots & \ddots & \ddots & \\
0 & \cdots & 0 & b & a \\
\end{array}
\end{NiceMatrix}
\end{pNiceMatrix}
\]

46In 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 `\Dots` (and, in that case, their names are `shorten`, `shorten-start` and `shorten-end`), they apply to all the extremeties.

47The 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 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, hlines, and hvlines-except-borders and by the tools created by \customline are not drawn within the blocks).

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

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.

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 an example which uses the delimiters \lgroup and \rgroup.

\begin{NiceArray}{(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{NiceArray}

For this example, it would also have been possible to use the environment \{NiceArrayWithDelims\} (cf. section 14.10, p. 48) and the key last-col (cf. p. 26).

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

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.

\begin{pNiceArray}{{(c)(c)(c)}}
a_{(1)} & a_{(2)} & a_{(3)} \& \enskip \& a_{(1)} \\
a_{(2)} & \displaystyle \int_0^1 \dfrac{1}{x^2+1} \, dx & a_{(3)} \& \enskip \& a_{(3)} \\
a_{(3)} & a_{(3)} & a_{(3)}
\end{pNiceArray}

\begin{pmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & 1 & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{pmatrix}

\begin{pmatrix}
a_1 \\
\int_0^1 \frac{1}{x^2+1} \, dx \\
a_3 \\
\end{pmatrix}

48 On the other side, the command \Line in the \CodeAfter (cf. p. 34) does not create block.

49 This syntax is inspired by the extension blkarray.

50 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.
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 \Sect{12.2} p. 35.

12 The \CodeAfter

The option code-after may be used to give some code that will be executed after the construction of the matrix.\footnote{There is also a key code-before described p. 34.}

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. Although \CodeAfter is a keyword, it takes in an optional argument (between square brackets).\footnote{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.}

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. 48.

Moreover, several special commands are available in the \CodeAfter: line, \SubMatrix, \OverBrace, \UnderBrace and \TikzEveryCell. We will now present these commands.

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 \texttt{i-j} where is the number of the row and \texttt{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. 31).

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

\begin{verbatim}
\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}
\end{verbatim}

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. 54).
\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
\end{bNiceMatrix}

\begin{NiceArray}{ccc@{\hspace{1.5em}}c}
1 & 1 & 1 & x \\
\dfrac{1}{4} & \dfrac{1}{2} & \dfrac{1}{4} & y \\
1 & 2 & 3 & z
\end{NiceArray}

\SubMatrix({1-1}{3-3}) \SubMatrix({1-4}{3-4})

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 \textit{i-j} where \textit{i} the number of row and \textit{j} the number of column;
- 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 \texttt{key=value} pairs.\footnote{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}}).}

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 \texttt{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.

1 & 1 & 1 & x \\
\dfrac{1}{4} & \dfrac{1}{2} & \dfrac{1}{4} & y \\
1 & 2 & 3 & z
\SubMatrix({1-1}{3-3}) \SubMatrix({1-4}{3-4})

Eventually, in this example, it would probably have been easier to put the delimiters directly in the preamble of \texttt{NiceArray} (see section \ref{tabular}, p. \pageref{tabular}) with the following construction.

In fact, the command \SubMatrix also takes in two optional arguments specified by the traditional symbols \texttt{^} and \texttt{_} for material in superscript and subscript.
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.

Here is the same example with the key **slim** used for one of the submatrices.
There is also a key **name** which gives a name to the submatrix created by \texttt{\SubMatrix}. That name is used to create PGF/TikZ nodes: cf p. 53.

Despite its name, the command \texttt{\SubMatrix} may also be used within a \texttt{\NiceTabular}. Here is an example (which uses \texttt{\bottomrule} and \texttt{\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{{2-2}{3-2}}
\end{NiceTabular}

The command \texttt{\SubMatrix} is, in fact, also available in the \texttt{\CodeBefore}. By using \texttt{\SubMatrix} in the \texttt{\CodeBefore}, the delimiters drawn by those commands \texttt{\SubMatrix} are taken into account to limit the continuous dotted lines (drawn by \texttt{\Cdots}, \texttt{\Vdots}, etc.) which have an open extremity. For an example, see voir 18.9 p. 68.

*Caution*: The following functionality is fragile and does not work with the compilation \texttt{latex–dvips–ps2pdf}.

The key **code** of the command \texttt{\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 \texttt{i-|j} are interpreted with \texttt{i} and \texttt{j} as numbers of row and columns relative to the submatrix.\footnote{Be careful: the syntax \texttt{j|-i} is not allowed.}

\begin{NiceArray}{ccc@{}w{c}{5mm}@{}ccc}
& & \textbf{\texttt{-1 \ 1 \ 2}} & \\
& & \textbf{\texttt{0 \ 3 \ 4}} & \\
\textbf{\texttt{1 \ 2 \ 3}} & \textbf{\texttt{-1 \ 7 \ 25}} & \\
\textbf{\texttt{0 \ 4 \ 5}} & \textbf{\texttt{0 \ 12 \ 41}} & \\
\textbf{\texttt{0 \ 0 \ 6}} & \textbf{\texttt{0 \ 0 \ 30}} & \\
\end{NiceArray}

As we see, the drawing done by our command \texttt{\MyDraw} is relative to the submatrix to which it is applied.
12.3 The commands \texttt{\textbackslash OverBrace} and \texttt{\textbackslash UnderBrace} in the \texttt{\textbackslash CodeAfter}

The commands \texttt{\textbackslash OverBrace} and \texttt{\textbackslash 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 \texttt{i-j} where \texttt{i} the number of row and \texttt{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 \texttt{nicematrix} (with \texttt{PGF}) above the brace (for the command \texttt{\textbackslash OverBrace}) or under the brace (for \texttt{\textbackslash UnderBrace}).

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

\textit{Caution:} There is no vertical space reserved for those braces and their labels.\footnote{See: https://tex.stackexchange.com/questions/685755}

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

- \texttt{\textbackslash left-shorten} and \texttt{\textbackslash right-shorten} which do not take in value; when the key \texttt{\textbackslash 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 \texttt{\textbackslash right-shorten}).

- \texttt{\textbackslash shorten}, which is the conjunction of the keys \texttt{\textbackslash left-shorten} and \texttt{\textbackslash right-shorten};

- \texttt{\textbackslash yshift}, which shifts vertically the brace (and its label);

- \texttt{\textbackslash color}, which sets the color of the brace (and its label).

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

12.4 The command \texttt{\textbackslash TikzEveryCell} in the \texttt{\textbackslash CodeAfter}

\textbf{New 6.24} The command \texttt{\textbackslash TikzEveryCell} execute with TikZ the rectangular path corresponding to each cell of the tabular with parameters of TikZ the argument of \texttt{\textbackslash 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. 26) and those in the empty corners (when the key \texttt{\textbackslash corners} is used: cf. p. 2). It applies in fact to each block (excepted those with the key \texttt{\textbackslash transparent}) and does not apply to the cells located within these blocks.

In fact, in the list of keys provided as argument of \texttt{\textbackslash TikzEveryCell}, it’s possible to put a key \texttt{\textbackslash offset}. That key is not provided by TikZ but by \texttt{nicematrix}. It will narrow the rectangular frame corresponding to the block by a margin (horizontally and vertically) equal to the value (of that key

\footnote{See: https://tex.stackexchange.com/questions/685755}
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}

The command \TikzEveryCell has two keys:

- with the key empty, the command only acts on the empty cells;
- with key non-empty, the command only acts on the non-empty cells.

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. \[p. 26\]), 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 \[1\], \[p. 12\] 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) and 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 \texttt{\tabularnote} which gives the ability to specify notes that will be composed at the end of the array with a width of line equal to the width of the array (except the potential exterior columns specified by \texttt{first-col} and \texttt{last-col}: cf. \S\ 9, p. 26). 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 \texttt{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 \texttt{enumitem}.

\begin{NiceTabular}{@{}llr@{}}
\toprule 
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}

- If you have several successive commands \texttt{\tabularnote{...}} \textit{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 \texttt{multiple} of \texttt{footmisc} for the footnotes).
- If a command \texttt{\tabularnote{...}} is exactly at the end of a cell (with no space at all after), 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. However, since the version 6.20a, that behaviour is in force only when the horizontal alignment mode of the corresponding column is \texttt{c} or \texttt{r}.
- If the key \texttt{notes/para} is used, the notes are composed at the end of the array in a single paragraph (as with the key \texttt{para} of \texttt{threeparttable}).
- There is a key \texttt{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 \texttt{\CodeAfter}).
- If the package \texttt{booktabs} has been loaded (before or after nicematrix), the key \texttt{notes/bottomrule} draws a \texttt{\bottomrule} of \texttt{booktabs} after the notes.
• The command \texttt{\tabularnote} may be used \texttt{before} the environment of \texttt{nicematrix}. Thus, it’s possible to use it on the title inserted by \texttt{\caption} in an environment \texttt{\{table\}} of \LaTeX{} (or in a command \texttt{\captionof} of the package \texttt{caption}). It’s also possible, as expected, to use the command \texttt{\tabularnote} in the caption provided by the \texttt{key} \texttt{caption} of the environment \texttt{\{NiceTabular\}}.

If several commands \texttt{\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 \texttt{notes/detect-duplicates}.

• It’s possible to create a reference to a tabular note created by \texttt{\tabularnote} (with the usual command \texttt{\label} used after the \texttt{\tabularnote}).

• The command \texttt{\tabularnote} has an optional argument (between square brackets) to change the symbol of the reference of the note.

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

For an illustration of some of those remarks, see table \texttt{1} p. 42. This table has been composed with the following code (the package \texttt{caption} has been loaded in this document).

\begin{table}
\centering
\NiceMatrixOptions{caption-above}
\begin{NiceTabular}{@{}llc@{}}
\begin{tabularnote}{$\star$}
\texttt{It’s possible to put a tabular note in the caption.}
\end{tabularnote}
\toprule
Last name & First name & Length of life \\
\midrule
Churchill & Wiston & 91 \\
Nightingale & \texttt{Considered as the first nurse of history} & \texttt{\textquotesingle \textquotesingle \texttt{Considered as the first nurse of history\textquotesingle \textquotesingle \texttt{the Lady with the Lamp\textquotesingle \textquotesingle}}} \\
& \texttt{\textquotesingle \textquotesingle \texttt{the Lady with the Lamp\textquotesingle \textquotesingle}} & 90 \\
Schoelcher & Victor & 89 \\
Touchet & Marie & \texttt{The label of the note is overlapping.} \\
Wallis & John & 87 \\
\bottomrule
\end{NiceTabular}
\end{table}

13.4 Customisation of the tabular notes

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

• notes/para

• notes/bottomrule

• notes/style

• notes/label-in-tabular

\footnote{For technical reasons, the final user is not allowed to put several commands \texttt{\tabularnote} with exactly the same argument in the caption of the tabular.}
Table 1: A tabular whose caption has been specified by the key `caption`\textsuperscript{*}

\begin{tabular}{lll}
\textbf{Last name} & \textbf{First name} & \textbf{Length of life} \\
Churchill & Wiston & 91 \\
Nightingale\textsuperscript{a,b} & Florence\textsuperscript{c} & 90 \\
Schoelcher & Victor & 89\textsuperscript{d} \\
Touchet & Marie\textsuperscript{c} & 89 \\
Wallis & John & 87 \\
\end{tabular}

Some text before the notes.

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

- \texttt{notes/label-in-list}
- \texttt{notes/enumitem-keys}
- \texttt{notes/enumitem-keys-para}
- \texttt{notes/code-before}

For sake of commodity, it is also possible to set these keys in \texttt{\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`:

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

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: \texttt{false}
  That key is also available within a given environment.

- The key `notes/bottomrule` adds a `\bottomrule` of \texttt{booktabs} after the notes. Of course, that rule is drawn only if there is really notes in the tabular. The package \texttt{booktabs} must have been loaded (before or after the package \texttt{nicematrix}). If it is not, an error is raised.
  Initial value: \texttt{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. 12).

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. 58.

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 (with a version of LaTeX at least 2020/10/01).

```
\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 pointless because of the functionalities provided by \nicematrix (see the key `caption` in the section 13.1, p. 39).

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. 12.

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\textsuperscript{57}). 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 when the key `color-inside` is used) will respect those rounded corners.

• When the key `hvlines` is used, the exterior rules will be drawn with rounded corners.\textsuperscript{58}

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

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

\textsuperscript{57}This value is the initial value of the `rounded corners` of Tikz.

\textsuperscript{58}Of course, when used in an environment with delimiters (`\pNiceMatrix`, `\bNiceArray`, etc.), the key `hvlines` does not draw the exterior rules.
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 \ \& & test \ \& & 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. The implementation doesn’t use explicitly any private macro of siunitx.

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

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

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.

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

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.59

\NiceMatrixOptions% 
\{code-for-first-row = \scriptstyle \rotate \text{image of } \}, 
\code-for-last-col = \scriptstyle \}
$A = \begin{pNiceMatrix}[first-row,last-col=4]
e_1 & e_2 & e_3 \ \& \\
e_1 & e_2 & e_3 \\
e_2 & e_3 \\
e_3 \\
\end{pNiceMatrix}$

59It can also be used in \RowStyle (cf. p. 22)
If the command \rotate is used in the “last row” (exterior to the matrix), the corresponding elements are aligned upwards as shown below.

\begin{NiceMatrixOptions}
\{code-for-last-row = \scriptstyle \rotate ,
\hspace{2mm} 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}$

The command \rotate has a key \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 on \W; if it is used in other column types (such as p{...}), the result will maybe differ from what is expected.

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, 
\hspace{2mm} code-for-last-col = \scriptscriptstyle, 
\hspace{2mm} columns-width = 3mm ]
1 & -2 & 3 & 4 & 5 \ \backslash \\
0 & 3 & 2 & 1 & 2 & L_2 \ \gets 2 L_1 - L_2 \ \backslash \\
0 & 1 & 2 & 3 & L_3 \ \gets L_1 + L_3 \ \backslash 
\end{bNiceArray}$

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 amsmath) 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 \[1\], p. \[33\].
14.7 The counters \texttt{iRow} and \texttt{jCol}

In the cells of the array, it’s possible to use the \LaTeX\ counters \texttt{iRow} and \texttt{jCol} which represent the number of the current row and the number of the current column\textsuperscript{60}. Of course, the user must not change the value of these counters which are used internally by \texttt{nicematrix}.

In the \texttt{\CodeBefore} (cf. p. \pageref{CodeBefore}) and in the \texttt{\CodeAfter} (cf. p. \pageref{CodeAfter}), \texttt{iRow} represents the total number of rows (except the potential exterior rows) and \texttt{jCol} represents the total number of columns (except the potential exterior columns).

$$\begin{pNiceMatrix}% don’t forget the %
    & & & & \\
    & 1 & 2 & 3 & 4 \\
    & 5 & 6 & 7 & 8 \\
    & 9 & 10 & 11 & 12
\end{pNiceMatrix}$$

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

The package \texttt{nicematrix} also provides commands in order to compose automatically matrices from a general pattern. These commands are \texttt{\AutoNiceMatrix}, \texttt{\pAutoNiceMatrix}, \texttt{\bAutoNiceMatrix}, \texttt{\vAutoNiceMatrix}, \texttt{\VAutoNiceMatrix} and \texttt{\BAutoNiceMatrix}.

These commands take in two mandatory arguments. The first is the format of the matrix, with the syntax \texttt{n-p} where \texttt{n} is the number of rows and \texttt{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).

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

14.8 The key \texttt{light-syntax}

The option \texttt{light-syntax} (inspired by the package \texttt{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.

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

It’s possible to change the character used to mark the end of rows with the option \texttt{end-of-row}. As said before, the initial value is a semicolon.

When the option \texttt{light-syntax} is used, it is not possible to put verbatim material (for example with the command \texttt{\verb}) in the cells of the array\textsuperscript{61}.

\textsuperscript{60}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.

\textsuperscript{61}The reason is that, when the option \texttt{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.
14.9 Color of the delimiters

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

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

This color also applies to the delimiters drawn by the command \texttt{\SubMatrix} (cf. p. \texttt{35}) and for the delimiters directly specified in the preamble of the environments with preamble (cf. p. \texttt{33}).

14.10 The environment \texttt{NiceArrayWithDelims}

In fact, the environment \texttt{pNiceArray} and its variants are based upon a more general environment, called \texttt{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 \texttt{NiceArrayWithDelims} if we want to use atypical or asymmetrical delimiters.

\begin{NiceArrayWithDelims}
\begin{array}{ccc}
\downarrow & 1 & 2 \\
\uparrow & 4 & 5 \\
\end{array}
\end{NiceArrayWithDelims}

14.11 The command \texttt{\OnlyMainNiceMatrix}

The command \texttt{\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 \texttt{nicematrix}, that command is no-op.

For an example of utilisation, see tex.stackexchange.com/questions/488566

15 Use of TikZ with \texttt{nicematrix}

15.1 The nodes corresponding to the contents of the cells

The package \texttt{nicematrix} creates a PGF/TikZ node\textsuperscript{62} 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).\textsuperscript{63}

\textbf{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 \texttt{\NotEmpty}.\textsuperscript{63}

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

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

The command \texttt{\NiceMatrixLastEnv} provides the number of the last environment of \texttt{nicematrix} (for \LaTeX, it’s a “fully expandable” command and not a counter).

\textsuperscript{62}We recall that TikZ is a layer over PGF. The extension \texttt{nicematrix} loads PGF but does not load TikZ. We speak of PGF/TikZ nodes to emphase the fact that the PGF nodes created by \texttt{nicematrix} may be used with PGF but also with TikZ. The final user will probably prefer to use TikZ rather than PGF.

\textsuperscript{63}One should note that, with that command, the cell is considered as non-empty, which has consequences for the continuous dotted lines (cf. p. \texttt{27}) and the computation of the “corners” (cf. p. \texttt{13}).
However, it’s advisable to use instead the key `name`. 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 \texttt{nicematrix} doesn’t load TikZ by default. In the following examples, we assume that TikZ has been loaded.

\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) ;

Don’t forget the options \texttt{remember picture} and \texttt{overlay}.

In the \texttt{CodeAfter}, the things are easier: one must refer to the nodes with the form \texttt{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}

The nodes of the last column (except the potential “last column” specified by \texttt{last-col}) may also be indicated by \texttt{i-last}. Similarly, the nodes of the last row may be indicated by \texttt{last-j}.

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

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

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 \texttt{nicematrix/cell-node} and its definition in the source file \texttt{nicematrix.sty} is as follows:

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

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

For an example of utilisation, see part \texttt{18.10.1}, p. 68.

\footnote{Since the version 6.20a, the value of that key is \textit{expanded} (in the TeX sens).}
\footnote{For the exterior columns, cf. part \texttt{p. 24}.}
15.1.1 The key \texttt{pgf-node-code}

For the experienced users, \texttt{nicematrix} provides the key \texttt{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 precisely, the value given to the key \texttt{pgf-node-code} will be passed in the fifth argument of the command \texttt{\pgfnode}. That value should contain at least an instruction such as \texttt{\pgfusepath}, \texttt{\pgfusepathqstroke}, \texttt{\pgfusepathqfill}, etc.

15.1.2 The columns \texttt{V} of \texttt{varwidth}

When the extension \texttt{varwidth} is loaded, the columns of the type \texttt{V} defined by \texttt{varwidth} are supported by \texttt{nicematrix}. It may be interesting to notice that, for a cell of a column of type \texttt{V}, the PGF/TikZ node created by \texttt{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 \texttt{p}, \texttt{m} or \texttt{b} for which the nodes have always a width equal to the width of the column. In the following example, the command \texttt{\lipsum} is provided by the eponymous package.

\begin{NiceTabular}{V{10cm}}
  \bfseries \large
  \textbf{Titre} \\
  \lipsum[1][1-4]
\end{NiceTabular}

\begin{CodeAfter}
  \tikz \draw [rounded corners] (1-1) -| (last-|2) -- (last-|1) |- (1-1) ;
\end{CodeAfter}

\begin{NiceTabular}{V{10cm}}
  \bfseries \large
  \textbf{Titre} \\
  \lipsum[1][1-4]
\end{NiceTabular}


We have used the nodes corresponding to the position of the potential rules, which are described below (cf. p. 52).

15.2 The “medium nodes” and the “large nodes”

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

These nodes are not used by \texttt{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 “\texttt{-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 “\texttt{-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.

\[
\begin{pmatrix}
  a & a+b & a+b+c \\
  a & a & a+b \\
  a & a & a
\end{pmatrix}
\]

\[\text{\texttt{create-extra-nodes}}\] which is an alias for the conjunction of \texttt{create-medium-nodes} and \texttt{create-large-nodes}.

\[\text{\texttt{create-extra-nodes}}\] There is no “large nodes” created in the exterior rows and columns (for these rows and columns, cf. p. 52).
The “large nodes” of the first column and last column may appear too small for some usage. That’s why it’s possible to use the options `left-margin` and `right-margin` to add space on both sides of the array and also space in the “large nodes” of the first column and last column. In the following example, we have used the options `left-margin` and `right-margin`.

\[
\begin{array}{ccc}
\hline
a & a+b & a+b+c \\
\hline
a & a & a+b \\
\hline
a & a & a \\
\hline
\end{array}
\]

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.

\[
\begin{array}{ccc}
\hline
a & a+b & a+b+c \\
\hline
a & a & a+b \\
\hline
a & a & a \\
\hline
\end{array}
\]

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:

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

Here, we have colored all the cells of the array with \texttt{chessboardcolors}.

The nodes we have described are not available by default in the \texttt{CodeBefore} (described p. [16]). It’s possible to have these nodes available in the \texttt{CodeBefore} by using the key \texttt{create-cell-nodes} of the keyword \texttt{CodeBefore} (in that case, the nodes are created first before the construction of the array by using informations written on the \texttt{aux} file and created a second time during the construction of the array itself).

Here is an example which uses these nodes in the \texttt{CodeAfter}.

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

\footnote{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.}
\[\begin{align*}
\phantom{u_5} - \phantom{u_4} & \phantom{\ddots} & u_n - u_{n-1} &= r \\
\hline
u_n & - & u_0 &= nr
\end{align*}\]

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 is also a node called \(i.5\) midway between the node \(i\) and the node \(i+1\). These nodes are available in the \CodeBefore and the \CodeAfter.

\begin{NiceMatrix}
\tikz \draw [fill=red!15] (7-|4) |- (8-|5) |- (9-|6) |- cycle ;
\Body
1 \[1\]
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}

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 \((i-1j)\).

\begin{NiceMatrix}
\begin{align*}
1 & & \text{tulipe} & & \text{lys} \\
\text{arum} & & \text{violette mauve} & & \text{muguet} & & \text{dahlia} & & \text{violet} & & 4
\end{align*}
\end{NiceMatrix}
The nodes of the form $i.5$ 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 \\
\CodeAfter
\tikz \draw[red] (3.5-|1) -- (3.5-|last) ; \\
\end{pNiceArray}

15.4 The nodes corresponding to the command \texttt{\SubMatrix}

The command \texttt{\SubMatrix} available in the \texttt{\CodeAfter} has been described p. 35.

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

The nodes \texttt{MyName-left} and \texttt{MyName-right} correspond to the delimiters left and right and the node \texttt{MyName} correspond to the submatrix itself.

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

\[
\begin{array}{cccc}
121 & 23 & 345 & 345 \\
45 & 346 & 863 & 444 \\
3462 & 38458 & 34 & 294 \\
34 & 7 & 78 & 309 \\
\end{array}
\]

16 API for the developers

The package nicematrix provides two variables which are internal but public\footnote{\textit{According to the LaTeX3 conventions, each variable with name beginning with $\texttt{\g_nicematrix}$ or $\texttt{\l_nicematrix}$ is public and each variable with name beginning with $\texttt{\g__nicematrix}$ or $\texttt{\l__nicematrix}$ is private.}}:

- \texttt{\g_nicematrix_code_before_tl} ;

- \texttt{\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 \texttt{\CodeBefore} and \texttt{\Body}) and the “code-after” (usually specified at the end of the environment after the keyword \texttt{\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 \texttt{\g_nicematrix_code_before_tl} needs one compilation more (because the instructions are written on the \texttt{aux} file to be used during the next run).
Example: We want to write a command \texttt{\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 Ti\textit{k}Z before the drawing. It’s possible to program such command \texttt{\crossbox} as follows, explicitly using the public variable \texttt{\g_nicematrix_code_before_tl}.

\begin{verbatim}
\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 } } } }
\ExplSyntaxOff
\end{verbatim}

Here is an example of utilisation:

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

17 Technical remarks

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

17.1 Diagonal lines

By default, all the diagonal lines\footnote{We speak of the lines created by \texttt{\Ddots} and not the lines created by a command \texttt{\line} in the \texttt{\CodeAfter}.} 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 \texttt{\Ddots} in the array can have a marked effect on the final result.

In the following examples, the first \texttt{\Ddots} instruction is written in color:
Example with parallelization (default):

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

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

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

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

The same example without parallelization:

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

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

An instruction like \Ldots, \Cdots, etc. tries to determine the first non-empty cell on both sides. When the key corners is used (cf. p. 12), 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\textsuperscript{71} and X\textsuperscript{72}, 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, 1, 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).

- 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.

\textsuperscript{71}The columns of type V are provided by varwidth, which must be loaded: cf. p. 24
\textsuperscript{72}See p. 24
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\(^{73}\). The environment \{matrix\} of amsmath and its variants (\{pmatrix\}, \{vmatrix\}, etc.) of amsmath prefer to delete these spaces with explicit instructions \hskip -\arraycolsep\(^{74}\). 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

The package nicematrix is not compatible with the class ieeeaccess (because that class is not compatible with PGF/TikZ).\(^{75}\)

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

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

In order to use nicematrix with the class sn-jnl (of Springer Nature), pgf must be loaded before the \documentclass with \RequirePackage:

\RequirePackage{pgf}
\documentclass{sn-jnl}

However, it seems that this is no longer mandatory with the recent versions of sn-jnl.

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

The package nicematrix is not fully compatible with the package aredshln (because this package redefines many internals of \array) and does 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 \ref{sec:custom-line}, p. \pageref{sec:custom-line}), 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).

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.\(^{76}\) For the following example, we also need the TikZ library patterns.

\(^{73}\)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).

\(^{74}\)And not by inserting @$\$ on both sides of the preamble of the array. As a consequence, the length of the underline is not modified and may appear too long, in particular when using square brackets.

\(^{75}\)See https://tex.stackexchange.com/questions/528975/error-loading-tikz-in-ieeeaccess-class

\(^{76}\)By default, nicematrix only loads pgf, which is a sub-layer of TikZ.
\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}

In the following example, we use the key \texttt{tikz} to hatch a row of the tabular. Remark that you use the key \texttt{transparent} of the command \texttt{Block} in order to have the rules drawn in the block.\textsuperscript{77}

\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}

18.2 Use with \texttt{tcolorbox}

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

\textsuperscript{77}By default, the rules are not drawn in the blocks created by the command \texttt{Block}: cf. section \textsuperscript{5} p. 78
18.3 Notes in the tabulars

The tools provided by \texttt{nicematrix} for the composition of the tabular notes have been presented in the section \[ p. 38. \]

Let’s consider that we wish to number the notes of a tabular with stars.\(^7\)

First, we write a command \texttt{\stars} similar the well-known commands \texttt{arabic}, \texttt{alph}, \texttt{Alph}, etc. which produces a number of stars equal to its argument.\(^7\)

\begin{Verbatim}
\ExplSyntaxOn
\NewDocumentCommand \stars \{ m \}
\prg_replicate:n { \value { #1 } } { $ \star $ }
\ExplSyntaxOff
\end{Verbatim}

Of course, we change the style of the labels with the key \texttt{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

\begin{verbatim}
\newcommand{\arraystretch}{1.2}% <-- the \% is mandatory here
\begin{NiceTabular}{rcl}\hline
\CodeBefore\rowcolor{red!15}{1} \Body 
One & Two & Three \\ 
Men & Mice & Lions \\ 
Upper & Middle & Lower 
\end{NiceTabular}
\end{verbatim}
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 \texttt{value(tabularnote)} (because \texttt{tabularnote} is the LaTeX counter used by \texttt{tabularnote} and, therefore, at the end of the tabular, its value is equal to the total number of tabular notes). We use the key \texttt{widest*} of \texttt{enumitem} in order to require a width equal to that value: \texttt{widest*=value(tabularnote)}.

\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 \\
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}

\begin{NiceMatrixOptions}
\text{notes =}
\{
\text{style = \texttt{stars(#1)} ,
\texttt{enumitem-keys =}
\{
\text{widest* = value(tabularnote) ,
align = right
\}
\}
\}
\begin{NiceTabular}{{}llr{}}
\toprule \RowStyle{\bfseries} \\
Last name & First name & Birth day \\
\midrule \\
Achard & Jacques & June 5, 2005 \\
Lefebvre & Mathilde & January 23, 1975 \\
Vanesse & Stephany & October 30, 1994 \\
Dupont & Chantal & January 15, 1998 \\
\bottomrule
\end{NiceTabular}

18.4 Dotted lines

An example with the resultant of two polynomials:

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

59
An example for a linear system:

\[
\begin{pmatrix}
1 & 1 & 1 & \cdots & 1 & 0 & \\
0 & 1 & 0 & \cdots & 0 & \quad L_2 \gets L_2 - L_1 \\
0 & 0 & 1 & \cdots & \vdots & \quad L_3 \gets L_3 - L_1 \\
& & & \cdots & & \vdots & \vdots \\
\vdots & & & \cdots & 0 & \\
0 & & & \cdots & 1 & \quad L_n \gets L_n - L_1
\end{pmatrix}
\]

18.5 Dotted lines which are no longer dotted

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

\[
\begin{pmatrix}
1 & 1 & 1 & \cdots & 1 & 0 \\
0 & 1 & 0 & \cdots & 0 & \quad L_2 \gets L_2 - L_1 \\
0 & 0 & 1 & \cdots & 0 & \\
0 & \cdots & 0 & 1 & 0 & \quad L_n \gets L_n - L_1
\end{pmatrix}
\]

\[
\begin{pmatrix}
1 & 1 & 1 & \cdots & 1 & 0 \\
0 & 1 & 0 & \cdots & 0 & \quad L_2 \gets L_2 - L_1 \\
0 & 0 & 1 & \cdots & 0 & \\
0 & \cdots & 0 & 1 & 0 & \quad L_n \gets L_n - L_1
\end{pmatrix}
\]
In fact, it’s even possible to draw solid lines with the commands \texttt{\textbackslash Cdots}, \texttt{\textbackslash Vdots}, etc.\textsuperscript{80}

\begin{verbatim}
\NiceMatrixOptions{xdots={horizontal-labels,line-style = <->}}$
$\begin{pNiceArray}{ccc|cc}$\begin{bNiceMatrix}
& 1 & 0 & 1 \\
2 & 1 & 1 & 1 & 1 & \Vdotsfor{3}\cline{2-5}
1 & 1 & 1 & 1 & 1 & \Hline
1 & 1 & 1 & 1 & 1 & \Vdotsfor{2}\cline{2-5}
1 & 1 & 1 & 1 & 1$
\end{bNiceMatrix}$
\end{pNiceArray}$
\end{verbatim}

If you want the label on the line, you should use the special token “;”:

\begin{verbatim}
\NiceMatrixOptions{xdots={horizontal-labels,line-style = <->}}$
$\begin{pNiceArray}{ccc|cc}$\begin{bNiceMatrix}
& 1 & 0 & 1 \\
2 & 1 & 1 & 1 & 1 & \Vdotsfor{3};\cline{2-5}
1 & 1 & 1 & 1 & 1 & \Hline
1 & 1 & 1 & 1 & 1 & \Vdotsfor{2};\cline{2-5}
1 & 1 & 1 & 1 & 1$
\end{bNiceMatrix}$
\end{pNiceArray}$
\end{verbatim}

\textsuperscript{80}In this document, the TikZ library \texttt{arrows.meta} has been loaded, which impacts the shape of the arrow tips.
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 & 0 & 1 & 2 & 0 & 0 \\
0 & 0 & 0 & \Block[borders={left,top,tikz=dashed}]{2-2}{}
3 & 4 \\
0 & 0 & 0 & 0 & 1 & 4
\end{pNiceMatrix}

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{Verbatim}[formatcom=\small\color{gray}]
\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 }
\setlength{\extrarowheight}{1mm}
\quad $\begin{pNiceMatrix}
1 & 2 ; 0 & 0 & 0 & 0 \\
4 & 5 ; 0 & 0 & 0 & 0 \\
0 & 0 ; 7 & 1 & 0 & 0 \\
0 & 0 ; -1 & 2 & 0 & 0 \\
0 & 0 ; 0 & 3 & 4 \\
0 & 0 & 0 & 0 & 1 & 4
\end{pNiceMatrix}$
\end{Verbatim}
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 slimmer).

In order to 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 }
\setlength{\extrarowheight}{1mm}
\end{NiceMatrixBlock}
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.

\begin{NiceMatrix}[
 r, last-col=6, code-for-last-col = \scriptstyle \color{blue} ]
\setlength{\extrarowheight}{1mm}
\begin{array}{ccccccc}
12 & -8 & 7 & 5 & 3 \\
0 & 64 & -41 & 1 & 19 & L_2 \rightarrow L_1-4L_2 \\
0 & -192 & 123 & -3 & -57 & L_3 \rightarrow L_1+4L_3 \\
0 & -64 & 41 & -1 & -19 & L_4+3L_1+4L_4 \\
12 & -8 & 7 & 5 & 3 \\
0 & 64 & -41 & 1 & 19 \\
0 & 0 & 0 & 0 & 0 & L_3 \rightarrow 3L_2+L_3 \\
0 & -64 & 41 & -1 & -19 & L_4 \rightarrow 3L_1-4L_4 \\
\end{array}
\end{NiceMatrix}
\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}
In this tabular, the instructions \texttt{\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 \texttt{vlines-in-sub-matrix}, to choice a letter in the preamble of the array to specify vertical rules which will be drawn in the \texttt{\SubMatrix} only (by adding space between the columns).

\begin{verbatim}
\setlength{\extrarowheight}{1mm}
\begin{NiceArray}
 vlines-in-sub-matrix=I,
 last-col,
 code-for-last-col = \scriptstyle \color{blue}
 \{} rrrrIr \}
 \{}rrrrIr\}
 12 & -8 & 7 & 5 & 3 \\ 
 3 & -18 & 12 & 1 & 4 \\ 
 -3 & -46 & 29 & -2 & -15 \\ [1mm] 
 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 \\
\end{NiceArray}
\end{verbatim}

\codeafter
\SubMatrix({1-1}{4-5})
\SubMatrix({5-1}{8-5})
\SubMatrix({9-1}{11-5})
\SubMatrix({12-1}{13-5})
\end{NiceArray}\]

\[
\begin{array}{ccccc|c}
12 & -8 & 7 & 5 & 3 \\
3 & -18 & 12 & 1 & 4 \\
-3 & -46 & 29 & -2 & -15 \\
9 & 10 & -5 & 4 & 7 \\
12 & -8 & 7 & 5 & 3 \\
0 & 64 & -41 & 1 & 19 & L_2 \gets L_1-4L_2 \\
0 & -192 & 123 & -3 & -57 & L_3 \gets L_1+4L_3 \\
0 & -64 & 41 & -1 & -19 & L_4 \gets 3L_1-4L_4 \\
12 & -8 & 7 & 5 & 3 \\
0 & 64 & -41 & 1 & 19 \\
0 & 0 & 0 & 0 & 0 & L_3 \gets 3L_2+L_3 \\
12 & -8 & 7 & 5 & 3 \\
0 & 64 & -41 & 1 & 19 \\
\end{array}
\]
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\(^{81}\)).

\[
\begin{pNiceArray}{>{\strut}cccc}
\Block[draw]{}{a_{11}} & a_{12} & a_{13} & a_{14} \\
a_{21} & \Block[draw]{}{a_{22}} & a_{23} & a_{24} \\
a_{31} & a_{32} & \Block[draw]{}{a_{33}} & a_{34} \\
a_{41} & a_{42} & a_{43} & \Block[draw]{}{a_{44}} \\
\end{pNiceArray}
\]

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.\(^{82}\)

It’s possible to color a row with `\rowcolor` in the code-before (or with `\rowcolor` in the first cell of the row if the key `color-inside` is used—even when `colortbl` is not loaded).

\[
\begin{pNiceArray}{>{\strut}cccc}
\rowcolor{red!15} A_{11} & A_{12} & A_{13} & A_{14} \\
A_{21} & \rowcolor{red!15} A_{22} & A_{23} & A_{24} \\
A_{31} & A_{32} & \rowcolor{red!15} A_{33} & A_{34} \\
A_{41} & A_{42} & A_{43} & \rowcolor{red!15} A_{44} \\
\end{pNiceArray}
\]

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 lines in the preamble of your document do the job:

\begin{verbatim}
\usepackage{tikz}
\usetikzlibrary{fit}
\end{verbatim}

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`.

\(^{81}\) 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.

\(^{82}\) For the command `\cline`, see the remark p. 10.
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) (1-3)] {} ;
\node [highlight = (2-1) (2-3)] {} ;
\node [highlight = (3-1) (3-3)] {} ;
\end{tikzpicture}
\Body
a & a + b & a + b + c & L_1 \\
a & a & a + b & L_2 \\
a & a & a & L_3
\end{pNiceArray}\]

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] [name suffix = -medium]
\begin{tikzpicture}
\node [highlight = (1-1) (1-3)] {} ;
\node [highlight = (2-1) (2-3)] {} ;
\node [highlight = (3-1) (3-3)] {} ;
\end{tikzpicture}
\Body
a & a + b & a + b + c & L_1 \\
a & a & a + b & L_2 \\
a & a & a & L_3
\end{pNiceArray}\]
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.

\[
\begin{array}{*{6}{c}@{\hspace{6mm}}*{5}{c}}
\CodeBefore
\begin{tikzpicture}
  \node [highlight = (9-2) (9-6)] { } ;
  \node [highlight = (2-9) (6-9)] { } ;
\end{tikzpicture}
\CodeAfter
\end{array}
\]

\[\begin{NiceArray}{*{6}{c}@{\hspace{6mm}}*{5}{c}} \nullify-dots \CodeBefore \\begin{array}{*{6}{c}@{\hspace{6mm}}*{5}{c}}
\CodeBefore
\begin{tikzpicture}
  \node [highlight = (9-2) (9-6)] { } ;
  \node [highlight = (2-9) (6-9)] { } ;
\end{tikzpicture}
\CodeAfter
\end{array}\]

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).
\begin{NiceTabular} [ hvlines, corners=SE, baseline = line-9 ] { cccccccc }
\CodeBefore
\chessboardcolors{red!15}{blue!15}
\Body
1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 
1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 
1 & 3 & 6 & 10 & 15 & 21 & 21 \\ 
1 & 4 & 10 & 20 & 35 & 35 & 35 \\ 
1 & 5 & 15 & 35 & 35 & 35 & 35 \\ 
1 & 6 & 21 & 21 & 21 & 21 & 21 \\ 
1 & 7 & 7 & 7 & 7 & 7 & 7 \\ 
1
\end{NiceTabular}
19 History

The successive versions of the file nicematrix.sty provided by TeXLive are 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 6.25 and 6.26

Special color \texttt{nocolor}.

Changes between version 6.24 and 6.25

Correction of several bugs.
An instruction \texttt{\rowlistcolors} in the tabular stops the effect of a previous \texttt{\rowlistcolors}.

Changes between version 6.23 and 6.24

New command \texttt{\TikzEveryCell} available in the \texttt{CodeAfter} and the \texttt{CodeBefore} to apply the same TikZ instruction to all the cells and blocks of the array.
New key \texttt{offset} in the key \texttt{tikz} of a command \texttt{\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 \texttt{opacity} for the command \texttt{\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 \texttt{c} for the command \texttt{\tabularnote}.
New commands \texttt{\rowcolors} and \texttt{\rowlistcolors} available in the array itself (previously, there were only two commands \texttt{\rowcolors} and \texttt{\rowlistcolors} available in the \texttt{CodeBefore}.

Changes between version 6.20 and 6.20a

The value of the key \texttt{name} of an environment of \texttt{nicematrix} is expanded (in the TeX sens).
The labels of the tabular notes (created by the command \texttt{\tabularnote}) are now composed in an overlapping position \texttt{only} when the horizontal alignment mode of the column is \texttt{c} or \texttt{r}.

Changes between version 6.19 and 6.20

New key \texttt{xdots/horizontal-labels}.

Changes between version 6.18 and 6.19

The command \texttt{\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`. 

71
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 (of 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 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.

Changes between versions 5.19 and 6.0

Columns X and environment \NiceTabularX.
Command \rowlistcolors available in the \CodeBefore.
In columns with fixed width, the blocks are composed as paragraphs (wrapping of the lines).
The key define-L-C-R has been deleted.

Changes between versions 5.18 and 5.19

New key tikz for the command \Block.
Changes between versions 5.17 and 5.18

New command \RowStyle

Changes between versions 5.16 and 5.17

The key define-L-C-R (only available at load-time) now raises a (non fatal) error.
Keys L, C and R for the command \Block.
Key hvlines-except-borders.
It’s now possible to use a key 1, r or c with the command \pAutoNiceMatrix (and the similar ones).

Changes between versions 5.15 and 5.16

It’s now possible to use the cells corresponding to the contents of the nodes (of the form i-j) in the \CodeBefore when the key create-cell-nodes of that \CodeBefore is used. The medium and the large nodes are also available if the corresponding keys are used.

Changes between versions 5.14 and 5.15

Key hvlines for the command \Block.
The commands provided by nicematrix to color cells, rows and columns don’t color the cells which are in the “corners” (when the key corner is used).
It’s now possible to specify delimiters for submatrices in the preamble of an environment.
The version 5.15b is compatible with the version 3.0+ of siunitx (previous versions were not).

Changes between versions 5.13 and 5.14

Nodes of the form (1.5), (2.5), (3.5), etc.
Keys t and b for the command \Block.
Key corners.

Changes between versions 5.12 and 5.13

New command \arraycolor in the \CodeBefore (with its key except-corners).
New key borders for the command \Block.
New command \Hline (for horizontal rules not drawn in the blocks).
The keys vlines and hlines takes in as value a (comma-separated) list of numbers (for the rules to draw).

Changes between versions 5.11 and 5.12

Keywords \CodeBefore and \Body (alternative syntax to the key code-before).
New key delimiters/max-width.
New keys hlines, vlines and hvlines for the command \SubMatrix in the \CodeAfter.
New key rounded-corners for the command \Block.

Changes between versions 5.10 and 5.11

It’s now possible, in the code-before and in the \CodeAfter, to use the syntax |(i-j) for the TikZ node at the intersection of the (potential) horizontal rule number i and the (potential) vertical rule number j.
Changes between versions 5.9 and 5.10

New command \SubMatrix available in the \CodeAfter. It's possible to provide options (between brackets) to the keyword \CodeAfter.

Changes between versions 5.8 and 5.9

Correction of a bug: in the previous versions, it was not possible to use the key line-style for the continuous dotted lines when the TikZ library babel was loaded.
New key cell-space-limits.

Changes between versions 5.7 and 5.8

Keys cols and restart of the command \rowcolors in the code-before. Modification of the behaviour of \ at the columns of type p, m or b (for a behaviour similar to the environments of array). Better error messages for the command \Block.

Changes between versions 5.6 and 5.7

New key delimiters-color
Keys fill, draw and line-width for the command \Block.

Changes between versions 5.5 and 5.6

Different behaviour for the mono-row blocks.
New command \NotEmpty.

Changes between versions 5.4 and 5.5

The user must never put \omit before \CodeAfter. Correction of a bug: the tabular notes \tabularnotes were not composed when present in a block (except a mono-column block).

Changes between versions 5.3 and 5.4

Key tabularnote.
Different behaviour for the mono-column blocks.

Changes between versions 5.2 and 5.3

Keys c, r and l for the command \Block.
It's possible to use the key draw-first with \Ddots and \Iddots to specify which dotted line will be drawn first (the other lines will be drawn parallel to that one if parallelization is activated).

Changes between versions 5.1 and 5.2

The vertical rules specified by | or || in the preamble respect the blocks.
Key respect-blocks for \rowcolors (with a s) in the code-before. The variable \g_nicematrix_code_before_tl is now public.
The key baseline may take in as value an expression of the form line-i to align the \hline in the row i.
The key hvlines-except-corners may take in as value a list of corners (eg: NW,SE).
Changes between versions 5.0 and 5.1

The vertical rules specified by | in the preamble are not broken by \hline\hline (and other).
Environment \{NiceTabular*\}
Command \Vdotsfor similar to \Hdotsfor
The variable \g_nicematrix_code_after_tl is now public.

Changes between versions 4.4 and 5.0

Use of the standard column types 1, c and r instead of L, C and R.
It’s now possible to use the command \diagbox in a \Block.
Command \tabularnote

Changes between versions 4.3 and 4.4

New key hvlines-except-corners (now deprecated).

Changes between versions 4.2 and 4.3

The horizontal centering of the content of a \Block is correct even when an instruction such as !{\qquad} is used in the preamble of the array.
It’s now possible to use the command \Block in the “last row”.

Changes between versions 4.1 and 4.2

It’s now possible to write \begin{pNiceMatrix}a&b\end{pNiceMatrix}^2 with the expected result.

Changes between versions 4.0 and 4.1

New keys cell-space-top-limit and cell-space-bottom-limit
New command \diagbox
The key hvline don’t draw rules in the blocks (commands \Block) and in the virtual blocks corresponding to the dotted lines.

Changes between versions 3.15 and 4.0

New environment \{NiceTabular\}
Commands to color cells, rows and columns with a perfect result in the PDF.

Changes between versions 3.14 and 3.15

It’s possible to put labels on the dotted lines drawn by \Ldots, \Cdots, \Vdots, \Ddots, \Iddots, \Hdotsfor and the command \line in the code-after with the tokens _ and ^.
The option baseline is now available in all the environments of nicematrix. Before, it was available only in \{NiceArray\}.
New keyword \CodeAfter (in the environments of nicematrix).

Changes between versions 3.13 and 3.14

Correction of a bug (question 60761504 on stackoverflow).
Better error messages when the user uses \& or \ when light-syntax is in force.
Changes between versions 3.12 and 3.13

The behaviour of the command \rotate is improved when used in the “last row”. The option dotted-lines-margin has been renamed in xdots/shorten and the options xdots/color and xdots/line-style have been added for a complete customisation of the dotted lines. In the environments without preamble (\{NiceMatrix\}, \{pNiceMatrix\}, etc.), it’s possible to use the options l (=L) or r (=R) to specify the type of the columns.

The code of nicematrix no longer uses TikZ but only PGF. By default, TikZ is not loaded by nicematrix.

Changes between versions 3.11 and 3.12

Command \rotate in the cells of the array. Options vlines, hlines and hvlines. Option baseline pour \{NiceArray\} (not for the other environments).

The name of the TikZ nodes created by the command \Block has changed: when the command has been issued in the cell i-j, the name is i-j-block and, if the creation of the “medium nodes” is required, a node i-j-block-medium is created.

If the user tries to use more columns than allowed by its environment, an error is raised by nicematrix (instead of a low-level error).

The package must be loaded with the option obsolete-environments if we want to use the deprecated environments.

Changes between versions 3.10 and 3.11

Correction of a bug linked to first-row and last-row.

Changes between version 3.9 and 3.10

New option light-syntax (and end-of-row).

New option dotted-lines-margin for fine tuning of the dotted lines.

Changes between version 3.8 and 3.9

New commands \NiceMatrixLastEnv and \OnlyMainNiceMatrix.

New options create-medium-nodes and create-large-nodes.

Changes between version 3.7 and 3.8

New programmation for the command \Block when the block has only one row. With this programmation, the vertical rules drawn by the specifier “|” at the end of the block is actually drawn. In previous versions, they were not because the block of one row was constructed with \multicolumn. An error is raised when an obsolete environment is used.

Changes between version 3.6 and 3.7

The four “corners” of the matrix are correctly protected against the four codes: code-for-first-col, code-for-last-col, code-for-first-row and code-for-last-row.

New command \pAutoNiceMatrix and its variants (suggestion of Christophe Bal).

Changes between version 3.5 and 3.6

LaTeX counters iRow and jCol available in the cells of the array.

Addition of vnormalbaselines before the construction of the array: in environments like \{align\} of amsmath the value of \baselineskip is changed and if the options first-row and last-row were used in an environment of nicematrix, the position of the delimiters was wrong.
A warning is written in the .log file if an obsolete environment is used.
There is no longer artificial errors Duplicate-name in the environments of amsmath.

**Changes between version 3.4 and 3.5**
Correction on a bug on the two previous versions where the code-after was not executed.

**Changes between version 3.3 and 3.4**
Following a discussion on TeX StackExchange\(^{83}\), optimization of TikZ externalization is disabled in the environments of nicematrix when the class standalone or the package standalone is used.

**Changes between version 3.2 and 3.3**
The options first-row, last-row, first-col and last-col are now available in the environments \{NiceMatrix\}, \{pNiceMatrix\}, \{bNiceMatrix\}, etc.
The option columns-width=auto doesn’t need any more a second compilation.

**Changes between version 3.1 and 3.2 (and 3.2a)**
Option small.

**Changes between version 3.0 and 3.1**
Command \Block to draw block matrices.
Error message when the user gives an incorrect value for last-row.
The vertical rules in the matrices (drawn by “|”) are now compatible with the color fixed by colortbl.
Correction of a bug: it was not possible to use the colon “:” in the preamble of an array when pdflatex was used with french-babel (because french-babel activates the colon in the beginning of the document).

**Changes between version 2.3 and 3.0**
Modification of \Hdotsfor. Now \Hdotsfor erases the \vlines (of “|”) as \hdotsfor does.
Composition of exterior rows and columns on the four sides of the matrix (and not only on two sides) with the options first-row, last-row, first-col and last-col.

**Changes between version 2.2.1 and 2.3**
Compatibility with the column type S of siunitx.
Option hlines.

**Changes between version 2.2 and 2.2.1**
Improvment of the vertical dotted lines drawn by the specifier “;” in the preamble.
Modification of the position of the dotted lines drawn by \hdottedline.

\(^{83}\)cf. tex.stackexchange.com/questions/510841/nicematrix-and-tikz-external-optimize
Changes between version 2.1.5 and 2.2

Possibility to draw horizontal dotted lines to separate rows with the command \hdottedline (similar to the classical command \hline and the command \hdashline of amsmath).
Possibility to draw vertical dotted lines to separate columns with the specifier “;” in the preamble (similar to the classical specifier “|” and the specifier “;” of amsmath).

Changes between version 2.1.4 and 2.1.5

Compatibility with the classes revtex4-1 and revtex4-2.
Option allow-duplicate-names.

Changes between version 2.1.3 and 2.1.4

Replacement of some options 0 { } in commands and environments defined with xparse by ! 0 { } (because a recent version of xparse introduced the specifier ! and modified the default behaviour of the last optional arguments).
See www.texdev.net/2018/04/21/xparse-optional-arguments-at-the-end

Changes between version 2.1.2 and 2.1.3

When searching the end of a dotted line from a command like \Cdots issued in the “main matrix” (not in the exterior column), the cells in the exterior column are considered as outside the matrix. That means that it’s possible to do the following matrix with only a \Cdots command (and a single \Vdots).
\[
\begin{pmatrix}
C_j \\
\vdots \\
0 \\
0
\end{pmatrix}
\]

Changes between version 2.1 and 2.1.1

Small corrections: for example, the option code-for-first-row is now available in the command \NiceMatrixOptions.
Following a discussion on TeX StackExchange\footnote{cf. tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package}, TikZ externalization is now deactivated in the environments of the package nicematrix.\footnote{Before this version, there was an error when using nicematrix with TikZ externalization. In any case, it’s not possible to externalize the TikZ elements constructed by nicematrix because they use the options overlay and remember picture.}

Changes between version 2.0 and 2.1

New implementation of the environment \pNiceArrayRC. With this new implementation, there is no restriction on the width of the columns.
The package nicematrix no longer loads mathtools but only amsmath.
Creation of “medium nodes” and “large nodes”.

Changes between version 1.4 and 2.0

The versions 1.0 to 1.4 of nicematrix were focused on the continuous dotted lines whereas the version 2.0 of nicematrix provides different features to improve the typesetting of mathematical matrices.

\footnote{cf. tex.stackexchange.com/questions/450841/tikz-externalize-and-nicematrix-package}

\footnote{Before this version, there was an error when using nicematrix with TikZ externalization. In any case, it’s not possible to externalize the TikZ elements constructed by nicematrix because they use the options overlay and remember picture.}

78
Changes between version 1.3 and 1.4

The column types \( \omega \) and \( \Omega \) can now be used in the environments \( \text{NiceArray} \), \( \text{pNiceArrayC} \) and its variants with the same meaning as in the package \text{array}.
New option \texttt{columns-width} to fix the same width for all the columns of the array.

Changes between version 1.2 and 1.3

New environment \( \text{pNiceArrayC} \) and its variants.
Correction of a bug in the definition of \( \text{BNiceMatrix} \), \( \text{vNiceMatrix} \) and \( \text{VNiceMatrix} \) (in fact, it was a typo).
Options are now available locally in \( \text{pNiceMatrix} \) and its variants.
The names of the options are changed. The old names were names in “camel style”.

Changes between versions 1.1 and 1.2

New environment \( \text{NiceArray} \) with column types L, C and R.

Changes between versions 1.0 and 1.1

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

## A

- \arraycolor (command of \CodeBefore), 16
- \arrayrulecolor, 10
- \arrayrulewidth, 10
- auto-columns-width (clé de \NiceMatrixBlock), 63
- key of \NiceMatrixBlock), 24
- \AutoNiceMatrix, 47

## B

- baseline (key for an environment), 3
- \AutoNiceMatrix, 47
- \bAutoNiceMatrix, 47
- \blkarray (package), 33
- \Block, 4
- Blocks in the tabulars, 5
- \Body, see \NiceMatrix
- bold (key of \RowStyle), 23
- booktabs (package), 9
- borders (key of \Block), 5
- bottomrule (subkey of “notes”), 41

## C

- Captions of the tabulars, 39
- caption (key of \NiceTabular), 39
- caption-above, 39
- \Cdots, 27
- \cdottedline, 15
- cell-space-bottom-limit, 2
- cell-space-limits, 2
- cell-space-top-limit, 2
- \cellcolor
- command in tabular, 21
- command of \CodeBefore, 16
- cellspace (package), 2
- \chessboardcolors (commande du \CodeBefore), 14, 68
- \cline (LaTeX command), 10
- code (key of \SubMatrix), 36
- code-after, 34
- code-before
- key for an environment, 16
- subkey of “notes”, 41
- code-for-first-col, 26
- code-for-first-row, 26
- code-for-last-col, 26
- code-for-last-row, 26
- \CodeAfter, 34
- \CodeBefore...
- \Body, 14, 67
- color
- for the delimiters of the matrices, 48
- key for the dotted rules, 31
- key of \Block, 5
- key of \OverBrace and \UnderBrace, 38
- key of \RowStyle, 23
- key of “custom-line”, 13
- color-inside, 21
- colorbl (package), 13

## D

- \Ddots, 27
- \definecolorseries (command of xcolor), 19
- delimiters
- —/color for an environment, 48
- —/color pour \SubMatrix, 36
- —/max-width, 63
- detect-duplicates (subkey of “notes”), 41
- \diagbox, 13

## E

- empty (key of \TikzEveryCell), 38
- end-of-row (to be used with light-syntax), 47
- \enumitem (package required to use \tabularnote), 40
- \enumitem-keys (subkey of “notes”), 41
- \enumitem-keys-para (subkey of “notes”), 41
- exterior-arraycolsep, 56
- extra-height (key of \SubMatrix), 36
extra-left-margin, \[51\]
extra-right-margin, \[51\]

\[F\]
fill (key of \Block), \[8\]
first-col, \[26\]
first-row, \[26\], \[44\]
footnote (package), \[83\]
footnote (key), \[39\]
footnotehyper (package), \[83\]
footnotehyper (key), \[39\]

\[H\]
\hdotsfor, \[28\]
\hdottedline, \[13\]
highlight (TikZ style defined in an example), \[69\]
\Hline, \[11\]
hlines, see Rules
  key for an environment, \[11\]
  key of \Block, \[3\]
  key of \SubMatrix, \[36\]
horizontal-labels (key for the dotted rules), \[31\]
\Hspace, \[28\]
hverticals, see Rules
  key for an environment, \[11\]
  key of \Block, \[3\]
  key of \SubMatrix, \[36\]
hverticals-except-borders, \[11\], \[24\]
\Idots, \[27\], \[54\]
Incompatibilities, \[56\]
inter (key for the dotted rules), \[31\]
iRow (LaTeX counter), \[47\]
\[J\]
jCol (LaTeX counter), \[47\]

\[L\]
label (key of \NiceTabular), \[44\]
label-in-list (subkey of “notes”), \[41\]
label-in-tabular (subkey of “notes”), \[41\]
last-col, \[24\], \[44\], \[60\]
last-row, \[24\], \[44\], \[60\]
\Ldots, \[27\]
\left: used by nicematrix for delimiters in the preambles, \[33\]
left-margin, \[50\]
left-shorten (key of \OverBrace and \UnderBrace), \[28\]
left-xshift (key of \SubMatrix), \[36\]
letter (key of “custom-line”), \[13\]
light-syntax, \[47\]
\line (command of \CodeAfter), \[34\]
line-style (key for the dotted rules), \[34\]
line-width (key of \Block), \[4\]
Lines in the tabulars, see Rules

\[M\]
mathdots (package), \[27\]
max-width (package), \[27\]
multiplicity (subkey of “delimiters”), \[33\]

\[N\]
name
  key for an environment, \[48\]
  key of \Block, \[3\]
  key of \SubMatrix, \[53\]
nb-rows (key of \RowStyle), \[22\]
{NiceArrayWithDelims}, \[18\]
{NiceMatrixBlock}, \[24\], \[62\]
\NiceMatrixOptions, \[1\]
{NiceTabularX}, \[24\]
nocolor, \[22\]

\[O\]
\OnlyMainNiceMatrix, \[48\]
opacity (key of \Block), \[3\]
opacity (key of commands such as \rowcolor, etc.), \[16\]
\OverBrace (command of \CodeAfter and \CodeBefore), \[58\]
\[P\]
para (subkey of “notes”), \[41\]
parallelize-diags, \[54\]
\AutoNiceMatrix, \[47\]
pgf-node-code, \[50\], \[68\]
\[R\]
radius (key for the dotted rules), \[31\]
\rectanglecolor (command of \CodeBefore), \[16\]
renew-dots, \[30\]
renew-matrix, \[30\]
\resetcolorseries (command of xcolor), \[19\]
respect-arraystretch (key of \Block), \[3\]
respect-blocks (key of \rowcolors of \CodeBefore), \[18\]
restart (key of \rowcolors of \CodeBefore), \[18\]
\right: used by nicematrix for delimiters in the preambles, \[33\]
right-margin, \[50\]
right-shorten (key of \OverBrace and \UnderBrace), \[38\]
right-xshift (key of \SubMatrix), \[36\]
\rotate, \[26\], \[23\], \[45\]
rounded-corners
  key of \Block, \[3\]
  key of \NiceTabular, \[44\]
### Rules in the tabulars
- \rowcolor (key of \RowStyle), 22
- \rowcolors (command of \CodeBefore), 16
- \rowlistcolors (command of \CodeBefore), 16
- \RowStyle, 22
- \ShowCellNames (command of \CodeAfter and \CodeBefore), 45
- \Siunitx (package), 15
- \slim (key of \SubMatrix), 36
- \{smallmatrix\} (environment of amsmath), 46
- \standardcline
- \style (subkey of “notes”), 41
- \sub-matrix (key of \CodeAfter, with subkeys), 24
- \SubMatrix (command of \CodeAfter and \CodeBefore), 35
- \TikzEveryCell (command of \CodeAfter and \CodeBefore), 38
- \tabularnote, 58
- \{TabularNote\}, 10
- \tabularnote (key of \{NiceTabular\}), 10
- \tabularx (package), 24
- \tcolorbox (package), 57
- \threeparttable (package), 44
- TikZ (utilisation with nicematrix), 48
- \textwidth
- \vlines, see Rules
- \key for an environment, 11
- \key of \Block, 3
- \key of \SubMatrix, 36
- \vlines-in-sub-matrix, 65
- \W
- width
- \key of \{NiceTabular\}, 24
- \subkey of “rules”, 10
- \X
- X (the columns X), 24
- \xdots (and its subkeys), 27
- \xshift (key of \SubMatrix), 36
- \Y
- yshift (key of \OverBrace and \UnderBrace), 38
- \T
- \\tabularnote, 58
- \\{TabularNote\}, 10
- \tabularnote (key of \{NiceTabular\}), 10
- \tabularx (package), 24
- \tcolorbox (package), 57
- \threeparttable (package), 44
- TikZ (utilisation with nicematrix), 48
- \key of \Block, 3
- \key of “borders” de \Block, 3
- \key of “custom-line”, 14
- \total-width (key of “custom-line”), 14
- \transparent (key of \Block), 1
- \U
- \\UnderBrace (command of \CodeAfter and \CodeBefore), 38
- \V
- V (the columns V of \varwidth), 25
- \v-center (key of \Block), 8
- \varwidth (package), 24
- \VAutoNiceMatrix, 47
- \VAutoNiceMatrix, 47
- \Vdots, 27
- \Vdotsfor, 29

---

82