The drawmatrix package

Elmar Peise
peise@aices.rwth-aachen.de

2017/06/05 v1.5.0

Abstract

drawmatrix provides macros to visually represent matrices. Various options allow to change the visualizations, e.g., drawing rectangular, triangular, or banded matrices.

Contents

1 Introduction

In many situations, visual representations of matrices facilitate the understanding of linear algebra properties, relations, and operations enormously. This package provides simple tools to bring such representations to LaTeX. For instance,

$$\boxed{A}\,\boxed{X} + \boxed{X}\,\boxed{B} = \boxed{C}$$

is typeset as follows:

```
\[
  \drawmatrix[upper]A \;
  \drawmatrix[width=.5]X +
  \drawmatrix[width=.5]X \;
  \drawmatrix[upper, size=.5, bbox height=1]B =
  \drawmatrix[width=.5]C
\]
```

2 Drawing Matrices

`\drawmatrix[`⟨*options*⟩`]{`⟨*label*⟩`}`

Draws a matrix labeled ⟨*label*⟩. The optional ⟨*options*⟩, which modify various aspects of drawn matrix through PGF's key-value system, are introduced in the following sections.

The label is typeset in the surrounding mode and style.

```
$\drawmatrix A$
{\bfseries\drawmatrix A}
{\large\drawmatrix A}
```

$\boxed{A}$ $\boxed{\mathbf{A}}$ $\boxed{\mathrm{A}}$

In equations, parentheses (spanned with `\left` and `\right`), subscripts, and superscripts naturally extend to the drawn shape:

```
$\left(
\drawmatrix A_i +
\drawmatrix B^{-1}
\right)
\drawmatrix C$
```

$$\left(\boxed{A}_{i} + \boxed{B}^{-1}\right)\boxed{C}$$

Used in matrix products, a little space (`\;`) helps to yield a more natural result:

```
$\drawmatrix A \drawmatrix B$

$\drawmatrix A \; \drawmatrix B$
```

$\boxed{A}\,\boxed{B}$

$\boxed{A}\;\boxed{B}$

/drawmatrix/label text=⟨*text*⟩ (no default, initially ⟨*label*⟩)

Stores the label text. It overwrites `\drawmatrix` → P. 2's ⟨*label*⟩ argument.

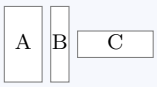
```
$\drawmatrix[label text=B]A$
```

2.1 Size

/drawmatrix/height=⟨*dimension*⟩ (no default, initially 1)
/drawmatrix/width=⟨*dimension*⟩ (no default, initially 1)

Width and height of the drawmatrix in Ti*k*Z's coordinate system `canvas`. May be given in units such as `em` or `cm`.

```
\drawmatrix[width=.5]A
\drawmatrix[width=2ex]B
\drawmatrix[height=.35cm]C
```



A width or height of `0` are useful to represent vectors:

```
\drawmatrix[width=0]x
```

/drawmatrix/size=⟨*dimension*⟩ (style, no default)

Shortcut for setting both `/drawmatrix/height` and `/drawmatrix/width` to the same ⟨*dimension*⟩, resulting in a square matrix.

2.2 Shape

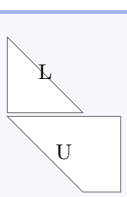
By default matrices are rectangular.

2.2.1 Triangular and Trapezoidal Matrices

/drawmatrix/lower (style, no value, initially unset)
/drawmatrix/upper (style, no value, initially unset)

Result in, respectively, lower- and upper-triangular matrices. Non-square matrices become trapezoidal.

```
\drawmatrix[lower]L
\drawmatrix[upper, width=1.5]U
```

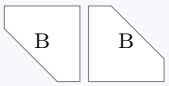


2.2.2 Banded Matrices

/drawmatrix/lower banded (style, no value, initially unset)
/drawmatrix/upper banded (style, no value)

Draw matrices as banded with bandwidth `0.3`.

```
\drawmatrix[lower banded]B
\drawmatrix[upper banded]B
```



/drawmatrix/banded (style, no value)

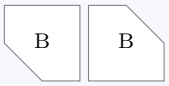
Shortcut for setting both `/drawmatrix/lower banded` and `/drawmatrix/upper banded`.

```
\drawmatrix[banded]B
```

/drawmatrix/lower bandwidth=⟨*dimension*⟩ (no default, initially empty)
/drawmatrix/upper bandwidth=⟨*dimension*⟩ (no default, initially empty)

The bandwidths, i.e., the horizontal/vertical extent from the diagonal.

```
\drawmatrix[lower bandwidth=.5]B
\drawmatrix[upper bandwidth=.5]B
```



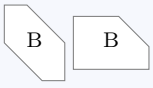
/drawmatrix/bandwidth=⟨*dimension*⟩ (style, no default)

Shortcut for setting both `/drawmatrix/lower bandwidth` and `/drawmatrix/upper bandwidth`.

```
\drawmatrix[bandwidth=.5]B
```

Banding on rectangular matrices applies to the smaller of the two dimensions:

```
\drawmatrix[banded, width=.8]B
\drawmatrix[upper banded, height=.7]B
```



`/drawmatrix/banded` can be combined with `/drawmatrix/lower` → P. 3 or `/drawmatrix/upper` → P. 3 to draw the intersection of both shapes.

```
\drawmatrix[banded, lower]L
```

2.2.3 Diagonal Matrices

/drawmatrix/diag (style, no value)

Shortcut for /drawmatrix/bandwidth→ P. 4=0.

```
\drawmatrix[diag]D
```

2.2.4 Super- and Subscripts

/drawmatrix/label base=⟨*text*⟩ (no default, initially empty)

Defines the label to be centered in the drawmatrix, and to which the actual /drawmatrix/label text→ P. 3 is aligned. This feature is useful to, e.g., draw centered labels with exponents:

```
$\drawmatrix[size=.5]{A^T}$
$\drawmatrix[size=.5, label base=A]{A^T}$
```

A^T A^T

/drawmatrix/label base anchor=⟨*anchor*⟩ (no default, initially base west)

The anchor of the /drawmatrix/label text→ P. 3 with respect to the /drawmatrix/label base.

```
$\drawmatrix[size=.5, label base=A, label ↩
      (base anchor=base east]{^0_1A}$
```

^0_1A

/drawmatrix/exponent=⟨*text*⟩ (style, no default)

Shortcut to add an exponent to matrix without offsetting the label. It sets the /drawmatrix/label base to the current /drawmatrix/label text→ P. 3 and adds the exponent ⟨*text*⟩ to /drawmatrix/label text→ P. 3.

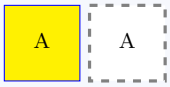
```
$\drawmatrix[size=.5, exponent=T]A$
```

A^T

2.3 Colors and Style

By default, matrices are drawn in gray and filled white. The Ti*k*Z keys draw=⟨*color*⟩ and fill=⟨*color*⟩ change these colors. In fact, all keys not recognized by this package are passed to the Ti*k*Z \filldraw command drawing the matrix.

```
\drawmatrix[fill=yellow, draw=blue]A
\drawmatrix[very thick, dashed]A
```



2.4 The Bounding Box

All matrices are contained in a rectangular bounding box.

`/drawmatrix/bbox` (style, no default, initially empty)

Add options to the Ti*k*Z `\node` that is the bounding box.. This is useful to, e.g., to visualize the 0 entries in the matrix:

```
\drawmatrix[lower, bbox/.append ↴
      {style={fill=blue!10}}]L
```

`/drawmatrix/bbox style={`⟨*style*⟩`}` (style, no default)

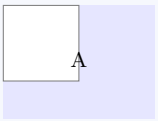
Shortcut for `/drawmatrix/bbox/.append style=`⟨*style*⟩.

`/drawmatrix/bbox height=`⟨*dimension*⟩ (no default, initially empty)

`/drawmatrix/bbox width=`⟨*dimension*⟩ (no default, initially empty)

Explicitly set the height and width of the bounding box. If unset, the bounding box is just large enough to contain the matrix. The label of the matrix (and thus the alignment with respect to the surrounding text) are by default fixed at the center[1] of the bounding box, while the matrix is positioned at its top-left corner.

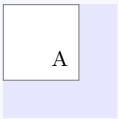
```
\drawmatrixset{bbox style={fill=blue!10}}

\drawmatrix[bbox width=2, bbox height=1.5]A
```



`/drawmatrix/bbox size=`⟨*dimension*⟩ (style, no default)

Shortcut for setting `/drawmatrix/bbox height` and `/drawmatrix/bbox width` to the same value.

```
\drawmatrixset{bbox style={fill=blue!10}}

\drawmatrix[bbox size=1.5]A
```



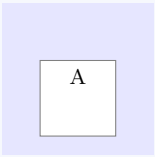
`/drawmatrix/offset height=`⟨*dimension*⟩ (no default, initially `0`)

`/drawmatrix/offset width=`⟨*dimension*⟩ (no default, initially `0`)

Sets the vertical and horizontal offset of the drawn matrix within its bounding box.

[1] See `/drawmatrix/label anchor` → P. 7.

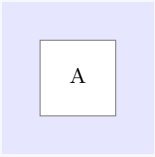
```
\drawmatrixset{bbox style={fill=blue!10}}

\drawmatrix[bbox size=2, offset width=.5, ↩
      ↪offset height=.75]A
```



/drawmatrix/offset=⟨*dimension*⟩ (style, no default)

Shortcut for setting `/drawmatrix/offset height` → P. 6 and `/drawmatrix/bbox width` → P. 6 to the same value.

```
\drawmatrixset{bbox style={fill=blue!10}}

\drawmatrix[bbox size=2, offset=.5]A
```



2.5 Coordinate System Transformations

/drawmatrix/scale=⟨*factor*⟩ (style, no default)

Scales all dimensions passed to a matrix. Can be used repeatedly to multiply scales

```
\drawmatrix[scale=.6]A
\drawmatrix[scale=.6, width=.5]B
\drawmatrix[scale=.7, scale=.7]B
```

/drawmatrix/x=⟨*value*⟩ (style, no default)
/drawmatrix/y=⟨*value*⟩ (style, no default)

Define the coordinate system for all unit-less dimensions.

```
\drawmatrix[x=.6cm, y=.4cm]A
\drawmatrix[x=.6cm, y=.4cm, width=1cm]B
```

2.6 Position of the Label and Baseline

By default, the label's `mid` is positioned at the bounding box's `center` and its `base` is used as the whole drawing's baseline.

/drawmatrix/label anchor=⟨*anchor*⟩ (style, no default, initially `mid`)

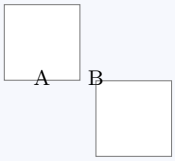
Set the anchor of label's Ti*k*Z `\node`.

```
\drawmatrix[label anchor=north]A
```

/drawmatrix/label pos=⟨*position*⟩ (style, no default, initially `bbox.center`)

Define the position of the label's Ti*k*Z `\node` within the picture. The following nodes and their anchors are available: `bbox` (the bounding box) and the `matrix` (matrix itself).

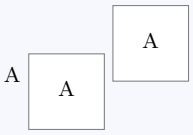
```
\drawmatrix[label pos=bbox.south]A
\drawmatrix[label pos=matrix.north west]B
```



/drawmatrix/baseline=⟨*position*⟩ (style, no default, initially `label.base`)

Specify how the picture is vertically aligned with the surrounding text's baseline. Options are the same anchors as for `/drawmatrix/label pos` and anchors of `label` (the label).

```
A
\drawmatrix[baseline=label.north]A
\drawmatrix[baseline=bbox.south]A
```



3 Changing Defaults

\drawmatrixset{⟨*options*⟩}

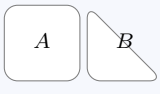
Applies options to to all following uses of `\drawmatrix` → P. 2 within the current scope.

```
\drawmatrixset{height=.5, lower}

$\drawmatrix A \; \drawmatrix B$
```

/drawmatrix/every picture (style, no value)
/drawmatrix/every bbox (style, no value)
/drawmatrix/every drawmatrix (style, no value)
/drawmatrix/every label (style, no value)
/drawmatrix/every label (style, no value)

Settings for all drawmatrix pictures, bounding boxes, matrices, and labels. Options should be added not with `/.style=` but with `/.append style=` to avoid messing with internals.

```
\drawmatrixset{every drawmatrix/.append ↴
      ↳style={rounded corners=5pt}}

$\drawmatrix A \; \drawmatrix[lower]B$
```



4 Externalization

`\drawmatrix`→P. 2 behaves as any other Ti*k*Z picture, therefore when externalization is enabled, all matrix visualizations are also externalized. However, since there are usually many `\drawmatrix`→P. 2 pictures, each of which is very small and fast to produce, their externalization would mean a tremendous overhead.

`/drawmatrix/externalize=true|false` (no default, initially `true`)

Explicitly disables externalization for all `\drawmatrix`→P. 2 pictures. It does not enable externalization.

5 Implementation

This section describes the implementation details of the drawmatrix package.

5.1 Package: Ti*k*Z

The tikz package is used for drawing.

```
\RequirePackage{tikz}
```

5.2 If for externalization

TEX `if` representing whether to explicitly disable Ti*k*Z externalization.

```
\newif\ifdrawmatrix@externalize
```

5.3 Key Declarations and Defaults

We rely on PGF keys as much as we can.

```
\pgfkeys{
```

Everything happens in the path `/drawmatrix`.

```
    drawmatrix/.is family,
    drawmatrix/.cd,
```

`picture` is the style for the `\tikzpicture` in which the matrix is drawn. `baseline` sets the baseline of the picture to a named coordinate of the matrix (default: base of the label).

```
    picture/.style={},
    path/.style={},
    baseline/.style={picture/.append style={baseline=(drawmatrix #1)}},
    scale/.style={path/.append style={scale=#1}},
    x/.style={path/.append style={x=#1}},
    y/.style={path/.append style={y=#1}},
    baseline=label.base,
```

`bbox` is the style of the bounding box, to which `bbox style` appends keys.

```
    bbox/.style={},
    bbox style/.style={bbox/.append style={#1}},
```

`bbox height` and `bbox width` don't have default values. `bbox size` sets them both to the same value.

```
    bbox height/.initial,
    bbox width/.initial,
    bbox size/.style={bbox height=#1, bbox width=#1},
```

`offset height` and `offset width` are `0` by default. `offset` sets them both to the same value.

```
    offset height/.initial=0,
    offset width/.initial=0,
    offset/.style={offset height=#1, offset width=#1},
```

`width` and `height` are `1` (Ti*k*Z unit) by default. `size` sets them both to the same value.

```
    height/.initial=1,
    width/.initial=1,
    size/.style={height=#1, width=#1},
```

The `lower bandwidth` and `upper bandwidth` don't have default values. `bandwidth` sets them both to the same value.

```
    lower bandwidth/.initial,
    upper bandwidth/.initial,
    bandwidth/.style={lower bandwidth=#1, upper bandwidth=#1},
```

`lower banded` and `upper banded` are shortcuts to set the corresponding bandwidths to the default value of `0.3` (Ti*k*Z units). `banded` sets them both.

```
    lower banded/.style={lower bandwidth=.3},
    upper banded/.style={upper bandwidth=.3},
    banded/.style={lower banded, upper banded},
```

`lower` and `upper` are implemented by setting the opposite bandwidth to `0`. `diag` sets them both.

```
    lower/.style={upper bandwidth=0},
    upper/.style={lower bandwidth=0},
    diag/.style={bandwidth=0},
```

`label` is the style for the label with the text `label text`. `label pos` sets the label at a named coordinate of the matrix (default: center of the bounding box). `label anchor` sets the label's `anchor` (default: in the middle).

```
    label text/.initial,
    label/.style={},
    label pos/.style={label/.append style={at=(drawmatrix #1)}},
    label pos=bbox.center,
    label anchor/.style={label/.append style={anchor=#1}},
    label anchor=mid,
```

`label base` and `label base anchor` allow to ofset labels with exponents.

```
    label base/.initial,
    label outer/.style={},
    label base anchor/.style={label outer/.append style={
        anchor=#1, at=(drawmatrix label.#1)
```

```
    }},
    label base anchor=base west,
```

`exponent` is a shortcut to add an expontent to the label text without using the `label base`.

```
    exponent/.style={
        label base/.expanded=\pgfkeysvalueof{/drawmatrix/label text},
        label text/.append=^{#1}
    },
```

Unknown keys are collected in `/drawmatrix/drawmatrix`.

```
    drawmatrix/.style={},
    .unknown/.code={%
        \let\dm@currname\pgfkeyscurrentname%
        \let\dm@currval\pgfkeyscurrentvalue%
        \ifx#1\pgfkeysnovalue\pgfkeysalso{
            drawmatrix/.append style/.expand once={\dm@currname}
        }\else\pgfkeysalso{
            drawmatrix/.append style/.expand twice={%
                \expandafter\dm@currname\expandafter=\dm@currval%
            }
        }\fi%
    },
```

The default style for matrices: `every picture` applies to all `\tikzpictures` the matrices are drawn in, `every bbox` applies to all bounding boxes, `every drawmatrix` applies to the matrices themselves, and `every label` applies to the labels.

```
    every picture/.style={},
    every bbox/.style={
        name=drawmatrix bbox,
        inner sep=0
    },
    every drawmatrix/.style={
        fill=white,
        draw=gray
    },
    every label/.style={
        name=drawmatrix label,
        outer sep=0,
        inner sep=0
    },
    every node/.style={
        name=drawmatrix matrix,
        outer sep=0,
        inner sep=0,
        anchor=north west,
        at=(drawmatrix north west)
    },
```

`externalize` sets a TEX if (default: `true` = behave as all pictures).

```
    externalize/.is if=drawmatrix@externalize,
    externalize=true,
```

5.4 Hooks

Hooks for inserting code at various points of the render.

```
    pre code/.code={},
    post setup code/.code={},
    pre coordinate code/.code={},
    post coordinate code/.code={},
    pre draw code/.code={},
    post draw code/.code={},
    pre label code/.code={},
    post label code/.code={},
    post code/.code={},
```

5.5 Code Keys

The following keys contain the code that construct the drawmatrix.

Prepare the label text and, if needed label outer text (for alignment). This needs to be outside the tikzpicture to properly detect math mode.

```
    setup label code/.code={
    \ifmmode\edef\dm@labeltext{$\dm@labeltext$}\fi%
    \expandafter\ifx\dm@labelbase\pgfkeysnovalue\else%
        \let\dm@labeltextouter\dm@labeltext%
        \edef\dm@labelbase{%
            \ifmmode$\dm@labelbase$\else\dm@labelbase\fi%
        }%
        \def\dm@labeltext{\phantom{\dm@labelbase}}%
    \fi%
    },
```

Disable externalization if `externalize=false`.

```
    externalization code/.code={
        \ifdrawmatrix@externalize\else%
            \ifx\tikz@library@external@loaded\undefined\else%
                \tikzset{external/export=false}%
            \fi%
        \fi%
    },
```

Parse width, height, the minimum dimension and zero for comparison purposes.

```
    setup sizes code/.code={
        \path[/drawmatrix/path] (\dm@width, \dm@height);
        \pgfgetlastxy\dm@width\dm@height
        \path[/drawmatrix/path] (\dm@offsetwidth, \dm@offsetheight);
        \pgfgetlastxy\dm@offsetwidth\dm@offsetheight
        \pgfmathsetlengthmacro\dm@minsize{min(\dm@width, \dm@height)}
        \pgfmathsetlengthmacro\dm@zero{0.0}
```

Prepare the band widths: First, if the matrix is not banded, the bandwidth is set to the smaller matrix dimension. Then, the band width is limited by this smaller dimension.

```
        \expandafter\ifx\dm@lowerbandwidth\pgfkeysnovalue
            \def\dm@lowerbandwidth{\dm@minsize}
        \else
            \path[/drawmatrix/path] (\dm@lowerbandwidth, 0);
            \pgfgetlastxy\dm@lowerbandwidth\dm@zero
        \fi
        \expandafter\ifx\dm@upperbandwidth\pgfkeysnovalue
            \def\dm@upperbandwidth{\dm@minsize}
        \else
            \path[/drawmatrix/path] (0, \dm@upperbandwidth);
            \pgfgetlastxy\dm@zero\dm@upperbandwidth
        \fi
        \pgfmathsetlengthmacro\dm@lowerbandwidth{
            min(\dm@minsize, \dm@lowerbandwidth)
        }
        \pgfmathsetlengthmacro\dm@upperbandwidth{
            min(\dm@minsize, \dm@upperbandwidth)
        }
```

Set the default bounding box size.

```
        \expandafter\ifx\dm@bboxheight\pgfkeysnovalue
            \pgfmathsetlengthmacro\dm@bboxheight{
                \dm@height + \dm@offsetheight
            }
        \else
            \path[/drawmatrix/path] (0, \dm@bboxheight);
            \pgfgetlastxy\dm@zero\dm@bboxheight
        \fi
        \expandafter\ifx\dm@bboxwidth\pgfkeysnovalue
            \pgfmathsetlengthmacro\dm@bboxwidth{
                \dm@width + \dm@offsetwidth
            }
        \else
            \path[/drawmatrix/path] (\dm@bboxwidth, 0);
            \pgfgetlastxy\dm@bboxwidth\dm@zero
        \fi
    },
```

Reset the bounding box and begin with (drawing) the path for the bounding box.

```
    bbox code/.code={
        \pgfresetboundingbox
        \node[/drawmatrix/every bbox, /drawmatrix/bbox,
            minimum height=\dm@bboxheight,
            minimum width=\dm@bboxwidth] {};
    },
```

Whether needed or not, declare all matrix corners.

```
    coordinate code/.code={
        \path (drawmatrix bbox.north west)
            ++(\dm@offsetwidth, -\dm@offsetheight)
```

```
            ++(.5\pgflinewidth, -.5\pgflinewidth)
            coordinate (drawmatrix north west)
            ++(\dm@width, 0)
            +(-\dm@minsize + \dm@upperbandwidth, 0)
            coordinate (drawmatrix north)
            +(0, -\dm@minsize + \dm@upperbandwidth)
            coordinate (drawmatrix east)
            ++(0, -\dm@height)
            coordinate (drawmatrix south east)
            ++(-\dm@width, 0)
            +(\dm@minsize - \dm@lowerbandwidth, 0)
            coordinate (drawmatrix south)
            +(0, \dm@minsize - \dm@lowerbandwidth)
            coordinate (drawmatrix west);
    },
```

Add an invisible node the size of the matrix.

```
    node code/.code={
        \node[/drawmatrix/every node,
            minimum height=\dm@height,
            minimum width=\dm@width] {};
    },
```

Now, draw only what is needed of the matrix. Otherwise path modifications (e.g., such as rounded corners) might not work.

```
    draw code/.code={
        \filldraw[/drawmatrix/every drawmatrix, /drawmatrix/drawmatrix]
            (drawmatrix north west)
            \ifx\dm@upperbandwidth\dm@zero
                \ifx\dm@width\dm@minsize\else -- (drawmatrix north) \fi
                \ifx\dm@height\dm@minsize\else -- (drawmatrix east) \fi
            \else
                -- (drawmatrix north)
                \ifx\dm@upperbandwidth\dm@minsize\else
                    -- (drawmatrix east)
                \fi
            \fi
            -- (drawmatrix south east)
            \ifx\dm@lowerbandwidth\dm@zero
                \ifx\dm@width\dm@minsize\else -- (drawmatrix south) \fi
                \ifx\dm@height\dm@minsize\else -- (drawmatrix west) \fi
            \else
                -- (drawmatrix south)
                \ifx\dm@lowerbandwidth\dm@minsize\else
                    -- (drawmatrix west)
                \fi
            \fi
            -- cycle;
    },
```

The label.

```
    label code/.code={
        \node[/drawmatrix/every label, /drawmatrix/label]
            {\dm@labeltext};
        \expandafter\ifx\dm@labelbase\pgfkeysnovalue\else
            \node[/drawmatrix/every label, /drawmatrix/label,
                  /drawmatrix/label outer] {\dm@labeltextouter};
        \fi
    }
}
```

5.6 User Macros

`\drawmatrixset` → P. 8 as a simple shortcut like `\tikzset`.

```
\newcommand\drawmatrixset[1]{\pgfqkeys{/drawmatrix}{#1}}
```

Here we go, the main thing: `\drawmatrix` → P. 2. First, apply the options and extract the sizes from the PGF keys.

```
\newcommand\drawmatrix[2][]{{%
    \drawmatrixset{
        label text={#2},
        #1,
        label text/.get=\dm@labeltext,
        height/.get=\dm@height,
        width/.get=\dm@width,
        lower bandwidth/.get=\dm@lowerbandwidth,
        upper bandwidth/.get=\dm@upperbandwidth,
        offset height/.get=\dm@offsetheight,
        offset width/.get=\dm@offsetwidth,
        bbox height/.get=\dm@bboxheight,
        bbox width/.get=\dm@bboxwidth,
        label base/.get=\dm@labelbase,
        pre code,
        setup label code,
        externalization code
    }%
    \begin{tikzpicture}[/drawmatrix/every picture, /drawmatrix/picture]
        \drawmatrixset{
            setup sizes code,
            post setup code,
            bbox code,
            pre coordinate code,
            coordinate code,
            post coordinate code,
            node code,
            pre draw code,
            draw code,
            post draw code,
            pre label code,
            label code,
            post label code
```

```
248         }
249     \end{tikzpicture}%
250     \drawmatrixset{post code}%
251 }}
```

Index