

matrix.skeleton's Manual

Nicolas Dubebout

1 Introduction

The `TikZ matrix` library places nodes on a grid. However, this grid is discarded after the nodes have been placed. As a result, certain constructions involving multiple nodes become cumbersome. The following two examples highlight some of the difficulties.

1.1 Alignment Issues with `fit`

The `fit` library is used to highlight a subset of nodes in a matrix. If all the nodes in the matrix have the same dimension, as in Figure 1, `fit` produces the desired output.

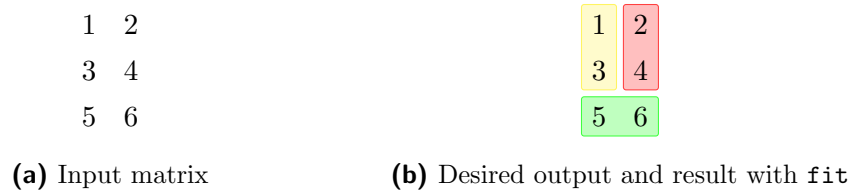


Figure 1. Highlighting in a matrix with nodes of identical dimensions

However, if the nodes have different heights and widths, as illustrated in Figure 2, some alignment issues arise.

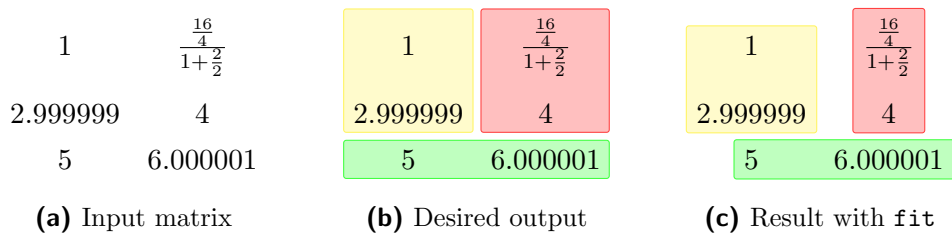


Figure 2. Highlighting in a matrix with nodes of different dimensions

These problems can be addressed using `minimum width` and `minimum height`. However, adjusting manually these parameters in every matrix is a waste of time.

The `matrix.skeleton` library provides a clean solution through the use of nodes called `cells`. These `cells` and other skeleton nodes are described in Section 2.

1.2 Working with Rows and Columns

The readability of a matrix can sometimes be improved by adding a background on every other row. This simple task is not easily achievable with `matrix` alone. The style `every odd column` only affects the nodes of the said columns. There is no real column object to work with.

The `matrix.skeleton` library provides TikZ styles to achieve this goal easily. These styles are described in Section 3

2 Skeleton

2.1 Nodes

`matrix.skeleton` works by positioning a set of nodes to recreate the `matrix` grid. The eight types of such nodes are illustrated in Figure 3.

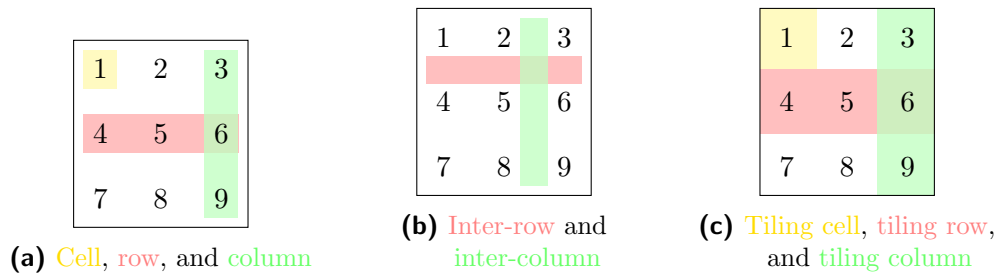


Figure 3. Skeleton nodes

2.2 Using `matrix.skeleton`

The recommended way of using `matrix.skeleton` is through TikZ. First, load the library with:

```
\usetikzlibrary{matrix.skeleton}
```

Then add an option to your matrix:

```
\matrix (m) [label skeleton] {...};
```

This creates a set of nodes that can be used for styling. For example, the nodes illustrated in Figure 3 are named: `m-cell-1-1`, `m-row-2`, `m-column-3`, `m-inter-row-1`, `m-inter-column-2`, `m-tiling-cell-1`, `m-tiling-row-2`, and `m-tiling-column-3`.

3 Styling

The skeleton nodes are PGF nodes not meant to be styled. Styles should be applied to nodes whose shapes depend on the skeleton ones.

3.1 Macros

Styling in `matrix.skeleton` is done with the `fit` library. The following macro creates a `fit` node with the specified style:

```
\fitandstyle{(m-cell-1-1) (m-cell-2-2)}{draw=red};
```

It takes an optional argument to place the node in a `pgfonlayer` environment:

```
\fitandstyle[background]{(m-cell-1-1) (m-cell-2-2)}{fill=red};
```

3.2 TikZ matrix Options

Common styling options are also provided as TikZ options. These options call `label skeleton` before styling the appropriate nodes. They take the following form:

```
\matrix (m) [style odd rows = {draw=red}] {...};
```

```
\matrix (m) [style odd tiling rows = {draw=red}] {...};
```

```
\matrix (m) [style grid = {draw}] {...};
```

```
\matrix (m) [style tiling grid = {draw}] {...};
```

All of these options have an `on layer` variant taking the following form:

```
\matrix (m) [style odd rows on layer = {background}{fill=red}] {...};
```

4 Examples

The following examples illustrate the styling capabilities offered by `matrix.skeleton`.

4.1 Grid

1	$\frac{\frac{16}{4}}{1+\frac{2}{2}}$	3
3.999999	5	6
7	8.000001	$3 + 3 + 3$

```
\begin{tikzpicture}
\matrix (m) [matrix of math nodes, style contour = {draw, very thick},
             style grid = {draw, thin}] {
1          & \frac{\frac{16}{4}}{1 + \frac{2}{2}} & 3          & \\
3.999999 & 5          & 6          & \\
7          & 8.000001   & 3 + 3 + 3 & \\
};
\end{tikzpicture}
```

4.2 Rows

1	$\frac{\frac{16}{4}}{1+\frac{2}{2}}$	3
3.999999	5	6
7	8.000001	$3 + 3 + 3$

```

\begin{tikzpicture}
\matrix (m) [matrix of math nodes, row sep = 10pt,
             style odd rows on layer={background}{fill=green!25},
             style even rows on layer={background}{fill=yellow!30}] {
1          & \frac{\frac{16}{4}}{1 + \frac{2}{2}} & 3          & \\
3.999999 & 5          & 6          & \\
7          & 8.000001   & 3 + 3 + 3 & \\
};

\fitandstyle{(m-inter-row-1)}{fill=red!25}
\fitandstyle{(m-inter-row-2)}{fill=red!25}
\end{tikzpicture}

```

4.3 Checker Board

This example is inspired by the following [T_EX - L^AT_EX Stack Exchange](#) question: [How can I set the background color of the rows and columns of a matrix node in Tikz?](#)

	①		①
		②	②
②			②
③			③

```

\begin{tikzpicture}
\matrix (m) [draw, matrix of nodes, row sep=2mm, column sep=1mm,
             nodes={draw, thick, circle, inner sep=1pt}, label skeleton] {
  & 1 & & [2mm] | [gray] | 1 \\
  & 2 & & | [gray] | 2 \\
  | [gray] | 2 & & & | [gray] | 2 \\
  3 & & & 3 \\
};
\foreach \row in {1, ..., 4} {
  \foreach \col in {1, ..., 4} {
    \pgfmathparse{Mod(\row + \col, 2) ? "red!25" : "yellow!30"}
    \colorlet{squarebg}{\pgfmathresult}
    \fitandstyle[background]{(m-tiling-cell-\row-\col)}{fill = squarebg}
  }
}
\end{tikzpicture}

```

5 Internals

`matrix.skeleton` was heavily inspired by [Andrew Stacey's](#) `matrixcells` L^AT_EX package. It has three distinctive features. First, it works with any `anchor`. Second, it provides finer control with respect to `row sep`, `column sep`, and `inner sep`. Third, the skeleton node positioning relies only on T_EX and PGF, not on L^AT_EX or TikZ.

`matrixcells` properly aligns its `cells` when the node `anchor` is `base`. However, when the alignment is different it runs into problems, as exposed in the following [T_EX - L^AT_EX Stack Exchange](#) question: [Matrixcells problem with the y-axis only](#). This shortcoming is the result of some loss of information in `pgfmodulematrix.code.tex`. A dimension used during the placement of nodes is overwritten. Therefore, this information is not available to build the grid. In `matrixcells`, this lost dimension is reconstructed as the average of two other dimensions. This method only gives the right dimension when the nodes are anchored at `base`. To always get proper alignment, the `pgfmodulematrix.code.tex` macro erasing the dimension was rewritten. Following [@percusse's](#) recommendation this change is transparent to the user and does not require updating PGF/TikZ.

`matrixcells` only provides `cells` corresponding the `tiling-cells` in `matrix.skeleton`. This tiling behavior is sometimes desired. However, it can result in unexpected behaviors when: using a non-base `anchor`, using `row sep` or `column sep`, or when working on boundary nodes.

6 Known Bugs

`matrix.skeleton` has one known bug, relating to multiple matrices.

6.1 Does not Work with Multiple Matrices

`matrix.skeleton` does not work on figures with multiple matrices. The skeleton is correctly constructed for the first matrix but not for subsequent ones. For them, the skeleton has the correct shape but is not positioned correctly. A number of examples were shared in [this Github issue](#). The example below was created to test solutions against.

Onur Tolga Şehitoğlu has made progress on [a promising solution](#). It does not currently handle the case where the tallest element in the last row is not in the first non-empty column.



$$\begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & \frac{1}{a'_{22}} \end{pmatrix} \begin{pmatrix} a_{11} \cdot b_1 + a_{12} \cdot b_2 \\ a_{21} \cdot b_1 + a_{22} \cdot b_2 \end{pmatrix}$$

```
\begin{tikzpicture}
\tikzset{my-matrix/.style={matrix of math nodes, label skeleton,
left delimiter=(, right delimiter =)}}

\matrix (A) [my-matrix] {
a_{11} & a_{12} \\
a_{21} & \frac{1}{a'_{22}} \\
};

\matrix (C) [my-matrix, right= of A] {
a_{11} \cdot b_1 + a_{12} \cdot b_2 \\
a_{21} \cdot b_1 + a_{22} \cdot b_2 \\
};

\matrix (B) [my-matrix, above=of C] {
b_1 \\
b_2 \\
};

\fitandstyle{(A-row-1)}{draw=red, fill=red!25, opacity=0.5}
\fitandstyle{(A-row-2)}{draw=blue, fill=blue!25, opacity=0.5}
\fitandstyle{(B-column-1)}{draw=yellow, fill=yellow!25, opacity=0.5}
```

```
\fitandstyle{(C-cell-1-1)}{draw=orange, fill=orange!25, opacity=0.5}  
\fitandstyle{(C-cell-2-1)}{draw=green, fill=green!25, opacity=0.5}  
\end{tikzpicture}
```