1 Introduction

The Ti\textit{k}Z \texttt{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 \texttt{fit}

The \texttt{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, \texttt{fit} produces the desired output.

\begin{figure}[h]
\centering
\begin{tabular}{ccc}
1 & 2 & 1 & 2 \\
3 & 4 & 3 & 4 \\
5 & 6 & 5 & 6
\end{tabular}
\caption{Input matrix (a) \hspace{1cm} Desired output and result with \texttt{fit} (b)}
\end{figure}

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

\begin{figure}[h]
\centering
\begin{tabular}{ccc}
1 & \dfrac{16}{1+\frac{2}{3}} & 1 & \dfrac{16}{1+\frac{2}{3}} \\
2.999999 & 4 & 2.999999 & 4 \\
5 & 6.000001 & 5 & 6.000001
\end{tabular}
\caption{Input matrix (a) \hspace{1cm} Desired output (b) \hspace{1cm} Result with \texttt{fit} (c)}
\end{figure}

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

The \texttt{matrix.skeleton} library provides a clean solution through the use of nodes called \texttt{cells}. These \texttt{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](image)

(a) Cell, row, and column  
(b) Inter-row and inter-column  
(c) Tiling cell, tiling row, and tiling column

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 \texttt{matrix.skeleton} is done with the \texttt{fit} library. The following macro creates a \texttt{fit} node with the specified style:

\begin{verbatim}
\fitandstyle{(m-cell-1-1) (m-cell-2-2)}{draw=red};
\end{verbatim}

It takes an optional argument to place the node in a \texttt{pgfonlayer} environment:

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

3.2 TikZ matrix Options

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

\begin{verbatim}
\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}] {...};
\end{verbatim}

All of these options have an \texttt{on layer} variant taking the following form:

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

4 Examples

The following examples illustrate the styling capabilities offered by \texttt{matrix.skeleton}.

4.1 Grid

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

\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{16}{1+2^2} & 3 \\
3.999999 & 5 & 6 \\
7 & 8.000001 & 3 + 3 + 3 
};
\end{tikzpicture}

4.3 Checker Board

This example is inspired by the following \LaTeX Stack Exchange question: How can I set the background color of the rows and columns of a matrix node in Tikz?
\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 \LaTeX 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 \TeX and PGF, not on \LaTeX or Ti\kZ.

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 \TeX - \LaTeX 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/Ti\kZ.

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.