The DuckBoat — Beginners' Pond: You do not need to be Neo to cope with a Ti*k*Z matrix

Herr Professor Paulinho van Duck

Abstract

In this installment, Prof. van Duck will show you some tips & tricks about the useful Ti*k*Z library `matrix`.

1 I am back!

Hi, (LA)TEX friends!

Did you miss me? Don't worry; I am back as quacky as usual!

Lately, I have been very busy at work. Moreover, I am helping my friend Carla with her thesis. Hence, I haven't much time to dedicate to the DuckBoats, so I decided to write only one article a year.

I have also changed the title of my column; now it is "Beginners' Pond," to better highlight the targeted audience.

Indeed, my articles may appear too trivial for a serious journal like the *TUGboat*, but, as the volleyball coach Julio Velasco once said, "there are no *easy* or *difficult*, but things someone *can* do or *cannot!*"

The name changing is also a way not to be linked only to TEX.SE, since new Q&A sites are emerging.

For example, as Barbara announced in her column, I would call your attention to Top Answers (`https://topanswers.xyz/tex`). The platform itself is open source, developed with the community in mind and totally non-profit. The TEX community there is still small, but eager to answer your questions, and would be more than happy to see new users. So if this sounds interesting to you, just drop by and have a look yourself.

My commitments did not prevent me from attending the GUIT *meeting 2019*, where I had the occasion to meet Bär once again, together with many other friends like Ulrike Fischer, her husband Gert, and prof. Enrico Gregorio.

It was a very special event because we celebrated prof. Claudio Beccari, who will retire from the group's official positions due to age limits, but he surely will not give up helping everyone on the Forum.

This time, I will show you a useful Ti*k*Z library, `matrix`. It allows building matrices of Ti*k*Z objects. It could be a convenient alternative for node positioning, and it is also useful, for example, to create math matrices with some graphical adding.

Before getting to the heart of the matter, I would like to highlight a handy post on TEX.SE Meta, which collects the most often referenced questions: `https://tex.meta.stackexchange.com/questions/2419/often-referenced-questions`.

They are grouped by topic, so it is straightforward to find what you are looking for; most of the posts listed should be a "must-read" for any beginner.

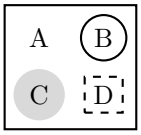
Hence, take a look at the *Often referenced questions* post on Meta before asking, maybe you will find a solution at once.

Last but not least, both newbies and experts are invited to contribute to keeping it up-to-date!

2 Quack Guide No. 5 The (Ti*k*Z) Matrix

A Ti*k*Z matrix is analogous to an ordinary `tabular`, but its elements are Ti*k*Z objects.

For example, you can build a table with nodes, paths, whatever you prefer:




```
\documentclass{standalone}
\usepackage{tikz}
\usetikzlibrary{matrix}
\begin{document}
  \begin{tikzpicture}
    \matrix[
      draw, thick,
      column sep=.2cm, row sep=.1cm
      ] {
      \node {A}; &
        \draw (0,0) circle (.3cm) node {B};\\
      \node[fill=gray!30, circle] {C}; &
        \node[draw, dashed] {D};\\
      };
  \end{tikzpicture}
\end{document}
```

Like any other Ti*k*Z library, the first thing is to load it with `\usetikzlibrary{matrix}`, after having loaded the package Ti*k*Z itself.

Then, in your `tikzpicture` environments, you will be able to use the command:

`\matrix[`⟨*options*⟩`] (`⟨*name*⟩`) at (`⟨*coords*⟩`)`
`{`⟨*content*⟩`};`

As usual, you have some ⟨*options*⟩ in square brackets. For example, `column sep` and `row sep` set the space between the columns/rows. Since the matrix itself is a node, you may use other node options, such as `draw` or `thick`.

Note that here they refer to the matrix, they are not always inherited by the nodes it contains. For instance, the option `thick` has effect both on the matrix and its elements (see B and D in the previous image); whereas the `draw` only draws the border *of the matrix* and the path (B), but not the borders of the `\node`s A and C. You can use `node={`⟨*options*⟩`}` to make some styles apply to every node of a matrix (or, in general, of a `tikzpicture`).

You may also give your matrix a ⟨*name*⟩, if needed, or position it at desired coordinates, ⟨*coords*⟩.

In the ⟨*content*⟩, there are the cells of your matrix, separated by `&`, and with `\\` for ending the rows, as in a usual table. But pay attention: the `\\` is mandatory also for the last row.

Another difference with respect to a `tabular` is that you do not have to state in advance the number of columns; you can add or remove as many columns you like, without the need to change any specification.

If the elements of your matrix are all nodes, you may use the option `matrix of nodes`, and put only the *text* of the nodes in the cells.

Indeed, it would be annoying to write
`\node{...};`
in every cell, but LaTeX is fun, quack! Boring activities are avoided as much as cats avoid water!

Matrices of nodes are by far the most popular; I will show you some examples in the following.

2.1 A convenient way for positioning

In the DuckBoat "The Morse code of Ti*k*Z" [1], I used the `positioning` library to place a node above/below/left/right to another.

It can be done very easily also with a Ti*k*Z matrix. The advantage of using a matrix is that, if you have to add a node between other existing nodes, you do not have to change the positioning option of the other nodes.

Let me explain better with an example. Suppose you are using `right = of ...` to position two nodes in the following way:

A B

```
\documentclass{standalone}
\usepackage{tikz}
\usetikzlibrary{positioning}
\begin{document}
  \begin{tikzpicture}[nodes={draw}]
    \node (A) {A};
    \node[right= of A] {B};
  \end{tikzpicture}
\end{document}
```

The same result can be achieved also with a Ti*k*Z matrix:[1]

```
\usetikzlibrary{matrix} % <-- in preamble
...
\begin{tikzpicture}
  \matrix[
    matrix of nodes,
    nodes={draw},
    column sep=1cm
    ] {
    A & B\\
    };
\end{tikzpicture}
```

Now, if you would like to add a node in the middle, with the `positioning` library you have to change the reference node of B:

A Middle B

```
\usetikzlibrary{positioning} % <-- in preamble
...
\begin{tikzpicture}[nodes={draw}]
  \node (A) {A};
  \node[right= of A] (midnode) {Middle};
  \node[right= of midnode] {B};
\end{tikzpicture}
```

Of course, with only two nodes, it is not a big effort, but imagine to have a more complex picture, with many nodes linked together, it could become complicated, and tedious.

With a Ti*k*Z matrix you can simply add the new node as a new element of the matrix:

```
\usetikzlibrary{matrix} % <-- in preamble
...
\begin{tikzpicture}
  \matrix[
    matrix of nodes,
    nodes={draw},
    column sep=1cm
    ]{
    A & Middle & B\\
    };
\end{tikzpicture}
```

It is convenient, is it not?

As I said earlier, with `column/rows sep` you can set the width between columns/rows, the same value for all the columns/rows of your matrix.

If you would like to increase or decrease the distance between two specific rows, use `[`⟨*height*⟩`]` after `\\`, as in an ordinary table.

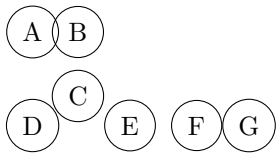
There is a little difference, instead, in the way to set the distance between two specific columns. In a

[1] Hereinafter, for convenience, I will show only the `tikzpicture` environments, unless something more is needed.

Ti*k*Z matrix, you cannot use the @-expression from `array` package (e.g., `@{\hspace{`⟨*width*⟩`}}`), because the parameter for table specification is not present.

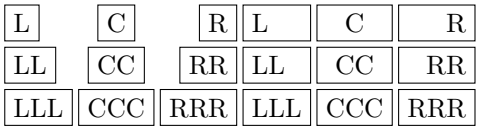
You can achieve the same result by adding, in the first row, the option `[`⟨*width*⟩`]` after the `&` which separates the two columns involved. If the first row has fewer elements of the other rows, you have to add the necessary empty elements to use this feature.

An example is worth a thousand words:




```
\begin{tikzpicture}
  \matrix[matrix of nodes,
    nodes={draw,circle}]{
      A &[-1mm] B  & &[2mm] \\[1.5mm]
      & C \\[-3.1mm]
      D &  & E & F & G\\};
\end{tikzpicture}
```

Since there is no table specification, you cannot set the column alignment with `l`, `c`, or `r` as usual, but you may use anchors or give a dimension to the nodes and align the text inside. Let's draw the nodes to see the difference between the two approaches:




```
\begin{tikzpicture}
  \matrix[
    matrix of nodes,
    nodes={draw},
    column sep=2pt, row sep=2pt,
    column 1/.style={nodes={anchor=base west}},
    column 2/.style={nodes={anchor=base}},
    column 3/.style={nodes={anchor=base east}},
    column 4/.style={
      nodes={text width=width("LLL"),
        align=left}},
    column 5/.style={
      nodes={text width=width("CCC"),
        align=center}},
    column 6/.style={
      nodes={text width=width("RRR"),
        align=right}},]{
    L   & C   & R   & L   & C   & R\\
    LL  & CC  & RR  & LL  & CC  & RR\\
    LLL & CCC & RRR & LLL & CCC & RRR\\
    };
\end{tikzpicture}
```

Please note that the option
`column` ⟨*m*⟩`/.style={`⟨*options*⟩`}`
sets a style for all the nodes in column ⟨*m*⟩, and the expression `width("`⟨*string*⟩`")` is the Ti*k*Z equivalent of `calc`'s `\widthof` to get the width of the box containing ⟨*string*⟩. There are also the analogous functions `height("`⟨*string*⟩`")` and `depth("`⟨*string*⟩`")`.

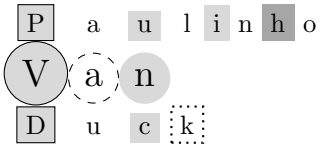
2.2 Formatting columns, rows, single cells

In the previous examples, we have already encountered `nodes={`⟨*options*⟩`}`, an abbreviation for
`every node/.append style=`⟨*options*⟩
(the `append` means that the style is added to the already existing options). We have also seen how to format the nodes of a column.

There are similar features for formatting a row or a specific cell. There are also options for formatting every even or odd column/row.

In the case of matrices of nodes, there is also an alternative way to set the options of a single node: putting `|[`⟨*options*⟩`]|` just before the node text.

An example:

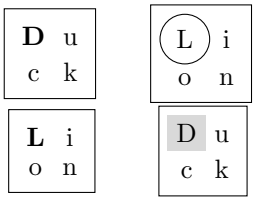



```
\begin{tikzpicture}
  \matrix[
    matrix of nodes,
    column 1/.style={nodes={draw}},
    row 2/.style={nodes={circle}},
    row 2 column 2/.style={
      nodes={draw, dashed}
      },
    every odd column/.style={
      nodes={fill=gray!30}
      },
    every even row/.style={
      nodes={font=\Large}
      },
    ]{
    P&a&u&l&i&n&|[fill=gray!70]|h&o\\
    V&a&n\\
    D&u&c&|[draw, dotted, thick]|k\\
    };
\end{tikzpicture}
```

If you have many matrices with the same formatting, you can create a style for the whole matrix and use it whenever you need it, or set a style for all your matrices with
`every matrix/.style={`⟨*options*⟩`}`.

As usual, you may set the style with `\tikzset`, if you'd like to use it in several of your Ti*k*Z pictures throughout your document, or as an option of the

specific `tikzpicture` environment where the style is needed.



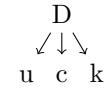

```
...
\tikzset{
% styles applied throughout the document
  every matrix/.style={
    matrix of nodes, draw
    },
  matrdoc/.style={
    row 1 column 1/.style={
      nodes={font=\bfseries}
      }
    },
  }
...
\begin{document}
\begin{tikzpicture}[
% style applied throughout this tikzpicture
  matrpic/.style={
    row 1 column 1/.style={
    nodes={draw, circle}}
    }
  ]
  \matrix[matrdoc] {
    D&u\\c&k\\
    };
  \matrix[matrpic] at (2,0) {
    L&i\\o&n\\
    };
\end{tikzpicture}

\begin{tikzpicture}[
% style applied throughout this tikzpicture
  matrpic/.style={
    row 1 column 1/.style={
      nodes={fill=gray!30}
      }
    }
  ]
  \matrix[matrdoc] {
    L&i\\o&n\\
    };
  \matrix[matrpic] at (2,0) {
    D&u\\c&k\\
    };
\end{tikzpicture}
\end{document}
```

2.3 Naming matrix elements

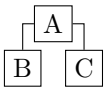
Once you give a ⟨*name*⟩ to your matrix, adding it in round brackets or with the option `name=`⟨*name*⟩, you can reference any cell with ⟨*name*⟩-⟨*n*⟩-⟨*m*⟩, where ⟨*n*⟩ is the row number of the cell and ⟨*m*⟩ the column number.

This notation is very convenient if you need to perform a repeated action on your cells:




```
\begin{tikzpicture}
  \matrix[
    matrix of nodes,
    text height=height("k")
    ] (mymatr) {
    &D\\[3mm]
    u&c&k\\
    };
  \foreach \myind in {1,2,3}
    \draw[->] (mymatr-1-2) --
      (mymatr-2-\myind);
\end{tikzpicture}
```

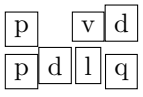
In addition, sometimes it is useful to give a mnemonic name to a node; you can do this by putting `|[name=`⟨*name*⟩`]|` before the node text:




```
\begin{tikzpicture}
  \matrix[
    matrix of nodes, nodes=draw
    ]{
    &[-1mm] |[name=start]|A &[-1mm]\\[1mm]
    |[name=nodeb]|B & & |[name=nodec]|C\\
    };
  \draw (start) -| (nodeb);
  \draw (start) -| (nodec);
\end{tikzpicture}
```

2.4 Tips & tricks about cell dimensions and borders

When the nodes are drawn, you would expect to have cells with homogeneous dimension, but the result could seem strange:




```
\begin{tikzpicture}
  \matrix[matrix of nodes, nodes={draw}]{
    p&&v&d\\p&d&l&q\\
    };
\end{tikzpicture}
```

It happens because Ti*k*Z builds the bounding box according to the node content.

Explicitly setting the node `minimum width` or `text width` solves the problem for that dimension, but you cannot do the same for the height.

Well, if you read "The Morse code of Ti*k*Z", you already know the solution: use `text depth` and `text height`.

Another thing that, at first sight, may seem strange is that the empty cells have no border. It happens because Ti*k*Z considers the empty cells... empty!

If you would like to have borders everywhere, you can use the option `nodes in empty cells`.

p		v	d
p	d	l	q

```
\begin{tikzpicture}
  \matrix[
    matrix of nodes,
    nodes in empty cells,
  nodes={draw,
      text depth=.14cm,
      text height=.3cm,
      minimum width=.7cm}
    ]{
    p&&v&d\\
    p&d&l&q\\
    };
\end{tikzpicture}
```

You can also fill the empty cell with a default value, using `execute at empty cell=`⟨*code*⟩.

However, there is still a little problem; the inner borders are drawn "twice", so they are thicker than the contour.

There is a little trick to solve this situation, using `-\pgflinewidth` as column/row separation.

Indeed, the border's width is stored in the PGF macro `\pgflinewidth`; if you decrease the column and row separators by it, you will have only one line drawn, not two.

p	·	v	d
p	d	l	q

```
\begin{tikzpicture}
  \matrix[
    matrix of nodes,
    column sep=-\pgflinewidth,
    row sep=-\pgflinewidth,
    execute at empty cell={\node{$\cdot$};},
    nodes={draw,
      text depth=.14cm,
      text height=.3cm,
      minimum width=.7cm}
    ]{
    p&&v&d\\
    p&d&l&q\\
    };
\end{tikzpicture}
```

2.5 Mysterious errors

When you put a Ti*k*Z matrix in a `\newcommand`, an error appears which leaves newbies a bit astonished:

```
! Package pgf Error: Single ampersand used
with wrong catcode.
```

You may also get something like:

```
! Extra alignment tab has been changed to \cr.
```

when the text in a matrix node is a `tabular`.

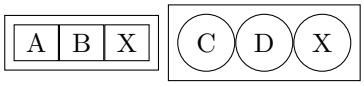
These problems happen because Ti*k*Z does not actually use `&` to separate cells, even if it seems so, but the command `\pgfmatrixnextcell`.

I will not explain (it is too difficult for me!) how Ti*k*Z substitutes the macro `\pgfmatrixnextcell` with `&`, but sometimes it fails. If you are interested, see Section 20.5 *Considerations Concerning Active Characters* of [2] for further details.

Well, don't panic, quack! There is a simple solution for it, just use the option `ampersand replacement=`⟨*macro name*⟩.

For instance, you can set `ampersand replacement=\&` and then use `\&` instead of `&` as column separator for the Ti*k*Z matrix, or `\pgfmatrixnextcell` directly.

An example of a `\matrix` in a macro follows. Note that if you put the `ampersand replacement` as an option of the `tikzpicture`, it is valid for any matrix in the picture.[2]



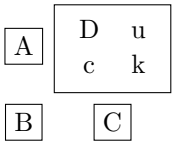

```
% The % signs present in the following macro
% definition are mandatory to avoid
% spurious spaces
\newcommand{\mymatrix}[3][]{%
  \begin{tikzpicture}[
    ampersand replacement=\&,
    nodes={draw},
    baseline=0pt
    ]
    \matrix[matrix of nodes, nodes={#1},
        text width=1em, text centered,
        ] {
        {#2} \& {#3} \& X\\
        };
  \end{tikzpicture}%
```

[2] When you define a new command, as prof. Enrico Gregorio always says, do not forget putting the `%`'s where needed, to avoid spurious spaces!

I take the occasion to thank him for suggesting the previous example and for his proofreading. Of course, all remaining errors are mine.

```
}
...
\begin{document}
  \mymatrix{A}{B} \mymatrix[circle]{C}{D}
\end{document}
```

And here is an example of a `tabular` in a cell:




```
\begin{tikzpicture}
  \matrix [
    nodes=draw,
    matrix of nodes,
    ampersand replacement=\&,
    row sep=4pt,
    column sep=4pt,
    ]{
    A \& \begin{tabular}{cc}
            D & u \\
            c & k
          \end{tabular}\\
    B \pgfmatrixnextcell C \\
    };
\end{tikzpicture}
```

2.6 Matrices of math nodes

A peculiar case of matrices of nodes are matrices of *math* nodes; we could compare them to `array` environments.

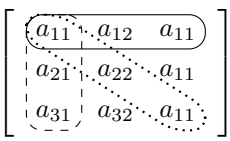
Indeed, the option `matrix of math nodes` allows you to avoid putting `$...$` or `\(...\)` in every cell. As you know, LaTeX hates repetitive boring actions, quack!

If you would like to have some delimiters around your matrix (or, in general, around any node), you can use the
`left/right/above/below delimiter =` ⟨*delimiter*⟩ option.

The ⟨*delimiter*⟩ can be anything acceptable to the ordinary `\left` command.

Of course, we do not need a Ti*k*Z matrix for a simple math matrix; there are already many math environments to create it without Ti*k*Z.

On the other hand, a Ti*k*Z matrix is useful if you would like to add some graphical accessories to it. In the following example, the first row and column are highlighted using `fit`, and the main diagonal with a `\draw`:




```
\documentclass{standalone}
\usepackage{tikz}
\usetikzlibrary{matrix, fit}
\begin{document}
\begin{tikzpicture}[
  every matrix/.style={
    matrix of math nodes,
    column sep =1mm,
    row sep =1mm,
    left delimiter={[},
    right delimiter={]},
    }
  ]
  \matrix (mymat) {
     a_{11} & a_{12} & a_{11}\\
     a_{21} & a_{22} & a_{11}\\
     a_{31} & a_{32} & a_{11}\\
     };
  \node[fit=(mymat-1-1)(mymat-1-3), draw,
    inner sep=0pt, rounded corners=6pt] {};
  \node[fit=(mymat-1-1)(mymat-3-1), draw,
    inner sep=0pt, rounded corners=6pt,
    dashed] {};
  \draw[rounded corners=5pt, dotted, thick]
    ([yshift=-2pt]mymat-1-1.west) |-
    ([xshift=2pt]mymat-1-1.north) --
    ([yshift=2pt]mymat-3-3.east) |-
    ([xshift=-2pt]mymat-3-3.south) --
    cycle;
\end{tikzpicture}
\end{document}
```

3 Conclusion

I hope you enjoyed my explanation, and if you are searching for The One, remember:

A duck can fly!
(Humans cannot)

References

[1] C. Maggi. The DuckBoat — news from TeX.SE: The Morse code of TikZ. *TUGboat* 39(1):21–26, 2018. https://tug.org/TUGboat/tb39-1/tb121duck-tikz.pdf

[2] T. Tantau. The Ti*k*Z and PGF packages. http://mirrors.ctan.org/graphics/pgf/base/doc/pgfmanual.pdf. Package page: https://ctan.org/pkg/pgf.

◇ Herr Professor Paulinho van Duck
Quack University Campus
Sempione Park Pond
Milano, Italy
paulinho dot vanduck (at) gmail dot com