The tikzquads Package
An Extension to CircuiTi\textit{k}Z
Version 1.0
Alceu Frigeri\textsuperscript{*}
January 2024

Abstract
This package defines a few extra shapes (single / dual port boxes) designed to be used together with the CircuiTi\textit{k}Z package.

Contents
1 Introduction ........................................... 2
  1.1 CircuiTi\textit{k}Z ................................... 2

2 Auxiliary Shapes and Basic Keys .......................... 2
  2.1 Auxiliary shapes ................................... 2
  2.2 General Keys ....................................... 3

3 \textit{Z}, \textit{Y}, \textit{G}, \textit{H} Quadripoles ....................... 3
  3.1 The Base Quadripole Shape ........................... 3
    3.1.1 Base Keys .................................. 3
  3.2 Quad ............................................... 5
    3.2.1 Quad Keys .................................... 5
    3.2.2 Examples of \textit{fit to use} .................... 5
  3.3 Quad \textit{Z} ........................................ 6
    3.3.1 Quad \textit{Z} keys ............................... 6
  3.4 Quad \textit{Y} ........................................ 7
    3.4.1 Quad \textit{Y} keys ............................... 7
  3.5 Quad \textit{G} ........................................ 7
    3.5.1 Quad \textit{G} keys ............................... 8
  3.6 Quad \textit{H} ........................................ 8
    3.6.1 Quad \textit{H} keys ............................... 9

4 Thevenin, Norton single port boxes ...................... 9
  4.1 The Base Black Box Shape ........................... 9
    4.1.1 Base Keys .................................. 9
  4.2 Black Box ......................................... 10
    4.2.1 Black Box keys ................................ 11
    4.2.2 Examples of \textit{fit to use} .................... 11
  4.3 Thevenin .......................................... 11
    4.3.1 Thevenin keys ................................ 12
  4.4 Norton .............................................. 12
    4.4.1 Norton keys .................................. 12

5 Pseudo-Graph Shape .................................. 12
  5.1 Pseudo-Graph Keys ................................. 13

6 Examples of use ..................................... 13

\textsuperscript{*}https://github.com/alceu-frigeri/tikzquads
1 Introduction

In standard text books, both for Circuits Theory and Electronics, quite frequently, in the process of modelling sub-circuits, one ends representing them either:

- as a single port *black box*, or
- as a dual port *black box*

This package defines a few, parameterized shapes for each case:

- for single port *black boxes*:
  - Black Box
  - Thevenin
  - Norton
- for dual port *black boxes*:
  - Quad
  - Quad Z
  - Quad Y
  - Quad G
  - Quad H

Lastly, this package also defines a *Pseudo-Graph load line* shape, for those moments where a true graph, *pgfplots*, isn’t needed.

1.1 CircuiTikZ

Unfortunately, some implementation details of these shapes don’t follow the code structure adopted by *CircuiTikZ*, and some significant part of this package’s code would have to be re-written if it were to be integrated directly in *CircuiTikZ*, and that’s the main reason this is, for the time being, a separate package. After all, even though this doesn’t follows *CircuiTikZ* code scheme, it does work nicely with it, as is.

2 Auxiliary Shapes and Basic Keys

Those shapes are not intended for end users.

2.1 Auxiliary shapes

A set of auxiliary shapes are defined, but not meant to be used otherwise, though their anchors might be relevant:

Note: The point being that, regardless of the sub-shape orientation, the intuitive geographical coordinates applies.
2.2 General Keys

These are the keys to fine tuning a shape:
- \texttt{outer sep} \hfill Text outer separation, initial value: 1.5pt
- \texttt{inner sep} \hfill Text inner separation, initial value: 1pt
- \texttt{thickness} \hfill Components thickness (relative to the drawing thickness), initial value: 2pt
- \texttt{tip len} \hfill tip(len) (current source), initial value: 4pt
- \texttt{tip type} \hfill possible values: \texttt{triangle} and \texttt{bezier}, initial value: \texttt{triangle}
- \texttt{minussign len} \hfill Minus sign len (voltage source), initial value: $\pgf@circ@Rlen/14$
- \texttt{plussign len} \hfill Plus sign len (voltage source), initial value: \texttt{1.1*pgf@circ@Rlen/14}
- \texttt{source radius} \hfill The base radius, initial value: \texttt{0.3*pgf@circ@Rlen}
- \texttt{round sources} \hfill Sources will be round ones
- \texttt{control sources} \hfill Sources will be control/diamond ones
- \texttt{generic, european} \hfill Impedances will be generic rectangles
- \texttt{zigzag, american} \hfill Impedances will be draw as zigzags

3 Z, Y, G, H Quadripoles

A set of configurable Quadripoles is defined, whereas quadripoles parameters (for instance $Z_{11}$, $Z_{12}$, $Z_{21}$ and $Z_{22}$) are (key-value) parameters.

3.1 The Base Quadripole Shape

The base shape just draws a base box and sets some connection anchors: 1+, 1−, \texttt{inner} 1+, \texttt{inner} 1−, 2+, 2−, \texttt{inner} 2+ and \texttt{inner} 2−, besides the geographic ones:

And also a set of (meant for) \texttt{text} anchors:

3.1.1 Base Keys

These applies to all Quad shapes:
- \texttt{base width} \hfill The 'box' width
- \texttt{half base width} \hfill Ditto, half width. Initial value: $2*\pgf@circ@Rlen$.
- \texttt{base height} \hfill The distance between 1+ and 1−. The 'box' full height is equal to $2*(\texttt{half base height} + \texttt{height ext} + \texttt{height ext+})$.
- \texttt{half base height} \hfill Ditto, half height. Initial value: $\pgf@circ@Rlen/7$
- \texttt{height ext} \hfill Initial value: $\pgf@circ@Rlen/7$
- \texttt{height ext+} \hfill Initial value: 0
- \texttt{inner ext} \hfill distance between the 'box' and \texttt{inner}1+/1−/2+/2−. Initial value: $\pgf@circ@Rlen/7$
- \texttt{outer ext} \hfill distance between the 'box' and 1+/1−/2+/2−. Initial value: 5$\pgf@circ@Rlen/14$
- \texttt{inner marks} \hfill If set, the inner anchors will be marked.
- \texttt{outer marks} \hfill If set, the outer anchors will be marked.
- \texttt{invert} \hfill The shape will be inverted, more or less like 'x scale=-1'.
- \texttt{alt, opt} \hfill Case a Voltage source is zero, a series impedance will be draw vertically.
- \texttt{outer x fit to} \hfill For any Quad, this is the same as \texttt{outer x fit to*}. 

3
outer x fit to* \(\text{outer x fit} \equiv \{\text{CoordA}\} \{\text{CoordB}\}\). The width will be set so that \(1+\) and \(2+\) (or \(1-\) and \(2-\), depending on the used anchor) will fit \((\text{CoordA})\) and \((\text{CoordB})\). This might result in a shape rotation.

outer x fit! \(\text{outer x fit} \equiv \{\text{CoordA}\} \{\text{CoordB}\}\). The width will be set so that the distance between \(1+\) and \(2+\) (or \(1-\) and \(2-\), depending on the used anchor) will be the same as \((\text{CoordA})\) and \((\text{CoordB})\). This will never result in a shape rotation.

inner x fit to For any Quad, this is the same as inner x fit to*.
inner x fit to* \(\text{inner x fit} \equiv \{\text{CoordA}\} \{\text{CoordB}\}\). The width will be set so that \(\text{inner } 1+\) and \(\text{inner } 2+\) (or \(\text{inner } 1-\) and \(\text{inner } 2-\), depending on the used anchor) will fit \((\text{CoordA})\) and \((\text{CoordB})\). This might result in a shape rotation.

inner x fit! \(\text{inner x fit} \equiv \{\text{CoordA}\} \{\text{CoordB}\}\). The width will be set so that the distance between \(\text{inner } 1+\) and \(\text{inner } 2+\) (or \(\text{inner } 1-\) and \(\text{inner } 2-\), depending on the used anchor) will be the same as \((\text{CoordA})\) and \((\text{CoordB})\). This will never result in a shape rotation.

inner x fit to! For any Quad, this is the same as y fit to!.

y fit to For any Quad, this is the same as y fit to!.
y fit to* \(\text{y fit} \equiv \{\text{CoordA}\} \{\text{CoordB}\}\). The height will be set so that \(1+\) and \(1-\) will fit \((\text{CoordA})\) and \((\text{CoordB})\). This might result in a shape rotation.
y fit! \(\text{y fit} \equiv \{\text{CoordA}\} \{\text{CoordB}\}\). The height will be set so that the distance between \(1+\) and \(1-\) will be equal to the distance between \((\text{CoordA})\) and \((\text{CoordB})\). This will never result in a shape rotation.

y fit to! It will place a label at the top left anchor
label top left

y fit to! It will place a label at the top center anchor
label top center

y fit to! It will place a label at the top right anchor
label top right

y fit to! It will place a label at the inner top left anchor
label inner top left

y fit to! It will place a label at the inner top center anchor
label inner top center

y fit to! It will place a label at the inner top right anchor
label inner top right

y fit to! It will place a label at the bottom left anchor
label bottom left

y fit to! It will place a label at the bottom center anchor
label bottom center

y fit to! It will place a label at the bottom right anchor
label bottom right

y fit to! It will place a label at the inner bottom left anchor
label inner bottom left

y fit to! It will place a label at the inner bottom center anchor
label inner bottom center

y fit to! It will place a label at the inner bottom right anchor
label inner bottom right

A small example of the fit to keys:

\begin{tikzpicture}
\draw (0,0) \pincoord{A}{blue,225} ++(4,0) \pincoord{B}{blue,-45} ++(2,2) \pincoord{C} ;
\draw (A) node[Quad,anchor=1+,outer x fit to={A}{B}](Qa){\footnotesize $Q_a$};
\draw (B) node[Quad,anchor=1+,outer x fit to={B}{C},I1=$I_a$,V2=$V_b$](Qb){\footnotesize $Q_b$};
\draw (Qb.2-) -- ++(2,0) \pincoord{D} ++(1,-2) \pincoord{E};
\draw (D) node[Black Box,anchor=1+,y fit to={D}{E}](Ba){\footnotesize $Ba$};
\draw (Qa.1-) ++(0,-1);
\end{tikzpicture}
3.2 Quad

This is just the base shape, to be used in cases whereas one just want to emphasises part of a circuit (using, for instance, the inner x fit to key, or just mark a two port black box.

Note: There is also a ToQuad to be used in a to[] path, in which case the key outer x fit to style will be triggered with the starting and ending points of the to[] path.

3.2.1 Quad Keys

**name**

⟨node-name⟩, when using a to[] path.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>I₁</td>
<td>Initial value: $I_{18}$</td>
</tr>
<tr>
<td>I₂</td>
<td>Initial value: $I_{28}$</td>
</tr>
<tr>
<td>V₁</td>
<td>Initial value: $V_{18}$</td>
</tr>
<tr>
<td>V₂</td>
<td>Initial value: $V_{28}$</td>
</tr>
</tbody>
</table>

3.2.2 Examples of fit to use

Squeezing a Quadripole between two parts of a circuit (nodes C and D):

```
\begin{center}
\begin{tikzpicture}
\draw (0,0) \pincoord(ref) to[R=R1] ++(0,2) \pincoord(A) to[R=R2] ++(0,2) \pincoord(B) -- ++(2,0) \pincoord(C,red,225) (C |- ref) \pincoord(C1,blue,135) -- (ref);
\draw (C) ++(7,0) \pincoord(D,red) -- ++(0.5,0) to[R=R3] ++(0,-3) -- ++(2,0) to[R=R4] ++(0,3) -- ++(0.5,0) \pincoord(E);
\draw (C) node[Quad,anchor=1+,y fit to=(C){C1},outer x fit to=(C){D}]{};
\end{tikzpicture}
\end{center}
```
Fitting some circuit inside the Quadripole (nodes C and E):

\begin{tikzpicture}
\draw (0,0) to[R=R1] ++(0,2) to[R=R2] ++(0,2) -- ++(2,0) (C |- ref) -- (ref);
\draw (C) ++(7,0) -- ++(0.5,0) to[R=R3] ++(0,-3) -- ++(2,0) to[R=R4] ++(0,3) -- ++(0.5,0) (E,red);
\draw (C) node[Quad,anchor=inner 1+,y fit to={C}{C1},inner x fit to={C}{E}]{};
\end{tikzpicture}

3.3 Quad Z

This shape, besides the base anchors (see 3) it has 4 internal nodes: \langle node \rangle-Z11, \langle node \rangle-Z12, \langle node \rangle-Z21 and \langle node \rangle-Z22 and each of those sub-nodes has geographic anchors as defined at 2.1.

**Note:** There is also a \texttt{ToQuad Z} to be used in a \texttt{to[ ]} path, in which case the key \texttt{outer x fit} to style will be triggered with the starting and ending points of the \texttt{to[]} path.

3.3.1 Quad Z keys

- \texttt{name} \langle node-name \rangle, when using a \texttt{to[ ]} path.
- \texttt{I1} Initial value: $I_1$
- \texttt{I2} Initial value: $I_2$
- \texttt{V1} Initial value: $V_1$
- \texttt{V2} Initial value: $V_2$
- \texttt{Z11} Initial value: $Z_{11}$
- \texttt{Z12} Initial value: $Z_{12}$
- \texttt{Z21} Initial value: $Z_{21}$
- \texttt{Z22} Initial value: $Z_{22}$
- \texttt{Z11 label pos} changes the label position. Defaults to: \{south west\}{top left}\``
- \texttt{Z12 label pos} changes the label position. Defaults to: \{south east\}{top left}\``
- \texttt{Z21 label pos} changes the label position. Defaults to: \{north west\}{bottom right}\``
- \texttt{Z22 label pos} changes the label position. Defaults to: \{south east\}{top right}\``

**Note:** The label pos keys expects two anchor names (\texttt{... label pos=\{(anchor A)\}(anchor B)}). The first anchors refers the sub-shape node and the second anchor is the text one.
3.4 Quad Y

This shape, besides the base anchors (see 3) it has 4 internal nodes: \(<\text{node}>-Y_{11}, <\text{node}>-Y_{12}, <\text{node}>-Y_{21} \text{ and } <\text{node}>-Y_{22}\) and each of those sub-nodes has geographic anchors as defined at 2.1.

**Note:** There is also a \(\text{ToQuad Y}\) to be used in a \(\text{to[ ]}\) path, in which case the key \(\text{outer x fit to}\) style will be triggered with the starting and ending points of the \(\text{to[ ]}\) path.

3.4.1 Quad Y keys

- **name**\(\langle\text{node-name}\rangle\), when using a \(\text{to[ ]}\) path.
- **I1**\(I_1\) Initial value: $I_1$
- **I2**\(I_2\) Initial value: $I_2$
- **V1**\(V_1\) Initial value: $V_1$
- **V2**\(V_2\) Initial value: $V_2$
- **Y11**\(Y_{11}\) Initial value: $Y_{11}$
- **Y12**\(Y_{12}\) Initial value: $Y_{12}$
- **Y21**\(Y_{21}\) Initial value: $Y_{21}$
- **Y22**\(Y_{22}\) Initial value: $Y_{22}$
- **Y11 label pos** changes the label position. Defaults to: \(\text{south west}\{\text{top left}\}\)
- **Y12 label pos** changes the label position. Defaults to: \(\text{south east}\{\text{top left}\}\)
- **Y21 label pos** changes the label position. Defaults to: \(\text{north west}\{\text{bottom right}\}\)
- **Y22 label pos** changes the label position. Defaults to: \(\text{north west}\{\text{bottom right}\}\)

**Note:** The label pos keys expects two anchor names (... label pos= {\langle anchor A\rangle} {\langle anchor B\rangle}). The first anchors refers the sub-shape node and the second anchor is the text one.

3.5 Quad G

This shape, besides the base anchors (see 3) it has 4 internal nodes: \(<\text{node}>-G_{11}, <\text{node}>-G_{12}, <\text{node}>-G_{21} \text{ and } <\text{node}>-G_{22}\) and each of those sub-nodes has geographic anchors as defined at 2.1.

**Note:** There is also a \(\text{ToQuad G}\) to be used in a \(\text{to[ ]}\) path, in which case the key \(\text{outer x fit to}\) style will be triggered with the starting and ending points of the \(\text{to[ ]}\) path.
3.5.1 Quad G keys

$name$ (node-name), when using a to[] path.

$I_1$ Initial value:$I_1$\textunderscore$1$
$I_2$ Initial value:$I_2$\textunderscore$2$
$V_1$ Initial value:$V_1$\textunderscore$1$
$V_2$ Initial value:$V_2$\textunderscore$2$
$G_{11}$ Initial value:$G_{11}$\textunderscore$11$
$G_{12}$ Initial value:$G_{12}$\textunderscore$12$
$G_{21}$ Initial value:$G_{21}$\textunderscore$21$
$G_{22}$ Initial value:$G_{22}$\textunderscore$22$

$G_{11}$ label pos changes the label position. Defaults to: \{south west\}\{top left\}
$G_{12}$ label pos changes the label position. Defaults to: \{south east\}\{top left\}
$G_{21}$ label pos changes the label position. Defaults to: \{north west\}\{bottom right\}
$G_{22}$ label pos changes the label position. Defaults to: \{south east\}\{top right\}

Note: The label pos keys expects two anchor names (... label pos=\{⟨anchor A⟩\} \{⟨anchor B⟩\}). The first anchors refers the sub-shape node and the second anchor is the text one.

3.6 Quad H

This shape, besides the base anchors (see 3) it has 4 internal nodes: \<node>-H11, \<node>-H12, \<node>-H21 and \<node>-H22 and each of those sub-nodes has geographic anchors as defined at 2.1.

Note: There is also a ToQuad H to be used in a to[J] path, in which case the key outer x fit to style will be triggered with the starting and ending points of the to[J] path.
3.6.1 Quad H keys

name \( \langle \text{node-name} \rangle \), when using a \text{to[\]} \path.

\begin{align*}
I_1 & \quad \text{Initial value: } \$I_1\$
I_2 & \quad \text{Initial value: } \$I_2\$
V_1 & \quad \text{Initial value: } \$V_1\$
V_2 & \quad \text{Initial value: } \$V_2\$
H_{11} & \quad \text{Initial value: } \$H_{11}\$
H_{12} & \quad \text{Initial value: } \$H_{12}\$
H_{21} & \quad \text{Initial value: } \$H_{21}\$
H_{22} & \quad \text{Initial value: } \$H_{22}\$
\end{align*}

\begin{align*}
H_{11} \text{ label pos} & \quad \text{changes the label position. Defaults to: } \{\text{south west}\}{\text{top left}}
H_{12} \text{ label pos} & \quad \text{changes the label position. Defaults to: } \{\text{south east}\}{\text{top left}}
H_{21} \text{ label pos} & \quad \text{changes the label position. Defaults to: } \{\text{north west}\}{\text{bottom right}}
H_{22} \text{ label pos} & \quad \text{changes the label position. Defaults to: } \{\text{north west}\}{\text{bottom right}}
\end{align*}

Note: The label pos keys expects two anchor names \( \langle \text{anchor A} \rangle \) \( \langle \text{anchor B} \rangle \). The first anchors refers the sub-shape node and the second anchor is the text one.

4 Thevenin, Norton single port boxes

4.1 The Base Black Box Shape

The base shape just draws a base box and sets some connection anchors: 1+, 1−, \text{inner} 1+, \text{inner} 1−, besides the geographic and text ones:

4.1.1 Base Keys

These applies to all \textit{Black Box} shapes:

\begin{align*}
\text{base width} & \quad \text{The 'box' width}
\text{half base width} & \quad \text{Ditto, half width. Initial value: } 2\texttt{pgf@circ@Rlen}.
\text{base height} & \quad \text{The distance between 1+ and 1−. The 'box' full height is equal to } 2\ast(\text{half base height} + \text{height ext} + \text{height ext+}).
\text{half base height} & \quad \text{Ditto, half height. Initial value: } \texttt{pgf@circ@Rlen}/7
\text{height ext} & \quad \text{Initial value: } 2\cdot\texttt{pgf@circ@Rlen}/7
\text{height ext+} & \quad \text{Initial value: } 0
\text{inner ext} & \quad \text{distance between the 'box' and \text{inner}1+/1−/2+/2−. initial value: } \texttt{pgf@circ@Rlen}/7
\text{outer ext} & \quad \text{distance between the 'box' and 1+/1−/2+/2−. initial value: } 5\cdot\texttt{pgf@circ@Rlen}/14
\text{inner marks} & \quad \text{If set, the inner anchors will be marked.}
\text{outer marks} & \quad \text{If set, the outer anchors will be marked.}
\end{align*}
The shape will be inverted, more or less like 'x scale=-1'.

Case a Voltage source is zero, a series impedance will be draw vertically.

For any Black Box, this is the same as outer x fit to.

outer x fit *={⟨CoordA⟩} {⟨CoordB⟩}. The width will be set so that (1+) and (2+) (or (1-) and (2-), depending on the used anchor) will fit (CoordA) and (CoordB). This might result in a shape rotation.

draw V (series) will be the same as (CoordA) and (CoordB). This will never result in a shape rotation.

inner x fit to
inner x fit *= {⟨CoordA⟩} {⟨CoordB⟩}. The width will be set so that the distance between (1+) and (2+) (or (1-) and (2-), depending on the used anchor) will fit (CoordA) and (CoordB). This might result in a shape rotation.

inner x fit *= {⟨CoordA⟩} {⟨CoordB⟩}. The width will be set so that the distance between (inner 1+) and (inner 2+) (or (inner 1-) and (inner 2-), depending on the used anchor) will be the same as (CoordA) and (CoordB). This will never result in a shape rotation.

inner x fit *= {⟨CoordA⟩} {⟨CoordB⟩}. The width will be set so that the distance between (inner 1+) and (inner 2+) (or (inner 1-) and (inner 2-), depending on the used anchor) will be the same as (CoordA) and (CoordB). This will never result in a shape rotation.

inner x fit *= {⟨CoordA⟩} {⟨CoordB⟩}. The width will be set so that the distance between (inner 1+) and (inner 2+) (or (inner 1-) and (inner 2-), depending on the used anchor) will be the same as (CoordA) and (CoordB). This will never result in a shape rotation.

y fit to
y fit *= {⟨CoordA⟩} {⟨CoordB⟩}. The height will be set so that 1+ and 1- will fit CoordA and CoordB. This might result in a shape rotation

y fit *= {⟨CoordA⟩} {⟨CoordB⟩}. The height will be set so that the distance between (1+) and (1-) will be equal to the distance between (CoordA) and (CoordB). This will never result in a shape rotation.

inner bottom center
inner bottom center
inner bottom left
inner top center
inner top left
inner top right

label top left
It will place a label at the top left anchor
label top center
It will place a label at the top center anchor
label top right
It will place a label at the top right anchor
label inner top left
It will place a label at the inner top left anchor
label inner top center
It will place a label at the inner top center anchor
label inner top right
It will place a label at the inner top right anchor
label bottom left
It will place a label at the bottom left anchor
label bottom center
It will place a label at the bottom center anchor
label bottom right
It will place a label at the bottom right anchor
label inner bottom left
It will place a label at the inner bottom left anchor
label inner bottom center
It will place a label at the inner bottom center anchor
label inner bottom right
It will place a label at the inner bottom right anchor

4.2 Black Box

This is just the base shape, to be used in cases whereas one just want to emphasises part of a circuit (using, for instance, the inner x fit to key, or just mark a single port black box.

Note: There is also a ToBlack Box to be used in a to[ ] path, in which case the key y fit to style will be triggered with the starting and ending points of the to[ ] path.
4.2.1 Black Box keys

name  (node-name), when using a to[ ] path.

$I_1$  Initial value:$I_1$

$V_1$  Initial value:$V_1$

4.2.2 Examples of fit to use

Squeezing a Black Box between two parts of a circuit (nodes C and D):

```
\begin{tikzpicture}
\draw (0,0) \pincoord(ref) to[R=R1] ++(0,2) \pincoord(A) to[R=R2] ++(0,2) \pincoord(B) -- ++(2,0) \pincoord(C,red) (C |- ref) \pincoord(C1) -- (ref);
\draw (C) ++(7,0) \pincoord(O,red) -- ++(0.5,0) to[R=R3] ++(0,-3) -- ++(2,0) to[R=R4] ++(0,3) -- ++(0.5,0) \pincoord(E);
\draw (C) node[Black Box,anchor=1+,y fit to={C}{C1},outer x fit to={C}{D}]{};
\end{tikzpicture}
```

Fitting some circuit inside the Black Box (nodes C and E):

```
\begin{tikzpicture}
\draw (0,0) \pincoord(ref) to[R=R1] ++(0,2) \pincoord(A) to[R=R2] ++(0,2) \pincoord(B) -- ++(2,0) \pincoord(C,red) (C |- ref) \pincoord(C1) -- (ref);
\draw (C) ++(7,0) \pincoord(O,red) -- ++(0.5,0) to[R=R3] ++(0,-3) -- ++(2,0) to[R=R4] ++(0,3) -- ++(0.5,0) \pincoord(E);
\draw (C) node[Black Box,anchor=inner 1+,y fit to={C}{C1},inner x fit to={C}{E}]{};
\end{tikzpicture}
```

4.3 Thevenin

% Node use
\node[Thevenin]{}

% To path use
(A) to[ToThevenin] (B)

This is the classical Thevenin circuit. Besides the base anchors (see 4.1) it has 2 internal nodes: <node>-Zth and <node>-Vth and each of those sub-nodes has geographic anchors as defined at 2.1.

Note: There is also a ToThevenin to be used in a to[ ] path, in which case the key y fit to style will be triggered with the starting and ending points of the to[ ] path.
4.3.1 Thevenin keys

\begin{verbatim}
name (node-name), when using a to[] path.
I1 Initial value:$I_1$
V1 Initial value:$V_1$
Zth Initial value:$Z_{th}$
V_th Initial value:$V_{th}$

Zth label pos changes the label position. Defaults to: {south west}{top left}
V_th label pos changes the label position. Defaults to: {south east}{top left}
\end{verbatim}

Note: The label pos keys expects two anchor names (... label pos={⟨anchor A⟩} {(anchor B)}). The first anchors refers the sub-shape node and the second anchor is the text one.

4.4 Norton

This is the classical Norton circuit. Besides the base anchors (see 4.1) it has 2 internal nodes: <node>-Yn and <node>-In and each of those sub-nodes has geographic anchors as defined at 2.1.

Note: There is also a ToNorton to be used in a to[] path, in which case the key y fit to style will be triggered with the starting and ending points of the to[] path.

4.4.1 Norton keys

\begin{verbatim}
name (node-name), when using a to[] path.
I1 Initial value:$I_1$
V1 Initial value:$V_1$
Yn Initial value:$Y_{n}$
In Initial value:$I_{n}$

Yn label pos changes the label position. Defaults to: {south west}{top left}
In label pos changes the label position. Defaults to: {south east}{top left}
\end{verbatim}

Note: The label pos keys expects two anchor names (... label pos={⟨anchor A⟩} {(anchor B)}). The first anchors refers the sub-shape node and the second anchor is the text one.

5 Pseudo-Graph Shape

Sometimes when representing a single port sub-circuit, one might use a X-Y graph, for which gnuplot and pgfplots are excellent choices, but a bit overkill if all you want is a crude representation of a linear load line.

This shape is just that, a X-Y graph mockup, that nicely fits inside a black box, and nothing else.
5.1 Pseudo-Graph Keys

These are the keys to fine tuning a shape:

- **X axis name**
  - Initial value: $V$
- **X axis val**
  - X axis val at the crossing point. Initial value: $V_{th}$
- **Y axis name**
  - Initial value: $I$
- **Y axis val**
  - Y axis val at the crossing point. Initial value: $I_N$
- **first quadrant**
  - First quadrant mock up. (which is also the default).
- **second quadrant**
  - Second quadrant mock up.
- **third quadrant**
  - Third quadrant mock up.
- **fourth quadrant**
  - Fourth quadrant mock up.

**base width**

- The graph width
- Ditto, half width. Initial value: $0.5\pgf@circ@Rlen$.

**half base width**

- Ditto, half width. Initial value: $0.5\pgf@circ@Rlen$.

Note: Besides these, one can also use the keys presented at 2.2.

6 Examples of use

First of a simple case of combining a generic Quad with equations and a generic Black Box with a Pseudo-Graph:

```latex
\begin{tikzpicture}
\draw (0,0) \node[Quad,anchor=1+] (Q1) {}; 
\draw (Q1.2+) -- ++(1,0) \node[Black Box,anchor=1+,V1=\$V_a\$,I1=\$I_a\$] (B1) {$\ldots$} ; 
\draw (Q1.2-) -- (B1.1-) ;
\node at (Q1.center) {$\begin{matrix}
V_1 &=& 5j*V_2 + 2*I_2 \\
I_1 &=& 3*V_2 + 2j*I_2 
\end{matrix}$} ; 
\end{tikzpicture}
```
All default Quadripoles and Thevenin/Norton.

\begin{tikzpicture}
\draw (0,0) \coord(ref) node[Quad Z,anchor=1+](Qz1){}
\draw (Qz1.2+) -- ++(1.5,0) \coord(X) -- ++(1.5,0) node[Quad Y,anchor=1+](Qy1){}
\draw (Qy1.2+) -- ++(1,0) node[Thevenin,anchor=1+](th1){}
\draw (Qz1.1-) -- ++(0,-1.5) node[Quad H,anchor=1+](Qh1){}
\draw (Qh1.2+) -- ++(1.5,0) \coord(Y) -- ++(1.5,0) node[Quad G,anchor=1+](Qg1){}
\draw (Qg1.2+) -- ++(1,0) node[Norton,anchor=1+](nr1){}
\draw (Qz1.2-) -- (Qy1.1-) (Qy1.2-) -- (th1.1-)
\draw (Qh1.2-) -- (Qg1.1-) (Qg1.2-) -- (nr1.1-)
\draw (X) to[R=$R_x$] (X |- Qz1.2-)
\draw (Y) to[R=$R_y$] (Y |- Qh1.2-)
\end{tikzpicture}

The same demo but with all parameter 11 and 22 zeroed, and changing the “control sources”

\begin{tikzpicture}
\draw (0,0) \coord(ref) node[Quad Z,anchor=1+,Z11=0,Z22=0,I1=$I_a$,V1=$V_a$,I2=$I_b$,V2=$V_b$](Qz1){}
\draw (Qz1.2+) -- ++(1.5,0) \coord(X) -- ++(1.5,0) node[Quad Y,anchor=1+,Y11=0,Y22=0,I1=$I_d$,V1=$V_d$,I2=$I_c$,V2=$V_c$](Qy1){}
\draw (Qy1.2+) -- ++(1,0) node[Thevenin,anchor=1+,Zth=0,I1=$I_h$,V1=$V_h$](th1){}
\draw (Qz1.1-) -- ++(0,-1.5) node[Quad H,anchor=1+,H11=0,H22=0,I1=$I_e$,V1=$V_e$,I2=$I_e$,V2=$V_e$](Qh1){}
\draw (Qh1.2+) -- ++(1.5,0) \coord(Y) -- ++(1.5,0) node[Quad G,anchor=1+,G11=0,G22=0,I1=$I_g$,V1=$V_g$,I2=$I_f$,V2=$V_f$](Qg1){}
\draw (Qg1.2+) -- ++(1,0) node[Norton,anchor=1+,Yn=0,I1=$I_i$,V1=$V_i$](nr1){}
\draw (Qz1.2-) -- (Qy1.1-) (Qy1.2-) -- (th1.1-)
\draw (Qh1.2-) -- (Qg1.1-) (Qg1.2-) -- (nr1.1-)
\draw (X) to[R=$R_x$] (X |- Qz1.2-)
\draw (Y) to[R=$R_y$] (Y |- Qh1.2-)
\end{tikzpicture}
Now with the 12 and 21 parameters zeroed, normal form:

\begin{tikzpicture}
\draw [->] (0,0) \ncoord(ref) node[Quad Z,anchor=1+,Z12=0,Z21=0,I1=\(I_a\),V1=\(V_a\),I2=\(I_b\),V2=\(V_b\)](Qz1){}
(Qz1.2+) -- ++(1.5,0) \ncoord(X) -- ++(1.5,0) node[Quad Y,anchor=1+,Y12=0,Y21=0,I1=\(I_d\),V1=\(V_d\),I2=\(I_c\),V2=\(V_c\)](Qy1){}
(Qy1.2+) -- ++(1,0) node[Thevenin,anchor=1+,Vth=0,I1=\(I_h\),V1=\(V_h\)](th1){}
(Qz1.1-) -- ++(0,-1.5) node[Quad H,anchor=1+,H12=0,H21=0,I1=\(I_e\),V1=\(V_e\),I2=\(I_e\),V2=\(V_e\)](Qh1){}
(Qh1.2+) -- ++(1.5,0) \ncoord(Y) -- ++(1.5,0) node[Quad G,anchor=1+,G12=0,G21=0,I1=\(I_f\),V1=\(V_g\),I2=\(I_f\),V2=\(V_f\)](Qg1){}
(Qg1.2+) -- ++(1,0) node[Norton,anchor=1+,In=0,I1=\(I_i\),V1=\(V_i\)](nr1){}
\draw (X) to[R=\(R_x\)] (X |- Qz1.2-)
\draw (Y) to[R=\(R_y\)] (Y |- Qh1.2-)
\end{tikzpicture}

Same as last one, but with an alternate form:

\begin{tikzpicture}
\draw (0,0) \ncoord(ref) node[Quad Z,anchor=1+,Z12=0,Z21=0,I1=\(I_a\),V1=\(V_a\),I2=\(I_b\),V2=\(V_b\)](Qz1){}
(Qz1.2+) -- ++(1.5,0) \ncoord(X) -- ++(1.5,0) node[Quad Y,anchor=1+,Y12=0,Y21=0,I1=\(I_d\),V1=\(V_d\),I2=\(I_c\),V2=\(V_c\)](Qy1){}
(Qy1.2+) -- ++(1,0) node[Thevenin,anchor=1+,Vth=0,I1=\(I_h\),V1=\(V_h\)](th1){}
(Qz1.1-) -- ++(0,-1.5) node[Quad H,anchor=1+,H12=0,H21=0,I1=\(I_e\),V1=\(V_e\),I2=\(I_e\),V2=\(V_e\)](Qh1){}
(Qh1.2+) -- ++(1.5,0) \ncoord(Y) -- ++(1.5,0) node[Quad G,anchor=1+,G12=0,G21=0,I1=\(I_f\),V1=\(V_g\),I2=\(I_f\),V2=\(V_f\)](Qg1){}
(Qg1.2+) -- ++(1,0) node[Norton,anchor=1+,In=0,I1=\(I_i\),V1=\(V_i\)](nr1){}
\draw (X) to[R=\(R_x\)] (X |- Qz1.2-)
\draw (Y) to[R=\(R_y\)] (Y |- Qh1.2-)
\end{tikzpicture}
Setting all parameters, some impedances as zig-zag, others as generic, per quadripole: