The tikzquads Package
An Extension to CircuiTi*k*Z
Version 1.0

Alceu Frigeri*

January 2024

Abstract

This package defines a few extra shapes (single / dual port boxes) designed to be used together with the *CircuiTikZ* package.

Contents

1 Introduction **2**
1.1 CircuiTi*k*Z . . . 2

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

3 Z, Y, G, 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 *fit to* use . . . 5
3.3 Quad Z . . . 6
3.3.1 Quad Z keys . . . 6
3.4 Quad Y . . . 7
3.4.1 Quad Y keys . . . 7
3.5 Quad G . . . 7
3.5.1 Quad G keys . . . 8
3.6 Quad H . . . 8
3.6.1 Quad 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 *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**

---
*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 CircuiTi*k*Z

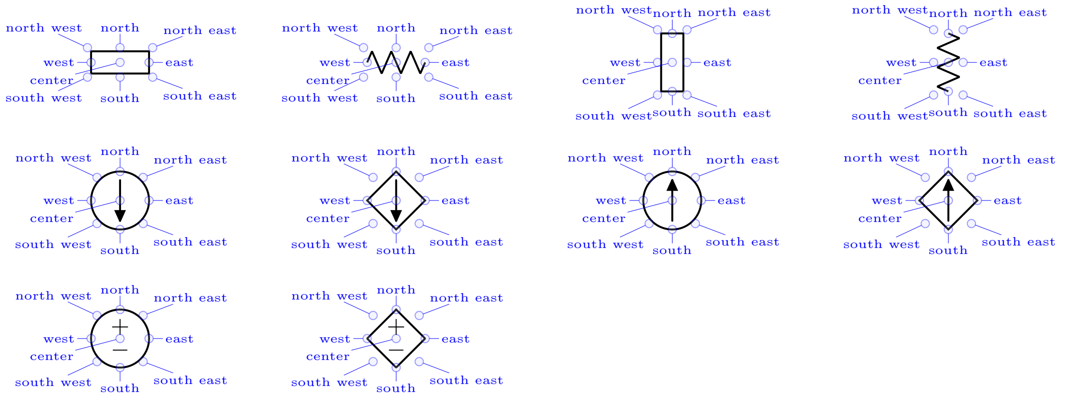
Unfortunately, some implementation details of these shapes don't follow the code structure adopted by `CircuiTikZ`, and some significant part of this package's code would have to be re-written if it were to be integrated directly in `CircuiTikZ`, and that's the main reason this is, for the time being, 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:

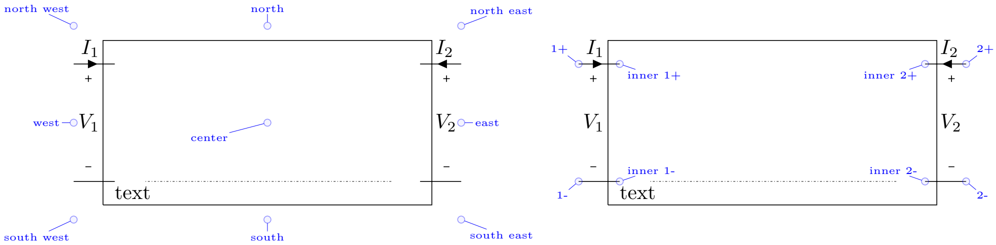
outer sep	Text outer separation, initial value: 1.5pt
inner sep	Text inner separation, initial value: 1pt
thickness	Components thickness (relative to the drawing thickness), initial value: 2
tip len	tip len (current source). initial value: 4pt
tip type	possible values: *triangle* and *bezier*. initial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

3 Z, Y, G, H Quadripoles

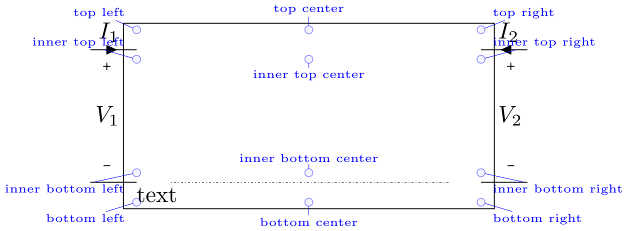
A set of configurable Quadripoles is defined, whereas quadripoles parameters (for instance Z_{11}, Z_{12}, Z_{21} and Z_{22}) are ⟨`key-value`⟩ parameters.

3.1 The Base Quadripole Shape

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



And also a set of (meant for) *text* anchors:



3.1.1 Base Keys

These applies to all Quad shapes:

base width	The 'box' width
half base width	Ditto, half width. Initial value:`2\pgf@circ@Rlen`.
base height	The distance between *1+* and *1-*. The 'box' full height is equal to 2*(*half base height* + *height ext* + *height ext+*).
half base height	Ditto, half height. Initial value:`\pgf@circ@Rlen/7`
height ext	Initial value:`2\pgf@circ@Rlen/7`
height ext+	Initial value:0
inner ext	distance between the 'box' and `inner1+/1-/2+/2-`. initial value: `\pgf@circ@Rlen/7`
outer ext	distance between the 'box' and `1+/1-/2+/2-`. initial value: `5\pgf@circ@Rlen/14`
inner marks	If set, the inner anchors will be marked.
outer marks	If set, the outer anchors will be marked.
invert	The shape will be inverted, more or less like 'x scale=-1'.
alt, opt	Case a Voltage source is zero, a series impedance will be draw vertically.
outer x fit to	For any Quad, this is the same as *outer x fit to**.

*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.

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

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

*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 to! — *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.

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

A small example of the *fit to* keys:

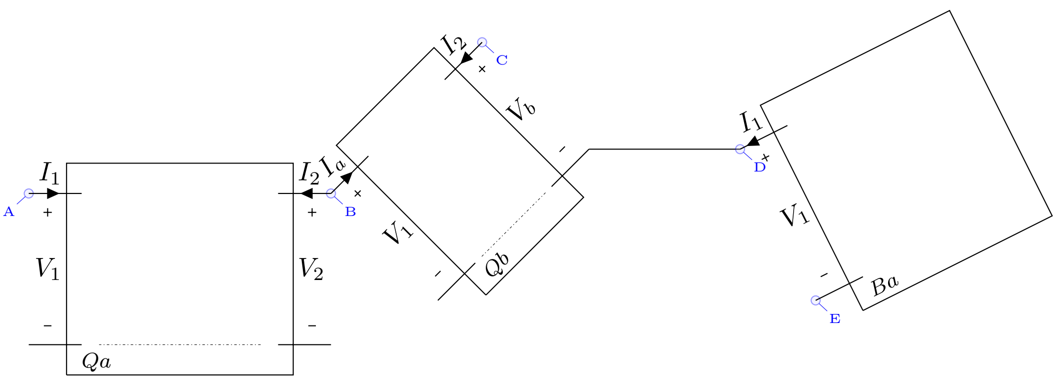
```
\begin{tikzpicture}
\draw (0,0)  \pincoord(A,blue,225) ++(4,0)  \pincoord(B,blue,-45) ++(2,2)  \pincoord(C) ;

\draw (A) node[Quad,anchor=1+,outer x fit to={A}{B}](Qa){\footnotesize$Qa$};
\draw (B) node[Quad,anchor=1+,outer x fit to={B}{C},I1=$I_a$,V2=$V_b$](Qb){\footnotesize$Qb$};

\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}
```



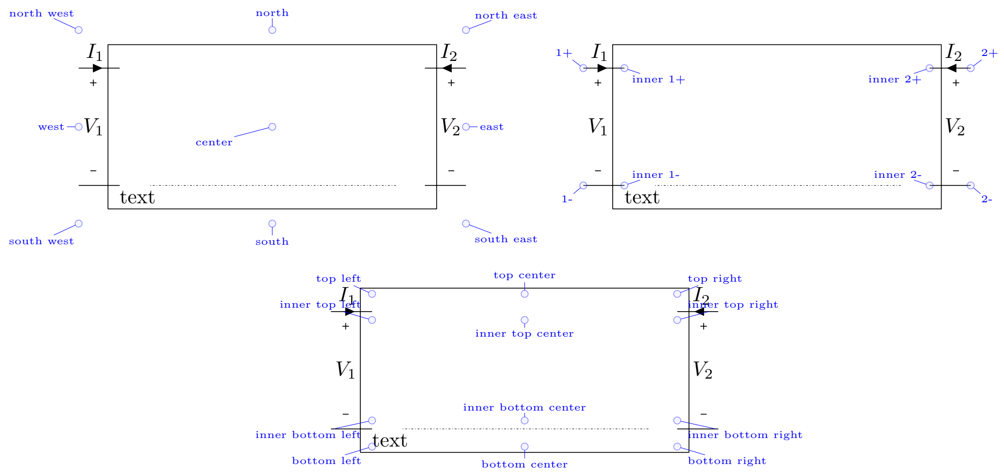
3.2 Quad

```
% Node use
node[Quad]{}

% To path use
(A) to[ToQuad]  (B)
```

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.



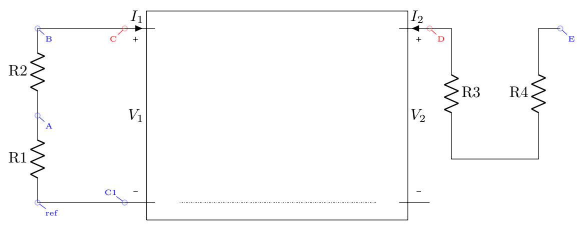
3.2.1 Quad Keys

name	⟨node-name⟩, when using a to[] path.
I1	Initial value:I_1
I2	Initial value:I_2
V1	Initial value:V_1
V2	Initial value:V_2

3.2.2 Examples of *fit to* use

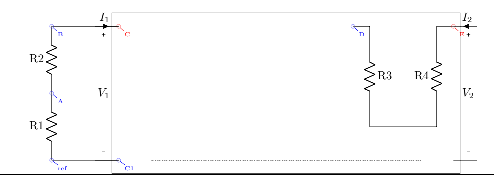
Squeezing a Quadripole between two parts of a circuit (nodes C and D):

```
\begin{center}
  \resizebox{0.5\textwidth}{!}{
    \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):

```
\resizebox{0.4\textwidth}{!}{
\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(D) -- ++(0.5,0) to[R=R3] ++(0,-3) -- ++(2,0) to[R=R4] ++(0,3) -- ++(0.5,0)
      \pincoord(E,red);
  \draw (C) node[Quad,anchor=inner 1+,y fit to={C}{C1},inner x fit to={C}{E}]{};
\end{tikzpicture}}
```



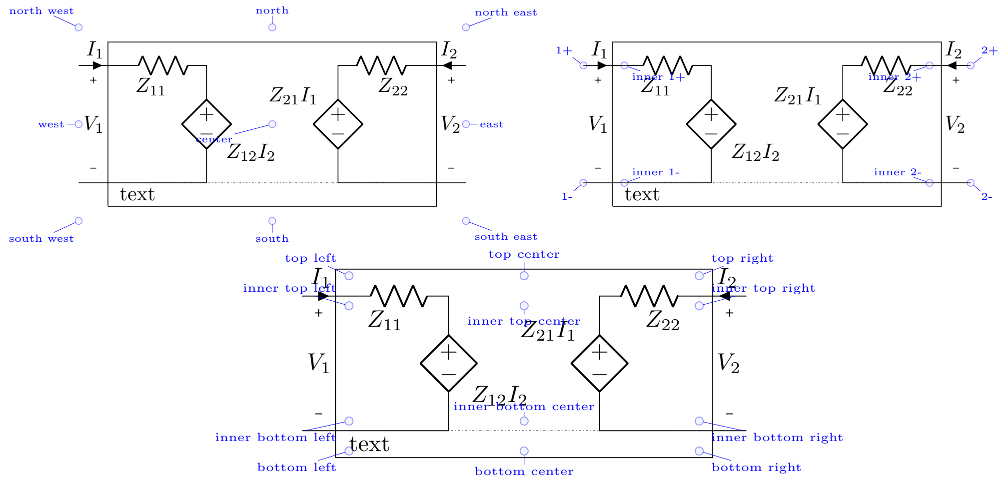
3.3 Quad Z

```
% Node use
node[Quad Z]{}

% To path use
(A) to[ToQuad Z]  (B)
```

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

Note: There is also a *ToQuad Z* 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.



3.3.1 Quad Z keys

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

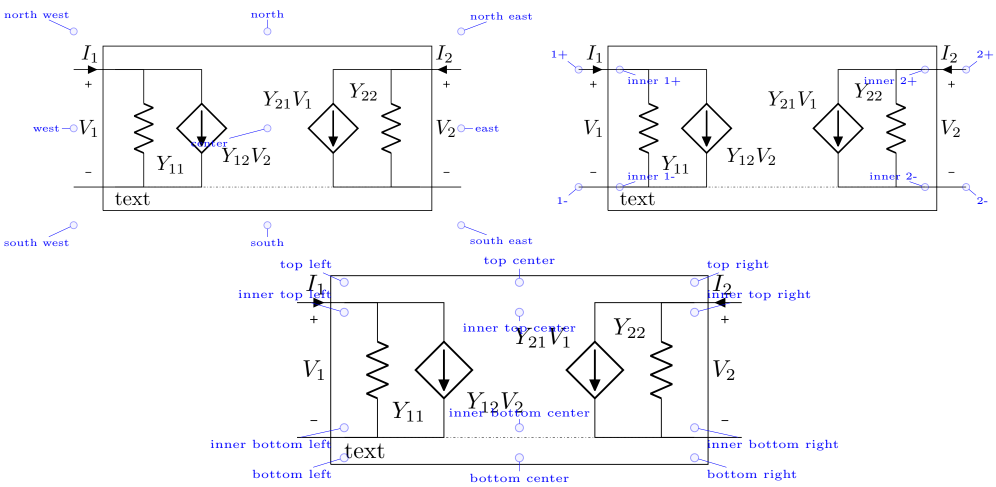
3.4 Quad Y

```
% Node use
node[Quad Y]{}

% To path use
(A) to[ToQuad Y]  (B)
```

This shape, besides the base anchors (see 3) it has 4 internal nodes: *<node>-Y11*, *<node>-Y12*, *<node>-Y21* and *<node>-Y22* and each of those sub-nodes has geographic anchors as defined at 2.1.

Note: There is also a *ToQuad Y* 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.



3.4.1 Quad Y keys

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

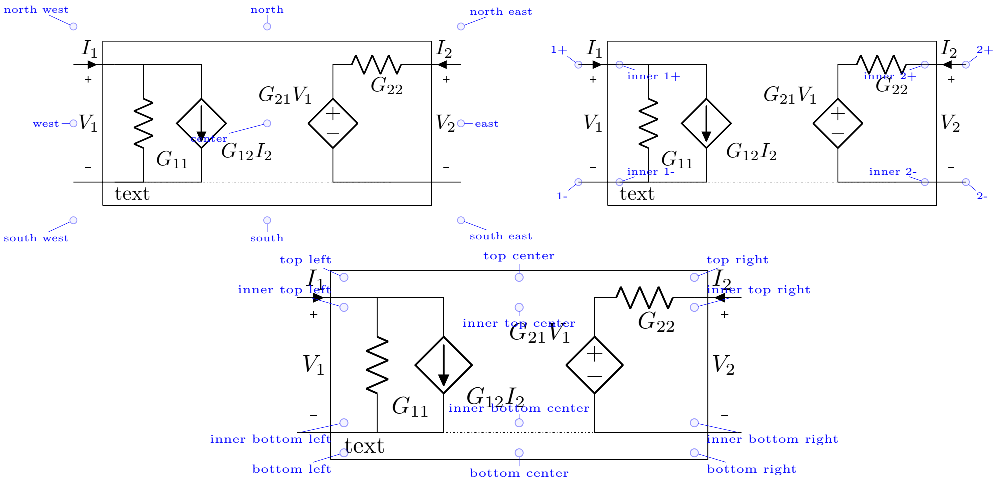
3.5 Quad G

```
% Node use
node[Quad G]{}

% To path use
(A) to[ToQuad G]  (B)
```

This shape, besides the base anchors (see 3) it has 4 internal nodes: *<node>-G11*, *<node>-G12*, *<node>-G21* and *<node>-G22* and each of those sub-nodes has geographic anchors as defined at 2.1.

Note: There is also a *ToQuad G* 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.



3.5.1 Quad G keys

name	⟨node-name⟩, when using a to[] path.
I1	Initial value:`$I_1$`
I2	Initial value:`$I_2$`
V1	Initial value:`$V_1$`
V2	Initial value:`$V_2$`
G11	Initial value:`$G_{11}$`
G12	Initial value:`$G_{12}$`
G21	Initial value:`$G_{21}$`
G22	Initial value:`$G_{22}$`
G11 label pos	changes the label position. Defaults to: `{south west}{top left}`
G12 label pos	changes the label position. Defaults to: `{south east}{top left}`
G21 label pos	changes the label position. Defaults to: `{north west}{bottom right}`
G22 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.

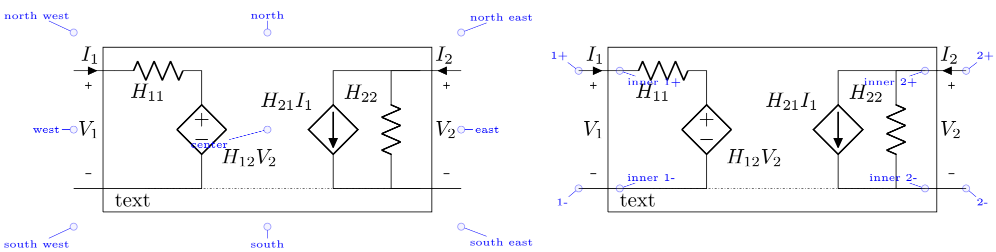
3.6 Quad H

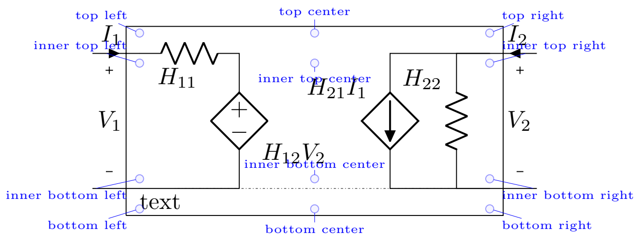
```
% Node use
node[Quad H]{}

% To path use
(A) to[ToQuad H]  (B)
```

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[ ]* path, in which case the key *outer x fit to* style will be triggered with the starting and ending points of the *to[ ]* path.





3.6.1 Quad H keys

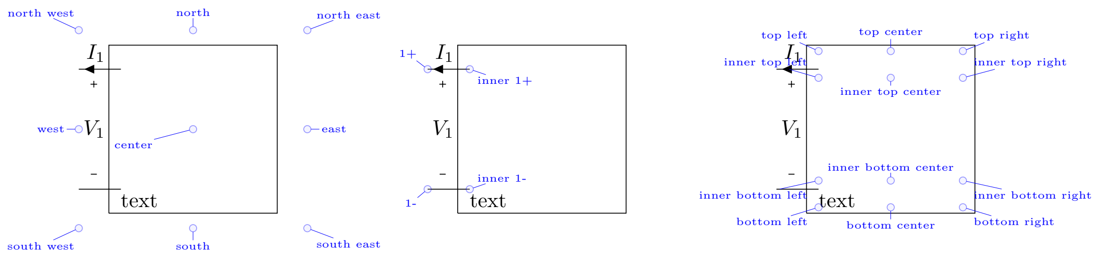
name	⟨node-name⟩, when using a to[] path.
I1	Initial value:`$I_1$`
I2	Initial value:`$I_2$`
V1	Initial value:`$V_1$`
V2	Initial value:`$V_2$`
H11	Initial value:`$H_{11}$`
H12	Initial value:`$H_{12}$`
H21	Initial value:`$H_{21}$`
H22	Initial value:`$H_{22}$`
H11 label pos	changes the label position. Defaults to: `{south west}{top left}`
H12 label pos	changes the label position. Defaults to: `{south east}{top left}`
H21 label pos	changes the label position. Defaults to: `{north west}{bottom right}`
H22 label pos	changes the label position. Defaults to: `{north west}{bottom 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.

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−, *inner* 1+, *inner* 1−, besides the geographic and text ones:



4.1.1 Base Keys

These applies to all *Black Box* shapes:

base width	The 'box' width
half base width	Ditto, half width. Initial value:`2\pgf@circ@Rlen`.
base height	The distance between *1+* and *1-*. The 'box' full height is equal to 2*(*half base height* + *height ext* + *height ext+*).
half base height	Ditto, half height. Initial value:`\pgf@circ@Rlen/7`
height ext	Initial value:`2\pgf@circ@Rlen/7`
height ext+	Initial value:0
inner ext	distance between the 'box' and `inner1+/1-/2+/2-`. initial value: `\pgf@circ@Rlen/7`
outer ext	distance between the 'box' and `1+/1-/2+/2-`. initial value: `5\pgf@circ@Rlen/14`
inner marks	If set, the inner anchors will be marked.
outer marks	If set, the outer anchors will be marked.

invert	The shape will be inverted, more or less like 'x scale=-1'.
alt, opt	Case a Voltage source is zero, a series impedance will be draw vertically.
outer x fit to	For any Black Box, this is the same as *outer x fit to!*.
*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.
outer x fit to!	*outer 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 be the same as ⟨CoordA⟩ and ⟨CoordB⟩. This will never result in a shape rotation.
inner x fit to	For any Black Box, this is the same as *inner x fit to!*.
*inner x fit to**	*inner x fit*=*{⟨CoordA⟩}{⟨CoordB⟩}. The width will be set so that ⟨inner 1+⟩ and ⟨inner 2+⟩ (or ⟨inner 1-⟩ and ⟨inner 2-⟩, depending on the used anchor) **will fit** ⟨CoordA⟩ and ⟨CoordB⟩. This might result in a shape rotation.
inner x fit to!	*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	For any Black Box, this is the same as *y fit to**.
*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 to!	*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.
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

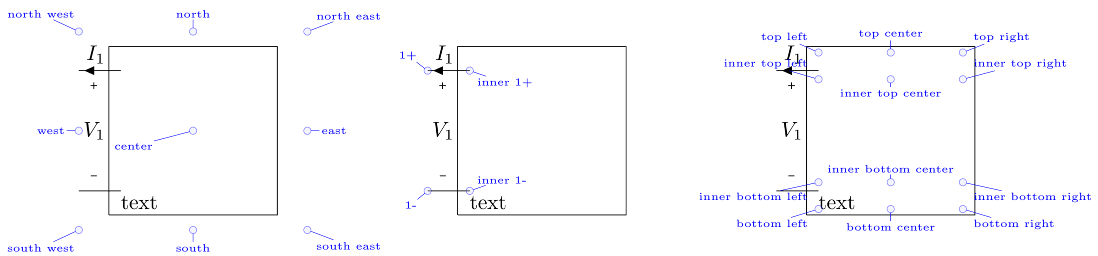
4.2 Black Box

```
% Node use
node[Black Box]{}

% To path use
(A) to[ToBlack Box]  (B)
```

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.



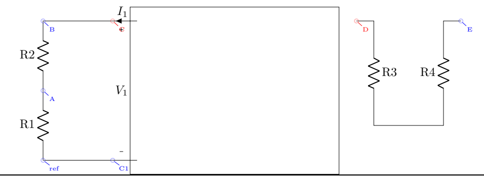
4.2.1 Black Box keys

name	⟨node-name⟩, when using a to[] path.
I1	Initial value:`$I_1$`
V1	Initial value:`$V_1$`

4.2.2 Examples of *fit to* use

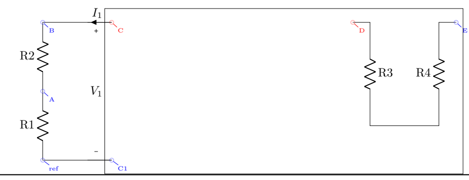
Squeezing a Black Box between two parts of a circuit (nodes C and D):

```
\resizebox{0.4\textwidth}{!}{
\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(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[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):

```
\resizebox{0.4\textwidth}{!}{
\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(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[Black Box,anchor=inner 1+,y fit to={C}{C1},inner x fit to={C}{E}]{};
\end{tikzpicture}
}
```



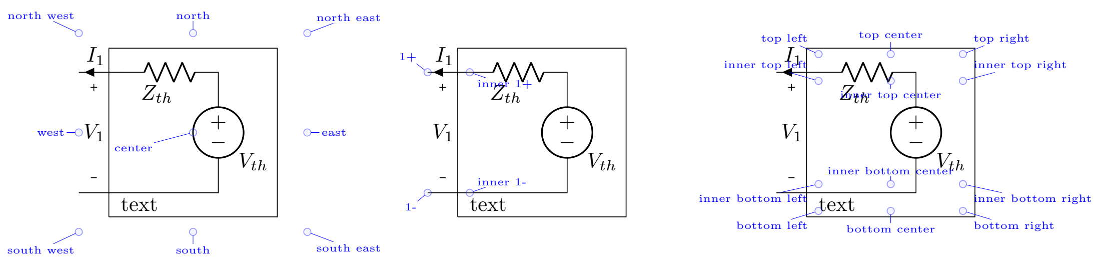
4.3 Thevenin

```
% 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

name	⟨node-name⟩, when using a to[] path.
I1	Initial value:`$I_1$`
V1	Initial value:`$V_1$`
Zth	Initial value:`$Z_{th}$`
Vth	Initial value:`$V_{th}$`
Zth label pos	changes the label position. Defaults to: `{south west}{top left}`
Vth label pos	changes the label position. Defaults to: `{south east}{top left}`

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.

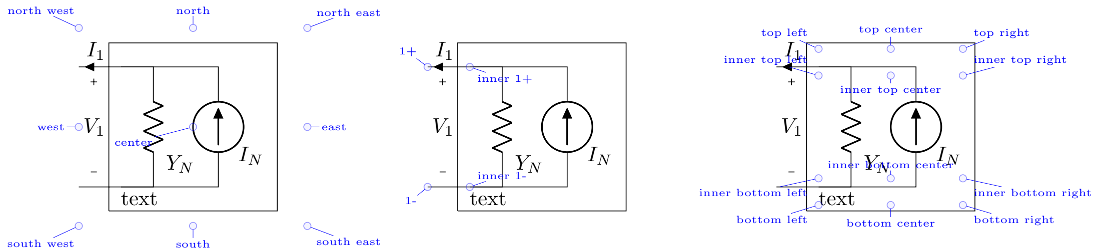
4.4 Norton

```
% Node use
node[Norton]{}

% To path use
(A) to[ToNorton]  (B)
```

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

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}`

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.

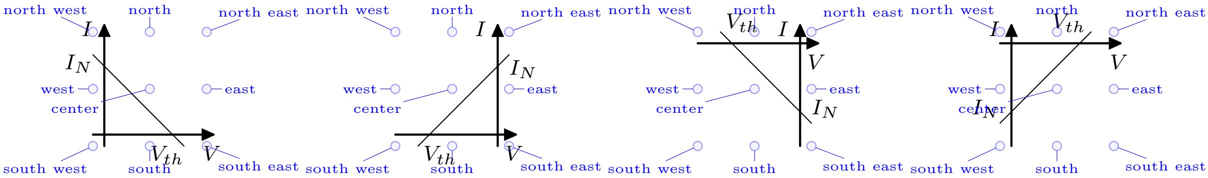
5 Pseudo-Graph Shape

```
% Node use
node[PG load line]{}

node[PG linear load line]
```

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:

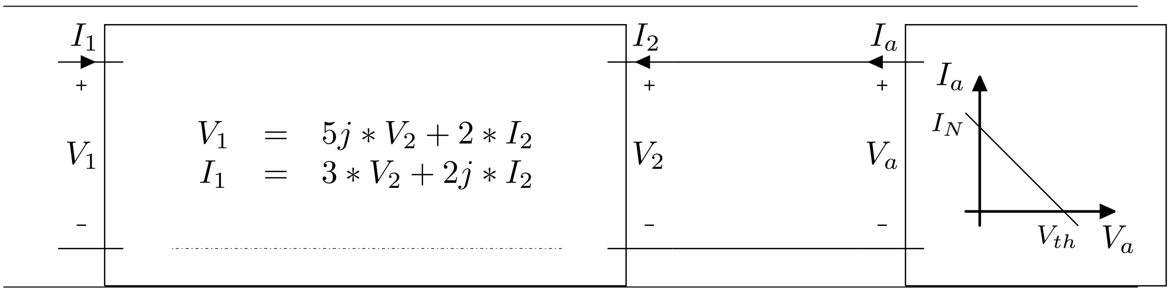
x axis	X axis *name*. Initial value: V
x val	X axis *val* at the crossing point. Initial value: V_{th}
y axis	Y axis *name*. Initial value: I
y val	Y axis *val* at the crossing point. Initial value: I_N
first quadrant	First quadrant mock up. (which is also the default).
second quadrant	Second quadrant mock up.
third quadrant	Third quadrant mock up.
fourth quadrant	Fourth quadrant mock up.
base width	The *graph* width
half base width	Ditto, half width. Initial value:`0.5\pgf@circ@Rlen`.
base height	The *graph* height
half base height	Ditto, half height. Initial value:`0.5\pgf@circ@Rlen`.

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

6 Examples of use

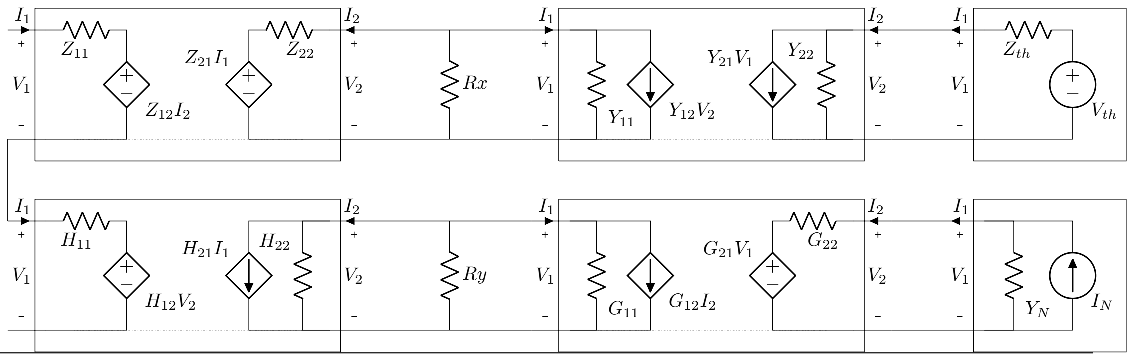
First of, a simple case of combining a generic Quad with equations and a generic Black Box with a Pseudo-Graph:

```
\resizebox{\textwidth}{!}{
\begin{tikzpicture}
  \draw (0,0) \ncoord(ref) node[Quad,anchor=1+](Q1){}
    (Q1.2+) -- ++(1,0) \ncoord(X) -- ++(1,0) node[Black Box,anchor=1+,V1=$V_a$,I1=$I_a$](B1){}
    (Q1.2-) -- (B1.1-)
    (B1.center) node[PG linear load line,x axis=$V_a$,y axis=$I_a$]{}
    (Q1.center) node{$ \begin{matrix}
          V_1 &=& 5j*V_2 + 2*I_2 \\
          I_1 &=& 3*V_2 + 2j*I_2
        \end{matrix} $%
    }  ;
\end{tikzpicture}
}
```



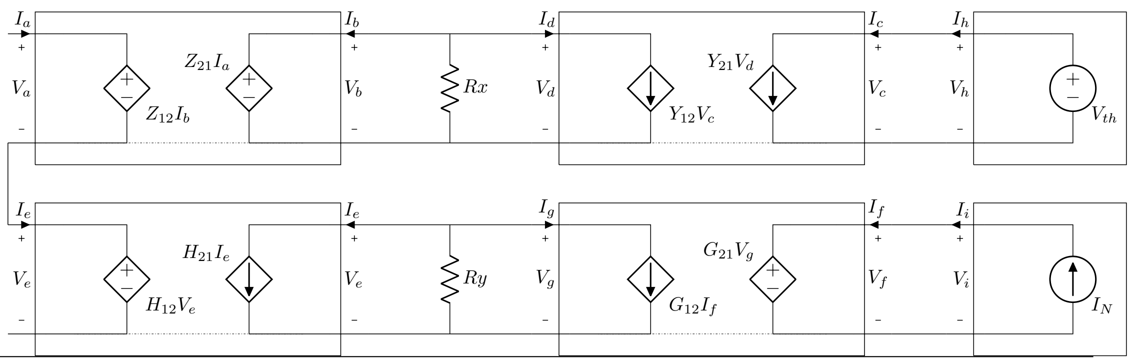
All default Quadripoles and Thevenin/Norton.

```
\resizebox{\textwidth}{!}{
\begin{tikzpicture}
  \draw (0,0) \ncoord(ref) node[Quad Z,anchor=1+](Qz1){}
    (Qz1.2+) -- ++(1.5,0) \ncoord(X) -- ++(1.5,0) node[Quad Y,anchor=1+](Qy1){}
    (Qy1.2+) -- ++(1,0) node[Thevenin,anchor=1+](th1){}
    (Qz1.1-) -- ++(0,-1.5) node[Quad H,anchor=1+](Qh1){}
    (Qh1.2+) -- ++(1.5,0) \ncoord(Y) -- ++(1.5,0) node[Quad G,anchor=1+](Qg1){}
    (Qg1.2+) -- ++(1,0) node[Norton,anchor=1+](nr1){}
    (Qz1.2-) -- (Qy1.1-) (Qy1.2-) -- (th1.1-)
    (Qh1.2-) -- (Qg1.1-) (Qg1.2-) -- (nr1.1-)
    ;
  \draw (X) to[R=$Rx$] (X |- Qz1.2-)
        (Y) to[R=$Ry$] (Y |- Qh1.2-)
        ;
\end{tikzpicture}
}
```



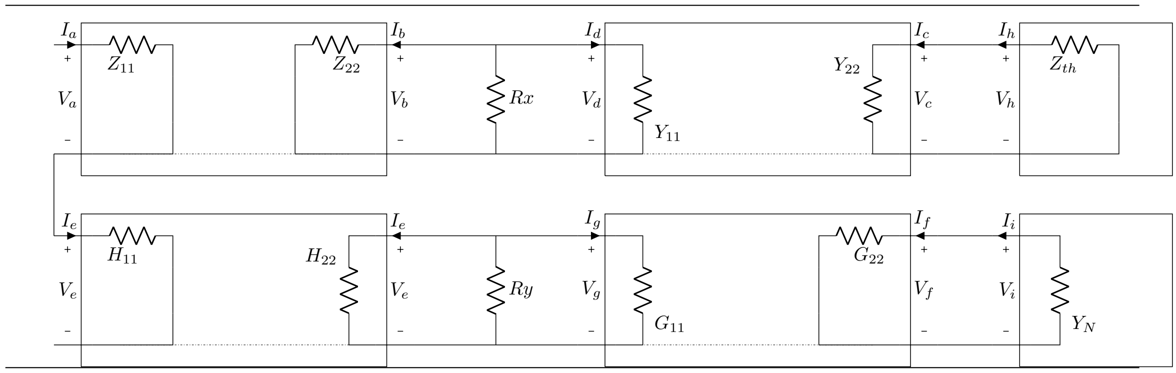
The same demo but with all parameter 11 and 22 zeroed, and changing the "control sources"

```
\resizebox{\textwidth}{!}{
\begin{tikzpicture}
  \draw (0,0) \ncoord(ref) node[Quad Z,anchor=1+,Z11=0,Z22=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+,Y11=0,Y22=0,I1=$I_d$,V1=$V_d$,I2=$I_c$,V
      2=$V_c$](Qy1){}
    (Qy1.2+) -- ++(1,0) node[Thevenin,anchor=1+,Zth=0,I1=$I_h$,V1=$V_h$](th1){}
    (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){}
    (Qh1.2+) -- ++(1.5,0) \ncoord(Y) -- ++(1.5,0) node[Quad G,anchor=1+,G11=0,G22=0,I1=$I_g$,V1=$V_g$,I2=$I_f$,V
      2=$V_f$](Qg1){}
    (Qg1.2+) -- ++(1,0) node[Norton,anchor=1+,Yn=0,I1=$I_i$,V1=$V_i$](nr1){}
    (Qz1.2-) -- (Qy1.1-) (Qy1.2-) -- (th1.1-)
    (Qh1.2-) -- (Qg1.1-) (Qg1.2-) -- (nr1.1-)
    ;
  \draw (X) to[R=$Rx$] (X |- Qz1.2-)
        (Y) to[R=$Ry$] (Y |- Qh1.2-)
        ;
\end{tikzpicture}
}
```



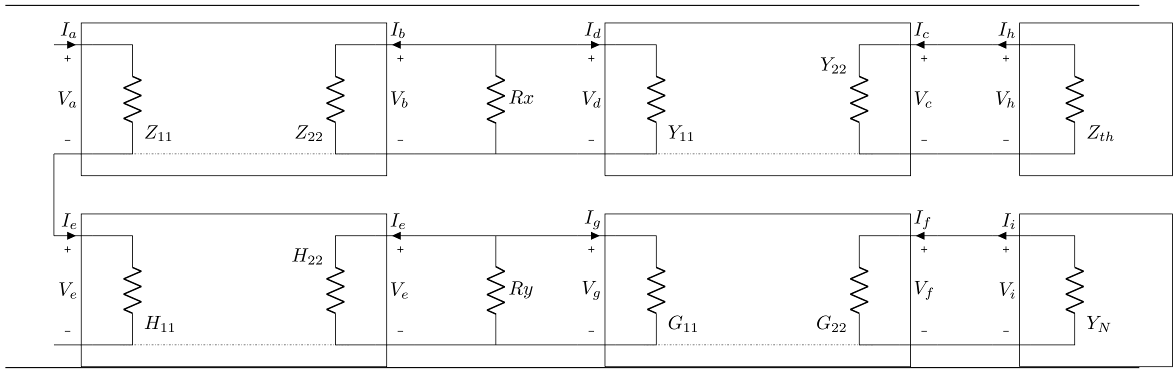
Now with the 12 and 21 parameters zeroed, normal form:

```
\resizebox{\textwidth}{!}{
\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=$I_g$,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){}
    (Qz1.2-) -- (Qy1.1-) (Qy1.2-) -- (th1.1-)
    (Qh1.2-) -- (Qg1.1-) (Qg1.2-) -- (nr1.1-)
    ;
  \draw (X) to[R=$Rx$] (X |- Qz1.2-)
        (Y) to[R=$Ry$] (Y |- Qh1.2-)
        ;
\end{tikzpicture}
}
```



Same as last one, but with an alternate form:

```
\resizebox{\textwidth}{!}{
\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=$I_g$,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){}
    (Qz1.2-) -- (Qy1.1-) (Qy1.2-) -- (th1.1-)
    (Qh1.2-) -- (Qg1.1-) (Qg1.2-) -- (nr1.1-)
    ;
  \draw (X) to[R=$Rx$] (X |- Qz1.2-)
        (Y) to[R=$Ry$] (Y |- Qh1.2-)
        ;
\end{tikzpicture}
}
```



Setting all parameters,some impedances as zig-zag, others as generic, per quadripole:

```
\resizebox{\textwidth}{!}{
\begin{tikzpicture}
  \draw (0,0) \ncoord(ref) node[Quad Z,alt,round sources,european,anchor=1+,Z11=$Z_a$,Z22=$Z_b$,Z12=$Z_{re}$,Z21=$Z_{fe}$,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,alt,anchor=1+,Y11=$Y_a$,Y22=$Y_b$,Y12=$Y_{re}$,Y21=$Y_{fe}$,I1=$I_d$,V1=$V_d$,I2=$I_c$,V2=$V_c$](Qy1){}
    (Qy1.2+) -- ++(1,0) node[Thevenin,alt,anchor=1+,Vth=$V_1$,Zth=$Z_a$,I1=$I_h$,V1=$V_h$](th1){}
    (Qz1.1-) -- ++(0,-1.5) node[Quad H,european,alt,anchor=1+,H11=$H_a$,H22=$H_b$,H12=$H_{re}$,H21=$H_{fe}$,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,alt,anchor=1+,G11=$G_a$,G22=$G_b$,G12=$G_{re}$,G21=$G_{fe}$,I1=$I_g$,V1=$V_g$,I2=$I_f$,V2=$V_f$](Qg1){}
    (Qg1.2+) -- ++(1,0) node[Norton,alt,control sources,european,anchor=1+,In=$I_b$,Yn=$Y_b$,I1=$I_i$,V1=$V_i$](nr1){}
    (Qz1.2-) -- (Qy1.1-) (Qy1.2-) -- (th1.1-)
    (Qh1.2-) -- (Qg1.1-) (Qg1.2-) -- (nr1.1-)
    ;
  \draw (X) to[R=$Rx$] (X |- Qz1.2-)
        (Y) to[R=$Ry$] (Y |- Qh1.2-)
        ;
\end{tikzpicture}
}
```