tzplot.sty
Plot Graphs with TikZ Abbreviations

In-Sung Cho
ischo <at> ktug.org
Economics, Kongju National University
2021/03/20  version 1.0.1

Abstract

This is a \LaTeX package that provides TikZ based macros to make it easy to draw graphs. The macros provided in the \texttt{tzplot} package are just abbreviations for TikZ codes, which can be complicated, but using the package, hopefully, makes drawing easier, especially when drawing repeatedly. The macros were chosen and developed with an emphasis on drawing graphs in \textit{economics}.

\textbf{Keywords:} lines, dots, curves, axes, functions, projection, ticks, intersections, tangent lines

Contents

Part I  Getting Ready

1 Introduction 1
1.1 About \texttt{tzplot.sty} .................................. 1
1.2 Preoccupied style names .................................. 1
1.3 How to read this document ................................. 2

Part II  Getting Started

2 An Intuitive Introduction I: Basics 3
2.1 Lines: Basics: \texttt{tzline(0,0)(3,1)} .................. 3
2.2 Dots: Basics ............................................. 3
2.2.1 A circle dot: \texttt{tzcdot(0,0)} .................... 3
2.2.2 A circle node dot: \texttt{tzdot(0,0)} ............... 4
2.2.3 Difference between \texttt{tzcdot} and \texttt{tzdot} .... 4
2.3 Coordinates: Basics: \texttt{tzcoor(0,0)(A)} .......... 5
2.4 Curves: Basics .......................................... 6
2.4.1 \texttt{tzto(0,0)(4,2)} .................................. 6
2.4.2 \texttt{tzbezier} ...................................... 6
2.4.3 \texttt{tzparabola} .................................... 6
2.5 Adding text: Nodes and placement ...................... 7
2.5.1 \texttt{tznode(3,1){text}[right]} .................... 7
2.5.2 Review: Main nodes and label nodes in TikZ .......... 7
2.5.3 Abbreviations of TikZ basic placement options: a, b, l, r, ar, bl, etc... 8
2.6 Labeling dots and coordinates ......................... 8
2.6.1 \texttt{tzdot, tzcdot} ................................. 8
2.6.2 \texttt{tzcoor} .......................... 8
2.6.3 \texttt{tzcoor*} ...................................... 9
5 Examples: Economics
5.1 Markets .................................................. 28
   5.1.1 Market equilibrium: step by step .............. 28
   5.1.2 Tax incidence: step by step ................. 29
5.2 Firms ..................................................... 30
17.3 Display the origin ................................. 102
  17.3.1 \tzshoworigin ............................... 102
  17.3.2 \tzshoworigin* ............................. 102
17.4 \tzaxesL(‘): L-type axes ........................ 103

18 Ticks
  18.1 \tzticks: Tick labels .......................... 104
  18.2 \tzticks*: Tick marks .......................... 106
  18.3 \tzticksx(*) and \tzticksy(*) ................. 106

19 Projections
  19.1 \tzproj(*): Intersection points ............... 108
  19.2 \tzprojx(*) and \tzprojy(*) .................. 109

20 Plot Functions
  20.1 \tzfn: Plot functions .......................... 110
    20.1.1 Syntax ..................................... 110
    20.1.2 Define and name functions ................ 110
    20.1.3 Name paths: name path .................... 111
    20.1.4 Move graphs: shift ......................... 111
    20.1.5 Extend paths: <code.append>, \tzfnAtBegin, \tzfnAtEnd ............... 112
  20.2 \tzLFn: Plot linear functions ................. 113
  20.3 Horizontal lines ................................ 114
    20.3.1 \tzhfnat .................................. 114
    20.3.2 \tzhfn .................................... 115
  20.4 Vertical lines ................................ 115
    20.4.1 \tzvfnat .................................. 115
    20.4.2 \tzvfn .................................... 116

21 Intersections
  21.1 \tzXpoint(*): Intersection points .............. 117
  21.2 Vertical intersection points ................... 118
    21.2.1 \tzvXpointat(*) ................................ 118
    21.2.2 \tzvXpoint(*) ................................ 119
  21.3 Horizontal intersection points .................. 119
    21.3.1 \tzhXpointat(*) ................................ 119
    21.3.2 \tzhXpoint(*) ................................ 120

22 Secant and Tangent Lines
  22.1 Secant lines ................................. 121
    22.1.1 \tzsecantat .................................. 121
    22.1.2 \tzsecant ................................... 123
  22.2 Tangent lines ................................ 123
    22.2.1 \tztangentat .................................. 123
    22.2.2 \tztangent ................................... 125

23 Miscellany
  23.1 \tzbrace(‘) ..................................... 127

Version history ............................... 129
Acknowledgement ............................. 129
Reference .................................. 129
Index ...................................... 130
Part I
Getting Ready

1 Introduction

1.1 About tzplot.sty

The tzplot package is just a collection of macros based on TikZ to save you time typing TikZ code.

In pstricks, a line connecting two points (A) and (B) is drawn by \psline(A)(B). With the package tzplot, you can do it by \tzline(A)(B).

\tzline(A)(B) \% is an abbreviation of:
\draw (A) -- (B);

\tzline[blue](A)(B){my line}[r] \% is an abbreviation of:
\draw [blue] (A) -- (B) node [right] {my line};

Some macros in this package represent one or a few lines of code, but some represent dozens of lines of TikZ code.

All of the drawing macros of tzplot are prefixed by \tz. Of course, it means TikZ. The syntax of the tzplot macros comes from tikz and pstricks. However, the input mode is more like pstricks.

To use the tzplot package you have to load the package in the preamble of your document as follows:

\usepackage{tzplot}

The package depends on the packages tikz, xparse, and expl3. And it uses the following TikZ libraries:

calc,backgrounds,positioning,intersections,arrows,shapes,patterns,plotmarks,
decorations.pathreplacing,calligraphy

This package sets the basic arrow style to stealth. If you don’t like this, as an alternative, you can set the style like \tikzset{>=to} after the tzplot package is loaded.

This package was originally motivated by drawing graphs in economics. Therefore, the macros in this package have been selected and developed for drawing graphs efficiently in economics. However, this package will do a good job of drawing basic graphics in any fields.

Finally, note that this is far from a TikZ tutorial. To make good use of this package, you need to familiarize yourself with TikZ.

1.2 Preoccupied style names

This package does not provide any environment. Since all the drawing macros prefixed by \tz are just abbreviations of TikZ code, you can use the macros in the tikzpicture environment together with any TikZ commands.

However, there are some preoccupied style names that you should not overwrite. Those are as follows:

<table>
<thead>
<tr>
<th>tzdot</th>
<th>tzmark</th>
<th>tzdotted</th>
</tr>
</thead>
<tbody>
<tr>
<td>tzdashed</td>
<td>tzhelplines</td>
<td>tznode</td>
</tr>
<tr>
<td>tzshorten</td>
<td>tzextend</td>
<td>tzshowcontrols</td>
</tr>
</tbody>
</table>
This package also predefined *abbreviations* of TikZ’s basic placement options as follows:

```
% preoccupied (alias) styles
\tikzset{%
a/.style={above=#1},
b/.style={below=#1},
c/.style={centered=#1},
l/.style={left=#1},
r/.style={right=#1},
al/.style={above left=#1},
ar/.style={above right=#1},
bl/.style={below left=#1},
br/.style={below right=#1},
}
```

You can use these abbreviations to place TikZ’s *main nodes* but not *label nodes*. To place label nodes, use non-abbreviated options or angles instead.

The `tzplot` package also defines graphic layers as follows:

```
\pgfdeclarelayer{background}
\pgfdeclarelayer{behind}
\pgfdeclarelayer{above}
\pgfdeclarelayer{foreground}
\pgfsetlayers{background,behind,main,above,foreground}
```

Therefore, you can select the graphic layers in sequence: `background`, `behind`, `main`, `above`, and `foreground`. For example, you can change the layer of a straight line from `main` (default) to `background` as follows:

```
\begin{tikzpicture}
  \zhelplines(4,3)
  \begin{pgfonlayer}{background}
    \tzline[blue](0,0)(3,1)
  \end{pgfonlayer}
\end{tikzpicture}
```

### 1.3 How to read this document

In drawing graphs, too many factors are involved: line style, color, fill, label, positioning, shift, and so on. Almost all macros of this package have many arguments that control these factors. Some are mandatory and some are optional. Optional arguments are hidden when not used.

The document has three essential parts: Part II, Part III, and Part IV. Part II introduces essential macros with only frequently used options. There are many options hidden in the macros introduced in Part II. Some macros are not introduced in Part II. Part III and IV describe all the features of all macros.

You must get started with Part II. Part II is sufficient for drawing needs in most cases.

Unless you are an experienced user of TikZ, it is recommended to move on to Part III and Part IV once you become familiar with Part II. In the meantime, use Part III and Part IV for reference only. Use the list of contents and the index efficiently to find macros you need.
2 An Intuitive Introduction I: Basics

All drawing macros provided in this package work within \texttt{tikzpicture} environment, just like any other TikZ commands.

2.1 Lines: Basics: \texttt{\textbackslash tzline(0,0)(3,1)}

To draw a line from (0,0) to (3,1), just do \texttt{\textbackslash tzline(0,0)(3,1)}.

\begin{verbatim}
\begin{tikzpicture}
\tzhelplines(4,2)
\tzline(0,0)(3,1)
\end{tikzpicture}
\end{verbatim}

You can use TikZ options to change the style of a line.

\begin{verbatim}
% \texttt{\textbackslash tzline}: change line style
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline(0,0)(3,1)
\tzline[dashed](0,1)(3,2)
\tzline[->,blue,thick](0,3)(4,1)
\end{tikzpicture}
\end{verbatim}

\texttt{\textbackslash tzline[dashed](0,1)(3,2)} \texttt{\% works like:}
\texttt{\draw [dashed] (0,1) -- (3,2);} 

2.2 Dots: Basics

2.2.1 A circle dot: \texttt{\textbackslash tzcdot(0,0)}

\texttt{\textbackslash tzcdot(0,0)} prints a ‘circle dot’ *, with the radius 1.2pt by default, at the point (0,0). The starred version \texttt{\textbackslash tzcdot*} prints a filled dot *.

\begin{verbatim}
% \texttt{\textbackslash tzcdot: circle dot}
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcdot(0,0)
\tzcdot*(3,1)
\tzcdot[red](0,1)(3pt) \% radius=3pt
\tzcdot*[green](3,2)(3pt) \% radius=3pt
\end{tikzpicture}
\end{verbatim}

You can change of the size of a dot by specifying the \texttt{radius} of the circle, like, for example, \texttt{\tzcdot(0,0)(3pt)}. 

3
\texttt{\textbackslash tzcdot*[green](3,2)(3pt)} % is an abbreviation for:
\begin{verbatim}
\draw [fill,green] (0,0) circle (3pt);
\end{verbatim}

\begin{verbatim}
\begin{tikzpicture}
\helplines(4,3)
\tzdot(0,0) \tzdot*[3,1]
\tzdot[red](0,1)(2mm) \textcolor{red}{% radius=2mm}
\tzdot*[green](3,2)(3pt) \textcolor{green}{% radius=3pt}
\tzline(0,0)(3,1)
\tzline(0,1)(3,2)
\end{tikzpicture}
\end{verbatim}

### 2.2.2 A circle node dot: \texttt{\tzdot(0,0)}

\texttt{\tzdot} draws a ‘circle node dot’ at a specified coordinate. The starred version \texttt{\tzdot*} prints a filled dot. The default size (\texttt{diameter} or \texttt{minimum size}) is 2.4pt.

\begin{verbatim}
\begin{tikzpicture}
\helplines(4,3)
\tzdot(0,0) \tzdot*[3,1]
\tzdot[red](0,1)(4mm) \textcolor{red}{% minimum size=4mm}
\tzdot*[green](3,2)(6pt) \textcolor{green}{% minimum size=6pt}
\tzline(0,0)(3,1)
\tzline(0,1)(3,2)
\end{tikzpicture}
\end{verbatim}

The size (\texttt{diameter} or \texttt{minimum size}) of a node dot can be changed by the second (or the last) parenthesis option, like (6pt).

\begin{verbatim}
\begin{tikzpicture}
\helplines(4,3)
\tzdot(0,0)(5pt) \tzdot*[3,1]
\tzdot[red](0,1)(4mm) \textcolor{red}{\tzdot*[green](3,2)(6pt)}
\tzline(0,0)(3,1)
\tzline(0,1)(3,2)
\end{tikzpicture}
\end{verbatim}

\texttt{\tzdot[green](0,0)(6pt)} % works like:
\begin{verbatim}
\draw [green] (0,0) node [\tzdot,minimum size=6pt] {};
\end{verbatim}

\% \tzdot style is predefined

### 2.2.3 Difference between \texttt{\tzcdot} and \texttt{\tzdot}

A ‘circle dot’ drawn by \texttt{\tzcdot} is affected by \texttt{xscale} or \texttt{yscale} in Ti\textit{k}Z, but a ‘circle node dot’ drawn by \texttt{\tzdot} is not. Note also that \texttt{\tzcdot} controls the \texttt{radius} of a circle dot (following Ti\textit{k}Z practice), while \texttt{\tzdot} controls the \texttt{diameter} of a circle node dot.
Remark: The circle dots drawn by `\tzdot` are affected by TikZ scale factors. It gets bigger or smaller by `scale`. Let us see what happens to circle dots, especially when `xscale` and `yscale` are not symmetric.

\begin{tikzpicture}[xscale=.5,yscale=1.1]
\helplines(4,3)
\tzdot(0,0) \tzdot* (3,1)
\tzdot[red](0,1)(4mm) \tzdot*[green](3,2)(6pt)
\tzline(0,0)(3,1)
\tzline(0,1)(3,2)
\end{tikzpicture}

\textbf{5. Coordinates: Basics: `\tzcoor(0,0)(A)`}

To define a coordinate, use `\tzcoor` with a coordinate followed by its name in parentheses.

\begin{tikzpicture}
\helplines(4,3)
\tzcoor(0,0)(A) \tzcoor(60:3cm)(B)
\tzline[->](A)(B)
\tzcoor(0,0)(A) \tzcoor(60:3cm)(B)
\tzline[->](A)(B)
\tzcoor(0,1)(C) \tzcoor(4,2)(D)
\tzline[dashed](C)(D)
\end{tikzpicture}

The starred version `\tzcoor*` prints a filled node dot at a specified coordinate.

\begin{tikzpicture}
\helplines(4,3)
\tzcoor(0,0)(A) \tzcoor(60:3cm)(B)
\tzline[->](A)(B)
\tzcoor(0,0)(A) \tzcoor(60:3cm)(B)
\tzcoor(0,1)(C) \tzcoor*[fill=none](4,2)(D)
\tzline[dashed](C)(D)
\end{tikzpicture}
2.4 Curves: Basics

2.4.1 \texttt{tzto}(0,0)(4,2)

\texttt{tzto} connects two points with a line or a curve.

\begin{tikzpicture}
\draw[help lines] (4,3);
\draw (0,0) to (4,2);
\draw[bend right,dashed] (0,1) to (3,2);
\draw[out=90,in=-90,->,blue] (0,2) to (4,1);
\end{tikzpicture}

\texttt{tzto[bend right]}(0,1)(3,2) % works like:
\begin{tikzpicture}
\draw[bend right] (0,1) to (3,2);
\end{tikzpicture}

2.4.2 \texttt{tzbezier}

\texttt{tzbezier} draws a Bézier curve with one or two control points. The style \texttt{tzshowcontrols}, which is predefined in the package, reveals the control point(s).

\begin{tikzpicture}
\draw[blue,thick] (0,0) .. controls (2,0) .. (4,2);
\draw[->,tzshowcontrols] (0,2) .. controls (1,3) and (3,0) .. (4,1);
\end{tikzpicture}

\texttt{tzbezier[blue]}(0,0)(2,0)(4,2) % works like:
\begin{tikzpicture}
\draw[blue] (0,0) .. controls (2,0) .. (4,2);
\end{tikzpicture}

\texttt{tzbezier[->,tzshowcontrols]}(0,2)(1,3)(3,0)(4,1) % works like:
\begin{tikzpicture}
\draw (0,2) .. controls (1,3) and (3,0) .. (4,1);
\end{tikzpicture}

2.4.3 \texttt{tzparabola}

\texttt{tzparabola} draws a parabola controlled by several options of TikZ’s \texttt{parabola} operation.

\begin{tikzpicture}
\draw (0,0) parabola (2,4);
\draw[blue] (0,0) parabola (2,4);
\draw[->,tzshowcontrols] (0,2) parabola bend (3,3) (4,1) % works like:
\end{tikzpicture}

\texttt{tzparabola}(0,0)(2,4) % works like:
\begin{tikzpicture}
\draw (0,0) parabola (2,4);
\end{tikzpicture}

\texttt{tzparabola}(2,0)(3,3)(4,1) % works like:
\begin{tikzpicture}
\draw (2,0) parabola bend (3,3) (4,1);
\end{tikzpicture}
2.5 Adding text: Nodes and placement

2.5.1 `\tznodere(3,1){text}[right]`

With `\tznodere(<coor>){<text>}[<node opt>]`, you can put some text at a specified position. The starred version `\tznodest` draws the node perimeter, which is a rectangle by default.

```
\begin{tikzpicture}
  \tzhelplines(4,4)
  \tznodere(0,0){A}
  \tznodest(1,1){text}
  \tdotst(2,2)
  \tznodere(2,2){text}[above right,blue]
\end{tikzpicture}
```

```
\tznodere(0,0){A}; % works like:
\node at (0,0) {A};
\tznodere(2,2){text}; % works like:
\draw (2,2) node [above right,blue] {text};
```

2.5.2 Review: Main nodes and label nodes in TikZ

In TikZ, there are two kinds of nodes: main nodes and label nodes.

When a main node with text in it is placed at a specific point, its label node with a label in it is optionally placed in the direction of a designated angle relative to the main node.

```
\begin{tikzpicture}
  \tzhelplines(4,3)
  \tznodest(2,1){main node}[scale=2,label={[draw,red]90:label node}]
  \tznodere(3,3){A}[label={above:a},label={0:r}]
\end{tikzpicture}
```

The angles for label nodes can be replaced by the corresponding placement words. For example, the angle 90 degree for a label node can be replaced by `above`, 0 by `right`, -135 by `below left`, and the like. Note that the angle expression cannot be used for placing a main node.

Remark:

- In this package, macros related to ‘dots’ or ‘coordinates’ (like \tzcdot, \tdot, and \tzcro) can optionally have ‘label nodes,’ while macros related to ‘lines’ and ‘curves’ (like \tpline, \tzto, \tzbezier and \tzparabola) optionally have ‘main nodes’ or ‘text nodes.'
There is one exception:\texttt{\textbackslash tzshoworigin}. \texttt{\textbackslash tzshoworigin} can have a label, but its location is controlled by positional words such as \texttt{below left} but not by \texttt{<angle>}. (See Section 17.3 on page 102 for more details.)

### 2.5.3 Abbreviations of TiKZ basic placement options: a, b, l, r, ar, bl, etc.

You can use abbreviations a for \texttt{above}, r for \texttt{right}, bl for \texttt{below left}, and so on to place main nodes.

\begin{tikzpicture}[font=\ttfamily, text=blue]
\tzhelplines\[thick\](2,2)
\tznode\[centered\]\{draw\} \((1,1)\)\{a\} \tznode\{b\} \((1,0)\)
\tznode\{l\} \((0,1)\) \tznode\{r\} \((2,1)\)
\tznode\{al\} \((0,2)\) \tznode\{ar\} \((2,2)\)
\tznode\{bl\} \((0,0)\) \tznode\{br\} \((2,0)\)
\end{tikzpicture}

**Warning:** The abbreviations of TiKZ basic placement options \textit{cannot} be used to place labels. To place labels, use \texttt{angles} or the unabridged placement options of TiKZ such as \texttt{above} and \texttt{below left}.

### 2.6 Labeling dots and coordinates

#### 2.6.1 \texttt{\textbackslash tdot}, \texttt{\textbackslash tzdot}

To add a label to a dot generated by \texttt{\textbackslash tdot} or \texttt{\textbackslash tzdot}, you should specify, \textit{right after a coordinate}, \{<label>\} followed by \{<angle>\} (90 degree or above by default in TiKZ).

\begin{verbatim}
\begin{tikzpicture}
\tzhelplines\[4,3\]
\tzdot\*\{(1,1)\}\{A\} \% default: 90 or above
\tzdot\{(2,1)\}\{0\}\{0\}(4pt)
\tzdot\*\{(1,2)\}\{C\}\{180\}(2pt)
\tzdot\{(3,2)\}\{D\}\{45\}(4pt)
\tzdot\*\{(4,0)\}\{E\}\{[red,draw,180]\}(4pt)
\end{tikzpicture}
\end{verbatim}

\begin{verbatim}
\tzhelplines\[4,3\]
\tzdot\*\{(1,2)\}\{C\}\{180\}(2pt) \% works like:
\draw[fill] \{(1,2)\} circle (2pt) node [label={180:C}] \{};
\end{verbatim}

#### 2.6.2 \texttt{\textbackslash tzcoor}

\texttt{\textbackslash tzcoor} can add a label to a coordinate. Just append the optional arguments \{<label>\} and \{<angle>\} after the two mandatory parenthesis arguments.

**Remark:**

- **REMEMBER** the order of arguments is \((<\texttt{coor}>)(<\texttt{name}>)(<\texttt{label}>)[<\texttt{angle}>].
- It cannot be emphasized that \{<label>\} is placed by \{<angle>\} (by default 90 or above) and you \textit{cannot} use the abbreviations such as a, l, ar, and so forth to place labels for coordinates and dots.
You can see the full syntax of `\tzcoor` in Section 8.1 on page 47.

\begin{tikzpicture}
\tzhelplines(4,2)
\tzcoor(1,1)(A)(A) \ % default: 90 or above
\tzcoor(2,1)(B)(B)[[\text{red}0]]
\end{tikzpicture}

\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoor(1,1)(A)(A) \ % default: 90 or above
\tzcoor(2,1)(B)(B)[0]
\tzcoor*{(1,2)}(C){180}[4pt]
\tzcoor*[\text{fill=none}]{3,2}(D){D}[46]
\tzcoor*(4,0){E}[\text{red},\text{draw}][180][4pt]
\end{tikzpicture}

\begin{tikzpicture}
\tzhelplines(4,2)
\tzcoor(1,1)(A)(A)
\end{tikzpicture}

\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoor(1,1)(A)(A) \ % default: 90 or above
\tzcoor(2,1)(B)(B)[0]
\tzcoor*{(1,2)}(C){180}[4pt]
\tzcoor*[\text{fill=none}]{3,2}(D){D}[46]
\tzcoor*(4,0){E}[\text{red},\text{draw}][180][4pt]
\end{tikzpicture}

\begin{tikzpicture}
\tzhelplines(4,2)
\tzcoor(1,1)(A)(A)
\end{tikzpicture}

\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoor(1,1)(A)(A) \ % default: 90 or above
\tzcoor(2,1)(B)(B)[0]
\tzcoor*{(1,2)}(C){180}[4pt]
\tzcoor*[\text{fill=none}]{3,2}(D){D}[46]
\tzcoor*(4,0){E}[\text{red},\text{draw}][180][4pt]
\end{tikzpicture}

% defaults
/edit(tzcoor)
\begin{tikzpicture}
\tzhelplines(4,2)
\tzcoor(1,1)(A)(A) \ % default: 90 or above
\tzcoor(2,1)(B)(B)[[\text{red}0]]
\end{tikzpicture}

\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoor(1,1)(A)(A) \ % default: 90 or above
\tzcoor(2,1)(B)(B)[0]
\tzcoor*{(1,2)}(C){180}[4pt]
\tzcoor*[\text{fill=none}]{3,2}(D){D}[46]
\tzcoor*(4,0){E}[\text{red},\text{draw}][180][4pt]
\end{tikzpicture}

% defaults
/edit(tzcoor)
\begin{tikzpicture}
\tzhelplines(4,2)
\tzcoor(1,1)(A)(A)
\end{tikzpicture}

\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoor(1,1)(A)(A) \ % default: 90 or above
\tzcoor(2,1)(B)(B)[0]
\tzcoor*{(1,2)}(C){180}[4pt]
\tzcoor*[\text{fill=none}]{3,2}(D){D}[46]
\tzcoor*(4,0){E}[\text{red},\text{draw}][180][4pt]
\end{tikzpicture}

% defaults
/edit(tzcoor)
\begin{tikzpicture}
\tzhelplines(4,2)
\tzcoor(1,1)(A)(A)
\end{tikzpicture}

% defaults
/edit(tzcoor)
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoor(1,1)(A)(A) \ % default: 90 or above
\tzcoor(2,1)(B)(B)[0]
\tzcoor*{(1,2)}(C){180}[4pt]
\tzcoor*[\text{fill=none}]{3,2}(D){D}[46]
\tzcoor*(4,0){E}[\text{red},\text{draw}][180][4pt]
\end{tikzpicture}

\begin{tikzpicture}
\tzhelplines(4,2)
\tzcoor(1,1)(A)(A)
\end{tikzpicture}

% defaults
/edit(tzcoor)
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoor(1,1)(A)(A) \ % default: 90 or above
\tzcoor(2,1)(B)(B)[0]
\tzcoor*{(1,2)}(C){180}[4pt]
\tzcoor*[\text{fill=none}]{3,2}(D){D}[46]
\tzcoor*(4,0){E}[\text{red},\text{draw}][180][4pt]
\end{tikzpicture}

2.6.3 \texttt{tzcoor*}

The starred version `\tzcoor*` designates a coordinate and prints a node dot with a label around the designated point like `\tzdot*` does.

\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoor(1,1)(A)(A) \ % default: 90 or above
\tzcoor(2,1)(B)(B)[0]
\tzcoor*{(1,2)}(C){180}[4pt]
\tzcoor*[\text{fill=none}]{3,2}(D){D}[46]
\tzcoor*(4,0){E}[\text{red},\text{draw}][180][4pt]
\end{tikzpicture}

2.7 Adding text next to lines or curves

2.7.1 \texttt{tzline}

`\tzline` accepts two mandatory coordinates. To add text to a line segment, just specify the optional arguments `{<text>}` and `[<node opt>]` in-between the two coordinates. The `[<node opt>]` is `[above,\text{midway}]`, by default.
The optional argument \texttt{\{<text>\}} \textit{following the second coordinate} can also be used as a name of the graph. By default, it is placed at the second coordinate.

\begin{tikzpicture}
  \zhelplines(4,3)
  \tzline{(0,0)}{(route A)(3,0)}
  \tzline[dashed]{(1,1)}{(route B)(4,1)}
  \tzline[blue]{(0,3)}{[plan C] [below,near end] (3,3)}
\end{tikzpicture}

\begin{tikzpicture}
  \tzto{(plan C)}{(route A)}{(3,3)}{(route B)}{(0,3)}{(demand)}
  \tzto[dashed,bend left]{(1,1)}{(route B)(4,1)}
  \tzto[blue,bend right]{(0,3)}{[plan C] [below,near end] (3,3)}
\end{tikzpicture}

\begin{tikzpicture}
  \tzbezier{(0,3)}{(4,0)}{(demand)}
  \tzbezier{[bend right=15]}{(0,0)}{(3,3)}{(supply)}[r]
\end{tikzpicture}

\subsection{\texttt{\tzto}}

To add text to a line or a curve drawn by \texttt{\tzto}, just specify the optional arguments \texttt{\{<text>\}} and \texttt{[<node opt>]} \textit{in-between} the two coordinates. By default, the \texttt{[<node opt>]} is \texttt{[above,midway]}.

\begin{tikzpicture}
  \tzto{[bend right=15]}{(0,3)}{(4,0)}{(demand)}
  \tzto[dashed,bend left]{(1,1)}{(route B)(4,1)}
  \tzto[blue,bend right]{(0,3)}{[plan C] [below,near end] (3,3)}
\end{tikzpicture}

\begin{tikzpicture}
  \tzbezier{[bend right=15]}{(0,3)}{(4,0)}{(demand)}
  \tzbezier{[bend right=10]}{(0,0)}{(3,3)}{(supply)}[r]
\end{tikzpicture}

\subsection{\texttt{\tzbezier}}

\texttt{\tzbezier} accepts three or four coordinates as arguments. You can add text to the curve drawn by \texttt{\tzbezier} using the optional arguments \texttt{\{<text>\}} and \texttt{[<node opt>]} after the last coordinate.
\begin{tikzpicture}
\tzhelplines(4,3)
\tzbezier(0,2)(1,3)(3,0)(4,1){curve}[r]
\tzbezier[red,yshift=-5mm](0,2)(1,3)(3,0)(4,1){curve}[midway]
\tzbezier[green,text=blue,yshift=-10mm](0,2)(1,3)(3,0)(4,1){curve}[b,near end]
\end{tikzpicture}

2.7.4 \texttt{tzparabola}

\texttt{tzparabola} accepts two or three coordinate arguments. You can add text to a parabola drawn by \texttt{tzparabola} using the optional arguments \{<text>\} and \{<node opt>\} following the last coordinate. The text is placed at (by default) or around the last coordinate according to \{<node opt>\}.

\begin{tikzpicture}
\tzhelplines(4,3)
\tzparabola(0,2)(1,.5)(3,2){$AC$}[r]
\tzparabola[bend at end,blue](0,0)(4,3){$u(x)$}[r,red]
\end{tikzpicture}

3 An Intuitive Introduction II: Repetition of Coordinates

3.1 Linking many coordinates: Semicolon versions

3.1.1 \texttt{tzlines}: Connected line segments

\texttt{tzlines} connects with line segments an arbitrary number of coordinates. The coordinate iteration must end with semicolon ;. Here, the semicolon ‘;’ indicates the end of repetition of coordinates. Let us call this kind of macro a \textit{semicolon version} macro.

\begin{tikzpicture}
\tzhelplines(4,3)
\tzlines(0,0)(1,2)(3,0)(4,1)(2,2)(3,3)(0,3)(0,1)(4,2);
\end{tikzpicture}

With the optional argument \{<text>\} followed by \{<node opt>\} \textit{in-between two coordinates}, you can print \{<text>\} at or around the middle point of the corresponding line segment in accordance with \{<node opt>\} (by default [midway]).

\begin{tikzpicture}
\tzhelplines(4,3)
\tzlines[dashed,->](0,0){up}[1]
(1,2){down}[r]
(3,0){up}[r]
(4,1);
\end{tikzpicture}
The optional argument \{<text>\} following the last coordinate can be used as a name of the whole connected line segments. The \{<text>\} is placed at (by default) or around the last coordinate according to [\langle node opt\rangle].

\begin{tikzpicture}[sloped,auto]
\zhelplines(4,3)
\tzlines[dashed,-\rangle](0,0){\up}
\(1,2\){\down}
\(3,0\){\up}
\(4,1\){\Weight}[r,blue]
\end{tikzpicture}

Remember the repeating pattern is the triple (\langle coor\rangle){\langle text\rangle}[\langle node opt\rangle] in that order. DO NOT FORGET to indicate when the repetition ends by typing a semicolon. So it will look like (){}[]..repeated..(){}[];

3.1.2 \tzpolygon, \tzpolygon*: Closed paths

\tzpolygon draws closed line segments. \tzpolygon is also one of semicolon versions, meaning that it has to end with a semicolon. In fact, \tzpolygon is a closed version of \tzlines.

The starred version \tzpolygon* does the same thing as \tzpolygon except for one thing. \tzpolygon*, by default, fills the interior of the polygon with black!50 with fill opacity=.3 but with text opacity=1. (Changing the fill opacity is not an issue in this introduction. See Section 15.3 on page 88 for more details.)

\begin{tikzpicture}
\zhelplines(4,3)
\tzpolygon[green,thick](1,3)(0,2)(2,0)(3,1)(4,2)(2,3);
\end{tikzpicture}

The optional arguments \{<text>\} and [\langle node opt\rangle] in-between two coordinates prints <text> according to [\langle node opt\rangle] (by default midway) around the middle point of the corresponding line segment.

\begin{tikzpicture}
\zhelplines(4,3)
\tzpolygon*[green,thick](1,3)(0,2)(2,0)(3,1)(4,2)(2,3);
\end{tikzpicture}

The options \{<text>\} and [\langle node opt\rangle] following the last coordinate can be used as a name of the connected line segments.

The entire repetition will look like (\langle coor\rangle){\langle text\rangle}[\langle node opt\rangle] ..repeated.. (){}[];

DO NOT FORGET to indicate when the repetition ends by typing a semicolon.

\begin{tikzpicture}[sloped,auto]
\zhelplines(4,3)
\tzpolygon[dashed](0,0){\up}(1,2){\down}(3,0){\up}(4,1){\Weight}[r]
\end{tikzpicture}
3.1.3 \texttt{\textbackslash tzpath\textdaggerdbl: Filling area}

\texttt{\textbackslash tzpath} accepts an arbitrary number of coordinates to form a path, like \texttt{\textbackslash tzlines} does, but the path is invisible. This is a \textit{semicolon version} macro, so the coordinate iteration must be ended by a semicolon \texttt{;}. With [\texttt{draw}] option you can visualize the invisible path.

\begin{tikzpicture}
\helplines(4,3)
\tzpath[\texttt{draw}](1,3)(0,2)(2,0)(3,1)(4,2)(2,3);
\end{tikzpicture}

You can fill the interior of a path formed by \texttt{\textbackslash tzpath} (after being closed) with color or pattern, in usual Ti\textit{k}Z way.

\begin{tikzpicture}
\helplines(4,3)
\tzline[\texttt{blue}](0,3)(4,0)
\tzpath[\texttt{pattern=crosshatch}]
(1,3)(0,2)(2,0)(3,1)(4,2)(2,3);
\tzpath[\texttt{pattern=bricks},\texttt{preaction={fill=brown}}]
(0,0) (1,2) (3,0) (4,1);
\end{tikzpicture}

The starred version \texttt{\textbackslash tzpath\textdaggerdbl} takes the default options \texttt{\textbackslash fill=black\textexclamstyle{50}}, \texttt{\textbackslash fill opacity=.3} to fill the area.

\begin{tikzpicture}
\helplines(4,3)
\tzline[\texttt{blue}](0,3)(4,0)
\tzpath*[\texttt{green}]
(1,3)(0,2)(2,0)(3,1)(4,2)(2,3);
\tzpath*[\texttt{fill}]
(0,0) (1,2) (3,0) (4,1);
\end{tikzpicture}

How to change the \texttt{\textbackslash fill opacity} with \texttt{\textbackslash tzpath\textdaggerdbl} is not discussed in this introduction, but one example is given below. (See Section 13.2 on page 80 for more details.)

\begin{tikzpicture}
\helplines(4,3)
\tzline[\texttt{blue}](0,3)(4,0)
\tzpath*[\texttt{green}]
(1,3)(0,2)(2,0)(3,1)(4,2)(2,3);
\tzpath*[\texttt{fill}]
(0,0) (1,2) (3,0) (4,1);\texttt{\{}1\texttt{\}}
\end{tikzpicture}
3.2 Many dots: Semicolon versions

3.2.1 \texttt{\textbackslash tzcdots(*)}

\texttt{\textbackslash tzcdots} accepts an arbitrary number of coordinates to print circle dots, but the coordinate repetition must be ended by ; (semicolon version). \texttt{\textbackslash tzcdots*} prints filled circle dots.

\begin{tikzpicture}
\blackelinestripes{(4,3)}
\tzcdots(0,0)(1,1)(2,1)(3,2)(4,3);
\tzcdots*[fill=red](0,3)(1,3)(2,3)(3,3)(4,2);
\end{tikzpicture}

Each coordinate can be labeled by specifying the optional argument \{\texttt{\textbackslash label}\} followed by \texttt{\textbackslash angle}. You can also change the size (\textit{radius}) of the dots by specifying the last parenthesis argument \{\texttt{\textbackslash dot radius}\} after the semicolon.

\begin{tikzpicture}
\blackelinestripes{(4,3)}
\tzcdots(0,0)(1,1){\texttt{B}}(2,1){\texttt{D}}[\texttt{-90}](4,3);(6pt)
\tzcdots*[fill=red](0,3){\texttt{A}}(1,3){\texttt{B}}(2,3){\texttt{C}}[\texttt{-90}](3,3)
(4,2){\texttt{E}}[[\texttt{blue}]0];(3pt) \% radius
\end{tikzpicture}

3.2.2 \texttt{\textbackslash tzdots(*)}

\texttt{\textbackslash tzdots} accepts an arbitrary number of coordinates to print circle node dots, but the repetition must be ended by ; (semicolon version). \texttt{\textbackslash tzdots*} prints filled circle node dots.

\begin{tikzpicture}
\blackelinestripes{(4,3)}
\tzdots(0,0)(1,1){\texttt{B}}(2,1){\texttt{D}}[\texttt{-90}](4,3);
\tzdots*[fill=red](0,3)(1,3)(2,3)(3,3)(4,2);
\end{tikzpicture}

Each coordinate can be labeled by specifying the optional argument \{\texttt{\textbackslash label}\} followed by \texttt{\textbackslash angle}. You can also change the size (\textit{diameter}) of the dots by specifying the last parenthesis argument \{\texttt{\textbackslash dot size}\} after the semicolon.

\begin{tikzpicture}
\blackelinestripes{(4,3)}
\tzdots(0,0)(1,1){\texttt{B}}(2,1){\texttt{D}}[\texttt{-90}](4,3);(10pt)
\tzdots*[fill=red](0,3)(1,3)(2,3)(3,3)(4,2);
\end{tikzpicture}
3.3 Many coordinates: Semicolon versions

3.3.1 \tzcoors, \tzcoors*

\tzcoor accepts a pair of mandatory arguments in parentheses: \((\text{<coor>})(\text{<name>})\). \tzcoors accepts an arbitrary number of the pairs to define multiple coordinates. It is a semicolon version, meaning that a semicolon is necessary to indicate when the coordinate repetition ends.

\begin{tikzpicture}
\zhelplines(4,3)
\tzcoors(0,0)(A)
(1,1)(B)
(2,1)(C)
(3,3)(D)
(4,2)(E);
\tzlines(A)(B)(C)(D)(E);
\end{tikzpicture}

The options \{\text{<label>}\} and [\text{<angle>}] following each pair of \((\text{<coor>})(\text{<name>})\) allow you to put \text{<label>} in the direction of \text{<angle>} around the coordinate. Here, the repeating pattern is the quadruple \((\text{<coor>})(\text{<name>})(\text{<label>})[\text{<angle>}]\). The first two parenthesis arguments are mandatory and others are optional. The pattern is repeated until \tzcoors meets a semicolon ;.

\begin{tikzpicture}
\zhelplines(4,3)
\tzcoors(0,0)(A){A}
(1,1){B}
(2,1){C}{C}[0]
(3,3){D}{D}[180]
(4,2){E}{E}[-90];
\tzlines(A)(B)(C)(D)(E);
\end{tikzpicture}

The starred version \tzcoors* does one more thing: prints node dots.

\begin{tikzpicture}
\zhelplines(4,3)
\tzcoors*(0,0)(A){A}
(1,1){B}
(2,1){C}{C}[0]
(3,3){D}{D}[180]
(4,2){E}{E}[-90];
\end{tikzpicture}

3.3.2 \tzcoorsquick

\tzcoorsquick is just to see the array of many coordinates at a glance. \tzcoorsquick works like \tzcoors, but it automatically prints the name of each coordinate as its label, right at the point, by default.
3.3.3 \tzcoorsquick*

The starred version \tzcoorsquick* prints node dots and automatically puts the labels above (in the direction of 90 degree from) them, by default.

3.4 plot coordinates: Semicolon versions

3.4.1 \tzplot*: Mark dots with [mark=*]

\tzplot* accepts an arbitrary number of coordinates to print bullets with the radius (\texttt{mark size} in Ti\textit{k}Z) of 2pt, which is the initial value in Ti\textit{k}Z. The repetition of coordinates must be ended by ; (semicolon version).

Each coordinate can be labeled by specifying the optional argument \{<text>\} followed by [\texttt{angle}]. With the option \texttt{mark=o} you can print hollow dots. You can also change the radius of the marks by specifying the \texttt{last} parenthesis argument (\texttt{mark size}) after the semicolon.
3.4.2 \texttt{\texttt{tzplot}}: Lines with \texttt{[tension=0]}\

\texttt{\texttt{tzplot}} accepts an arbitrary number of coordinates and draws line segments connecting them. The repetition of coordinates must be ended by ; (semicolon version).

\begin{tikzpicture}
\zhelplines(4,3)
\tzplot\[mark=o\](0,0){A}
(1,1){Best}
(2,1){Case}
(3,3){Done}
(4,2){END!}[\[blue\]0](1.2pt);
\end{tikzpicture}

3.4.3 \texttt{\texttt{tzplot*}}[draw]: Lines with dots\

\texttt{\texttt{tzplot*}}[draw] prints bullets and draws line segments connecting specified coordinates. The repetition of coordinates must be ended by ; (semicolon version).

\begin{tikzpicture}
\zhelplines(4,3)
\tzplot*[draw](0,0)
(1,1)
(2,1)
(3,3)
(4,2){END!}[\[blue\]0];
\end{tikzpicture}

3.4.4 \texttt{\texttt{tzplotcurve}}: Curves with \texttt{[smooth,tension=1]}\

\texttt{\texttt{tzplotcurve}} plots coordinates with the option \texttt{[smooth,tension=1]}, resulting in a curve connecting an arbitrary number of coordinates. The repetition of coordinates must be ended by ; (semicolon version).

\begin{tikzpicture}
\zhelplines(4,3)
\tzplotcurve(0,0)(1,1)(2,1);
\draw plot coordinates {(0,0)(1,1)(2,1)};
\end{tikzpicture}
% syntax: simplified
\tzplotcurve[<opt>]{<tension>}"<path name>"
 (\coor)(\label)[\angle]..repeated..(\coor)(\label)[\angle];
% defaults
 [smooth,tension=1]{1}"<m>{}..repeated..(){}[];
% <m> means mandatory

\tzplotcurve(0,0)(1,1)(2,1); % works like:
\draw [smooth,tension=1] plot coordinates {(0,0)(1,1)(2,1)};

You can change the tension value by specifying the optional argument {<tension>}, before the first coordinate.

\begin{tikzpicture}
\tzcoor(.5,.5)(A)
\tzplotcurve[blue]{2}(1,3)(A)(4,0);
\tzplotcurve[thick]{1}(1,3)(A)(4,0); % default: tension=1
\tzplotcurve[red]{1.5}(1,3)(A)(4,0);
\tzplotcurve[dashed]{0}(1,3)(A)(4,0);
\end{tikzpicture}

4 An Intuitive Introduction III: Plotting Functions

4.1 Axes

4.1.1 \tzaxes

% syntax: simplified
\tzaxes[<opt>]<-x-shift,y-shift>(<x1,y1>)(<x2,y2>)
 {<x-text>}{<x-opt>}{<y-text>}{<y-opt>}
% defaults
 [-]0,0,(0,0){<m>}{right}{above}
% <m> means mandatory

\tzaxes draws the x-axis from <x1> to <x2> and the y-axis from <y1> to <y2>.
If \((<x_1,y_1>)\) is omitted, it is regarded as \((0,0)\). And optionally the names of x-axis and y-axis can be printed at a specified place (by default, \([\text{right}]\) for x-axis and \([\text{above}]\) for y-axis).

With the optional argument \(<x\text{-shift},y\text{-shift}>\), the axes are shifted accordingly. Two axes intersect at \((<x\text{-shift},y\text{-shift}>),\) by default \((0,0)\).

4.1.2 \texttt{\textbackslash tzaxes*}

The starred version \texttt{\textbackslash tzaxes*} is just to set the current state to a bounding box when the \texttt{\textbackslash tzaxes} macro execution is completed. Use \texttt{\textbackslash tzaxes*} before any larger graphics.

4.1.3 \texttt{\textbackslash tzshoworigin, \textbackslash tzshoworigin*}

\texttt{\textbackslash tzshoworigin} prints ‘0’ (roughly) at the bottom left of the origin.

\texttt{\textbackslash tzshoworigin*} prints a node dot with the size of 2.4\text{pt} (by default) at the origin.
\begin{tikzpicture}[scale=.5]
\tzhelplines(-1,-1)(8,5)
\tzshoworigin*
\tzaxes(-1,-1)(8,5)
\end{tikzpicture}

\tzshoworigin*\textcolor{blue}{(7,4)}\{O_2\}
\tzaxes\{blue\}<7,4>(8,5)(-1,-1)
\tzshoworigin\textcolor{blue}{(7,4)}\{O_1\}[ar]

Notice that, in the above example, the two axes intersect at (7,4) by the shift option <7,4>.

4.1.4 \texttt{tzaxisx, tzaxisy}

\texttt{tzaxisx} and \texttt{tzaxisy} draw the x-axis and the y-axis, respectively.

\begin{tikzpicture}[scale=.5]
\tzhelplines(-1,-1)(8,5)
\tzshoworigin\textcolor{blue}{(8,5)}
\tzaxes(-1,-1)(8,5)
\tzaxes\textcolor{blue}{[blue]}<7,4>(8,5)(-1,-1)
\tzshoworigin\{7,4\}\{O_2\}[ar]
\end{tikzpicture}

\begin{tikzpicture}[scale=.5]
\tzhelplines(-1,-1)(8,5)
\tzshoworigin\textcolor{blue}{(8,5)}
\tzaxes\textcolor{blue}{(-1,-1)}<7,4>(8,5)(-1,-1)
\tzshoworigin\{7,4\}\{O_1\}[ar]
\end{tikzpicture}

\begin{tikzpicture}[scale=.5]
\tzhelplines(-1,-1)(8,5)
\tzshoworigin\textcolor{blue}{(-1,1)}(8,5)
\tzaxisx\textcolor{blue}{(8,5)}\{-1\}\{x\}
\tzaxisy\textcolor{blue}{[-1]}\{y\}
\tzaxisy\textcolor{blue}{[dashed,->]}<2><-1>\{-1\}\{y\}
\end{tikzpicture}

4.2 Ticks

4.2.1 \texttt{tzticks}

\texttt{tzticks\{x-tick places\}\{y-tick places\}} prints tick labels for x- and y-axis at specified places, which are comma separated. By default, tick labels are the numbers specified.
You can change the numbered labels, for example \{2,4,7\}, to any other form, by doing like, for example, \{2/mylabel,4,7\}. (Internally, `\tzticks` uses the `foreach` operation of Ti\kZ.)

\begin{tikzpicture}[scale=.5,font=\scriptsize]
\tzhelplines(8,5)
\tzshoworigin
\tzaxes(-1,-1)(8,5)
\tzticks\{1,2,\ldots,8\} \% x-ticks
\{1,2,\ldots,5\} \% y-ticks
\end{tikzpicture}

4.2.2 `\tzticks*`

The starred version `\tzticks*` prints tick marks from 0pt to 3pt by default, without printing tick labels.

\begin{tikzpicture}[scale=.5,font=\scriptsize]
\tzhelplines(8,5)
\tzshoworigin
\tzaxes(-1,-1)(8,5)
\tzticks*[1,2,\ldots,7]
\{1,2,3,4\}
\end{tikzpicture}

You can change the length of tick marks, for example, like (-2pt:3pt) and (-5pt:10pt) as shown in the following example.

\begin{tikzpicture}[scale=.5]
\tzhelplines(-1,-1)(8,5)
\tzshoworigin
\tzaxes(-1,-1)(8,5)
\tzticks*(-2pt:3pt){1,2,\ldots,7}
\{1,2,3,4\}
\end{tikzpicture}

4.2.3 `\tzticksx(*)`, `\tzticksy(*)`

`\tzticksx` and `\tzticksy` prints x-tick labels and y-tick labels, respectively.

\begin{tikzpicture}[scale=.5,font=\scriptsize]
\tzhelplines(8,5)
\tzshoworigin
\tzaxes(-1,-1)(8,5)
\tzticksx(1/x,3/y,5/z,7)
\tzticksy(-5pt:10pt){1,2,4/k}
\end{tikzpicture}
\zticksx and \zticksy suppress tick labels for their corresponding axes, like \tzticks.}

\begin{tikzpicture}[scale=.5,font=\scriptsize]
%\zhelplines(8,5)
\tzaxes(-1,-1)(8,5)
\zticksx*(1/x,3/y,5/z,7)
\zticksy*(-Spt:10p){1,2,4/k}
\end{tikzpicture}

4.3 Projections on the axes

4.3.1 \tzprojx(*), \tzprojy(*)

\tzproj draws a dotted (by default) line from a specified coordinate to its projection point on the x-axis and prints text around ([below] by default) the projection point. \tzprojy works similarly but for the projection point on the y-axis.

\begin{tikzpicture}
%\zhelplines(4,3)
\tzaxes(4,3)
\tznode(3,2){\$(x_1,x_2)\$}[ar]
\tzprojx*(3,2){\$x_1\$}[xshift=-3pt]
\tzprojy[->,dashed,draw=red](3,2){\$x_2\$}
\tzprojx*(1.5,2.5){\$a\$}
\tzprojy*(2.5,1.5){\$b\$}
\end{tikzpicture}

\tzprojx* does one more thing. It prints a node dot at a specified coordinate. \tzprojy* works similarly but for the projection point on the y-axis.

\begin{tikzpicture}
%\zhelplines(4,3)
\tzaxes(4,3)
\tznode(3,2){\$(x_1,x_2)\$}[ar]
\tzprojx*(3,2){\$x_1\$}[xshift=-3pt]
\tzprojy[->,dashed,draw=red](3,2){\$x_2\$}
\tzprojx*(1.5,2.5){\$a\$}
\tzprojy*(2.5,1.5){\$b\$}
\end{tikzpicture}

4.3.2 \tzproj(*)

\tzproj combines \tzprojx and \tzprojy. And \tzproj* combines \tzprojx* and \tzprojy*, so it suppresses tick labels.

\begin{tikzpicture}
%\zhelplines(4,3)
\tzaxes(4,3)
\tznode(3,2){\$(x_1,x_2)\$}[ar]
\tzproj[<->,solid,draw=red](3,2){\$x_1\$}[xshift=-3pt]
\tzproj*(1.5,2.5){\$a\$}[xshift=3pt,\text{\textcolor{blue}{text=blue}}]{\$b\$}
\end{tikzpicture}
4.4 Plot functions

4.4.1 \texttt{tznfn}

\texttt{tznfn\{<fn of \ x>\}[<a:b>]\} plots a function of \emph{x} over the specified domain \([a : b]\), which means that \(a \leq x \leq b\). Optionally, you can add \{<text>\} with \([<node opt>]\) as shown in the following example.

\begin{verbatim}
\begin{tikzpicture}[scale=.7]
\tzhelplines(5,5)
\tzaxes(5,5)
\tznfn((x-1)^2)[0:3] % y=(x-1)^2
\def\Gx{(x-2)^2+1}
\tznfn[dashed]{\Gx}[1:4] % g(x)=(x-2)^2+1
\tznfn[red]{\sin(x)+3}[0:2*pi] % y=sin x
\end{tikzpicture}
\end{verbatim}

You can name a path formed by \texttt{tznfn} by specifying the optional argument "<path name>" right before the mandatory curly brace argument \{<fn of \ x>\}. The name of a path is used to find intersection points.

\begin{verbatim}
% syntax: simplified
\tznfn[<opt>]"<path name>"\{<fn of \ x>\}[<a:b>]\{<text>\}[<node dot>]
% defaults
[]""\{<m>\}[<m>]{\}[]
% <m>: mandatory
\begin{tikzpicture}[scale=.7,font=\footnotesize]
\tzhelplines(5,5)
\tzaxes(5,5){$x$}{$y$}
\def\bgt{4-x}
\def\IC{3/x}
\def\ICa{4/x}
\tznfn"bgt"\{\bgt\}[0:4] % name path = bgt
\tznfn[blue]"IC"\{\IC\}[.75:4] % name path = IC
\tznfn[red]\{\ICa\}[1:4] % name path = ICa (automatically)
\end{tikzpicture}
\end{verbatim}

\textbf{Remark:} If the curly brace mandatory argument consists of \emph{only a macro name} like \{\texttt{Foo}\}, the macro name 	exttt{Foo} (without the backslash) is automatically assigned to the \textit{name of the path}. (See Section 20.1.3 on page 111, for more details.)

4.4.2 \texttt{tzhfnat, tzhfn: Horizontal lines}

\texttt{tzhfnat} draws a horizontal line (the graph of a constant function) at \(y\), by default, from left to right of the current bounding box.

\begin{verbatim}
% syntax
\tzhfnat[<opt>]"<path name>"\{<y-val>\}[<from:to>]\{<text>\}[<node opt>]
% defaults
[]""\{<m>\}[west:east (of current bounding box)]\{\}[]
\end{verbatim}

\texttt{tzhfn(<coor>)} accepts a coordinate, instead of the value of \(y\), to draw a horizontal line at the value of \(y\) coordinate of \(<\texttt{coor}>\), ignoring the value of \(x\) coordinate.
\begin{tikzpicture}
\tzhelplines(4,3)
\tzhfnat(0)
\tzhfnat[dashed]{1}[1:4]{Two}[1,at start]
\tzcoors(0,2)(A)(0,3)(B);
\tzhfn[blue]{A}[0:3]{Three}[b,near end]
\tzhfn->](B)[0:3]{Four}[r]
\end{tikzpicture}

4.4.3 \texttt{tzvfnat, tzvfn}: Vertical lines
\texttt{tzvfnat} draws a vertical line at \(x\) from bottom to top of the current bounding box by default.

\begin{verbatim}
\% syntax: simplified
\tzvfnat<opt>"<path name>"{<x-value>}[<from:to>][<text>][<pos>]
\%
\% defaults
[]""{<m>}[south:north (of current bounding box)]{}[]
\end{verbatim}

\texttt{tzvfn(<coor>)} accepts a coordinate, instead of the value of \(x\). It draws a vertical line at the value of \(x\) coordinate of \(<\text{coor}>\), ignoring the value of \(y\) coordinate.

\begin{tikzpicture}[scale=.7,font=\footnotesize]
\tzhelplines(4,3)
\tzvfnat(0)
\tzvfnat[dashed]{1}[1:3]{Two}[b, at start]
\tzvfn[blue]{2,0}[0:3]{Three}[a, near end, sloped]
\tzvfn->](3,0)[0:3]{Four}[r]
\end{tikzpicture}

4.4.4 \texttt{tzLFn}: Linear functions
\texttt{tzLFn(<coor1>)(<coor2>)...} draws a linear function passing through two points: \(<\text{coor1}>\) and \(<\text{coor2}>\). \texttt{tzLFn(<coor1>){<slope>}...} draws a linear function passing through one point, \(<\text{coor1}>\), with the slope of \(<\text{slope}>\). If both of the two coordinates are specified the option \(<\text{slope}>\) is ignored. The domain in the form of \([a:b]\) is also a required argument.

\begin{tikzpicture}
\tzhelplines(4,3)
\tzhfnat(0)
\tzvfnat[dashed]{1}[1:4]{Two}[1,at start]
\tzcoors(0,2)(A)(0,3)(B);
\tzhfn[blue]{A}[0:3]{Three}[b,near end]
\tzhfn->](B)[0:3]{Four}[r]
\end{tikzpicture}

4.5 Intersection points
4.5.1 Naming paths
In Ti\textit{k}Z, you can find intersection points when two named paths intersect. The name of a path is usually given by the option \{name path=<path name>\} in Ti\textit{k}Z.
With the package `tzplot`, you can name a path by specifying an optional argument within quotation marks such as "<path name>". (Of course, you can also name a path in usual TikZ way, like `[name path=<path name>]`.)

In this package, all macros (with a few exceptions) related to lines and curves accept this quote optional argument to name paths as follows:

```
tzline[<opt>]"<path name>"(<coor>)...
tzlines..."<path name>"(<coor>)...
tzpolygon..."<path name>"(<coor>)...
tzpath..."<path name>"(<coor>)...
tzto..."<path name>"(<coor>)...
tzparabola..."<path name>"(<coor>)...
tzbezier..."<path name>"(<coor>)...
tzfn..."<path name>"{<fn of \x}>...
tzLFn..."<path name>"(<coor>)...and more...
```

```
tzline[dashed]"foo"(1,1)(3,3) % works like\draw [dashed,name path=foo](1,1) -- (3,3);
```

In most cases, the quote optional arguments for naming paths are placed immediately before the first mandatory argument of the `tzplot` macros.

### 4.5.2 `tzXpoint(*)`: Intersection points of two paths

For example, `tzXpoint{path1}{path2}(A)` finds intersection points of `path1` and `path2` and names the first intersection point (A). This intersection point can be referred to as (A) or (A-1). If there are two or more intersection points found, they are called (A)=(A-1), (A-2), (A-3), and so on.

```
\begin{tikzpicture}
  \tzhelplines(4,3)
  \tzline"AA"(0,0)(3,2)
  \tzline"BB"(0,3)(3,0)
  \tzXpoint{AA}{BB}(X)
  \tzdot*(X){E}
\end{tikzpicture}
```

```
\tzXpoint* prints a node dot at the first intersection point. You can label the point by specifying `{<text>}` and `[<angle>]` after the specified intersection name.

```
\begin{tikzpicture}[scale=.7,font=\footnotesize]
  \tzhelplines(5,5)
  \tzaxes(5,5){$x$}{$y$}
  \def\bgt{4-\x}
  \def\IC{3/\x}
  \tzfn"bgt"{(\bgt)[0:4]} % name path = bgt
  \tzfn[blue]"IC"{(\IC)[.75:4]} % name path = IC
  \tzXpoint*(bgt){IC}(E){$A$}[45] % first intersection
  \tzdot(E-2){$B$}[45]{5pt} % second intersection
\end{tikzpicture}
```

**Remark:** You have to expect TikZ to take a few seconds (or less) to find intersection points.
4.5.3 \texttt{tzhXpointat}(\star), \texttt{tzvXpoint}(\star): Vertical intersection points

To find vertical intersection points at \( x \) to a curve, you should specify a path name and either the value of \( x \) or the coordinate \((x, y)\). Here the \( y \) coordinate is ignored.

\texttt{tzhXpointat}(\texttt{<path>}){\{\texttt{<x>}\}}(A) finds vertical intersection points of \texttt{<path>} at \( x = \texttt{<x>} \) and names it (A). The starred version \texttt{tzhXpointat*} additionally prints a node dot at the (first) intersection point.

The macro \texttt{tzvXpoint} uses \texttt{(<coor>)}, while \texttt{tzvXpointat} uses the value of \( x \). Here the \( y \) coordinate of \texttt{(<coor>)} is not important. \texttt{tzvXpoint*} additionally prints a node dot at the (first) intersection point.

\begin{verbatim}
\begin{tikzpicture}[scale=.7,font=\scriptsize]
\tzhelplines(5,5)
\tzaxes(5,5)
def Fx((x-1)^2)
tzfn Fx[0:3] % name path = Fx (automatically)
tzhXpointat(Fx){2.5}(A)
tzhXpointat*(Fx)(2.8,1)(B) % y=1 is ignored
tzproj*(->](A)\{$2.5\}\{f(2.5)\}$
tzdot(B){BB}[0]
\end{tikzpicture}
\end{verbatim}

4.5.4 \texttt{tzhXpointat}(\star), \texttt{tzhXpoint}(\star): Horizontal intersection points

\texttt{tzhXpointat}(\texttt{<path>}){\{\texttt{<y>}\}}(A) works like \texttt{tzvXpointat}(\texttt{<path>}){\{\texttt{<x>}\}}(A), but it uses the value of \( y \) instead of \( x \). The starred version \texttt{tzhXpointat*} additionally prints a node dot at the (first) intersection point.

\texttt{tzvXpoint} uses \texttt{(<coor>)}, while \texttt{tzhXpointat} uses the value of \( y \). Here the \( x \) coordinate of \texttt{(<coor>)} is ignored. The starred version \texttt{tzhXpoint*} prints a node dot at the (first) intersection point.

\begin{verbatim}
\begin{tikzpicture}[scale=.7,font=\scriptsize]
\tzhelplines(5,5)
\tzaxes(5,5)
def Fx((x-1)^2)
tzfn Fx[0:3] % name path = Fx (automatically)
tzhXpointat(Fx){1.2.8}(B) % x=1 is ignored
\tzhXpointat*(Fx){1.2.8}(B) % y=1 is ignored
\tzproj*[-](A)\{$f^{-1}(2.5)\}\{y=2.5\}$
tzdot(B){BB}[0]
\end{tikzpicture}
\end{verbatim}

4.6 Tangent lines

4.6.1 \texttt{tztangentat}

\texttt{tztangentat}(\texttt{<path>}){\{\texttt{<x>}\}[\texttt{<a:b>}] draws a tangent line to \texttt{<path>} at \( x = \texttt{<x>} \) over \( x \in [a, b] \). The domain is a mandatory argument and should be of the form \([\texttt{<from:to>}]\).

% syntax: simplified
\tztangentat{\texttt{<path>}}{\{\texttt{<x>}\}[\texttt{<domain>]}{\{\texttt{<text>}\}{\{\texttt{<node opt>}}]
% defaults
[]{<m>}{<m>}{<m>}{<m>}{<m>}}
% <m> means mandatory
Remark: The slope of a tangent line drawn by \texttt{tztangentat} is just approximate.

The line is drawn on the \texttt{behind} layer, by default.

You can also add some text next to the line by specifying the optional arguments \{<text>\} and \[<node opt>\], after the domain.

\begin{tikzpicture}[scale=.7]
\helplines(5,5)
\parabola[thick,blue]"curve"(0,1)(2,3)(4,2)
\vXpoint{curve}{1}
\vXpoint{curve}{3}
\tangentat{curve}{1}{0:3}
\tangentat{curve}{3}{1:5}{tangent at $x=3$}[b]
\end{tikzpicture}

tangent at $x = 3$

4.6.2 \texttt{tztangent}

\texttt{tztangent} works like \texttt{tztangentat}, but it accepts a coordinate instead of the value of $x$.

\texttt{tztangent\{<path>\}\{<coor>\}} draws a tangent line to \texttt{<path>} at the $x$ coordinate of \texttt{<coor>}. Here, the $y$ coordinate of \texttt{<coor>} is ignored. The line is drawn on the \texttt{behind} layer, by default.

\begin{tikzpicture}[scale=.7]
\helplines(5,5)
\parabola[thick,blue]"curve"(0,1)(2,3)(5,1)
\vXpoint{curve}{1}{A}[135]
\vXpoint{curve}{3}{B}[45]
\tangent{curve}{A}{0:3}
\tangent[red]{curve}{B}{1:5}{tangent at $B$}[b]
\end{tikzpicture}
tangent at $B$

4.6.3 \texttt{tzsecantat, tzsecant}

\texttt{tzsecantat\{<path>\}\{<x1>\}\{<x2>\}} draws a secant line segment of \texttt{<path>} from $x_1 = <x1>$ to $x_2 = <x2>$ on the \texttt{behind} layer, by default.

\begin{tikzpicture}[scale=.7]
\helplines(5,5)
\parabola"curve"(0,1)(2,3)(5,1)
\secantat{curve}{1}{2}
\secantat[red]{curve}{1}{3}
\secantat[blue]{curve}{1}{4}
\end{tikzpicture}

\texttt{tzsecant} works like \texttt{tzsecantat}, but it accepts two coordinates instead of two values of $x$.

\texttt{tzsecant\{<path>\}\{<coor1>\}\{<coor2>\}} draws a secant line segment of \texttt{<path>} from the $x$ coordinate of \texttt{<coor1>} to the $x$ coordinate of \texttt{<coor2>}, ignoring $y$ values of the coordinates, on the \texttt{behind} layer by default.
5 Examples: Economics

5.1 Markets

5.1.1 Market equilibrium: step by step

**Step 1** Determine the size of the graph.

**Step 2** Draw the demand and supply curves. Here, we are using `tzto`.
5.1.2 Tax incidence: step by step

Step 1: Determine the size of the graph.
Step 2: Define functions and plot them. And then, find an intersection point.

Step 3: Draw the shifted supply curve and find new equilibrium point. And then, project the point on each axis.

Step 4: To illustrate the social welfare loss (SWL), find a vertical intersection point of the original supply curve using new equilibrium point.

Step 5: Project the old equilibrium point and the vertical intersection point onto the y axis and add text.

Step 6: Fill the area of the social welfare loss with color.

Step 7: Add text ‘SWL’ at the appropriate place.

5.2 Firms

5.2.1 Cost curves
5.2.2 Equilibrium of a competitive firm

\begin{tikzpicture}[scale=1,font=\scriptsize]
%\tzhelplines(10,10)
\tzaxes(10,10){q}{$P$\ AC\ MC}
\tzparabola"MC"(2,2)(6,9)\ MC(4)\ point\ on\ MC\ at\ q=4
\tzparabola"AC"(2,7)\ minAC
\tzXpoint*{\price}(MC)(A)(4)\ point\ (A)\ on\ MC\ at\ q=4
\tzparabola"AC"(2,7)(A)(8,9)\ minAC
\tzXpoint\ price\(\mc\)(E)(q=\pi^*)
\tzhXpoint*{\mc\}(MC)(4)
\tzprojx(E){q^*}{\mc}(A)(4)
\tzprojy(ACeqm)\ point\ on\ AC\ in\ eqm
\tzpath*\pi\(\mc\)(E)(A)(\mc\)(ACeqm)(ACeqm-|0,0);
\tznode(2,5)\ \pi^*
\end{tikzpicture}

5.2.3 Monopoly equilibrium

\begin{tikzpicture}[scale=1,font=\scriptsize]
%\tzhelplines(10,10)
\tzaxes(10,10){Q}{$P$\ D\ MR\ MC}
\def\DD{8-\x}
\def\MR{8-2*\x}
\def\MC{\x}
\tzfn\DD[0:8]{\mc\}(E)
\tzfn\MR[0:4.5]{\mc\}(E)
\tzfn\MC[0:7]{\mc\}(E)
\tzXpoint\ price\(\mc\)(E)(MC)(E)
\tzproj\ price\(\mc\)(E)(C)(SWL)
\tzplotcurve(3,9)(A)(12,3);\ trial\ and\ error
\tznode(1,6)\ C.S.
\tzpath[pattern=horizontal lines](0,0)(E)(E-|0,0);
\tznode(1,2)\ P.S.
\end{tikzpicture}

5.3 Consumers: Budget lines and indifference curves

\begin{tikzpicture}[scale=0.3,font=\scriptsize]
%\tzhelplines(15,12)
\tzaxes(15,12){x}{$y$}
\def\bgt{-3/4*\x+9}\ 3x+4y=36
\tzfn\bgt[0:12]{\mc\}(E)
\tzXpoint\ bgt(MC)(E)(E)
\tzproj\ bgt(0,8)(EE)(E-|0,0)
\tznode(1,6)\ C.S.
\tzpath[pattern=horizontal lines](0,0)(E)(E-|0,0);
\tznode(1,2)\ P.S.
\end{tikzpicture}
%\begin{tikzpicture}[scale=.15,font=\scriptsize]
%\tzhelplines(35,25)
%\tzaxes(35,25){\$x\$}{\$y\$}
%\def\bgt{-2/3*\x+20} \% 2x+3y=60
%\def\IC{150/\x} \% u(x,y)=xy
%\tzfn\bgt[0:30]
%\tzfn\IC[7:30]
%\tzcoor*(15,10)(E) \{E\}[45]
%\tzproj(E)
%\tzticks{15,30}{10,20}
%\tzvXpointat*[red]{\IC}{12}(A)
%\tztangent[blue,densely dashed]{\IC}(A)[5:20]
%\end{tikzpicture}

\section{Production Possibility Curves}

%\begin{tikzpicture}[scale=.5,font=\scriptsize]
%\tzhelplines(9,7)
%\tzaxes(9,7){\$x_1\$}{\$y_1\$}
%\tzaxes<8,6>(8,6)(-1,-1)
%\tzshoworigin\{O_1\}
%\tzshoworigin(8,6)\{O_2\}[ar]
%\tzto[bend right](2,5)(7,2){\$u_1\$}[r]
%\tzto[bend left](1,4)(6,1){\$u_2\$}[b]
%\end{tikzpicture}

\section{Edgeworth box}

%\begin{tikzpicture}[scale=.5,font=\scriptsize]
%\tzhelplines(9,7)
%\tzaxes(9,7){\$x_1\$}{\$y_1\$}
%\tzaxes<8,6>(8,6)(-1,-1)
%\tzshoworigin\{O_1\}
%\tzshoworigin(8,6)\{O_2\}[ar]
%\tzto[bend right](2,5)(7,2){\$u_1\$}[r]
%\tzto[bend left](1,4)(6,1){\$u_2\$}[b]
%\end{tikzpicture}
5.6 Growth

\begin{tikzpicture}[scale=.5,font={\scriptsize}]
\begin{scope}%\tikzstyle{every node}=[font=\scriptsize]
%\tikzstyle{every path}=[font=\scriptsize]
\begin{scope}%\tikzstyle{every node}=[font=\scriptsize]
%\tikzstyle{every path}=[font=\scriptsize]
\end{scope}%\tikzstyle{every path}=[font=\scriptsize]
\end{scope}%\tikzstyle{every node}=[font=\scriptsize]
\begin{scope}%\tikzstyle{every path}=[font=\scriptsize]
\end{scope}%\tikzstyle{every node}=[font=\scriptsize]
\end{tikzpicture}

5.7 Liquidity trap

\begin{tikzpicture}[scale=.5,font={\scriptsize}]
\begin{scope}%\tikzstyle{every node}=[font=\scriptsize]
%\tikzstyle{every path}=[font=\scriptsize]
\end{scope}%\tikzstyle{every path}=[font=\scriptsize]
\begin{scope}%\tikzstyle{every node}=[font=\scriptsize]
%\tikzstyle{every path}=[font=\scriptsize]
\end{scope}%\tikzstyle{every node}=[font=\scriptsize]
\begin{scope}%\tikzstyle{every path}=[font=\scriptsize]
\end{scope}%\tikzstyle{every node}=[font=\scriptsize]
\end{tikzpicture}

LM curves are drawn with two paths. To shift the IS curve, \texttt{<shift coor>} is used. See Section 20.2 on page 113 for more details.  

\begin{tikzpicture}[scale=.5,font={\scriptsize}]
\begin{scope}%\tikzstyle{every node}=[font=\scriptsize]
%\tikzstyle{every path}=[font=\scriptsize]
\end{scope}%\tikzstyle{every path}=[font=\scriptsize]
\begin{scope}%\tikzstyle{every node}=[font=\scriptsize]
%\tikzstyle{every path}=[font=\scriptsize]
\end{scope}%\tikzstyle{every node}=[font=\scriptsize]
\begin{scope}%\tikzstyle{every path}=[font=\scriptsize]
\end{scope}%\tikzstyle{every node}=[font=\scriptsize]
\end{tikzpicture}
The money demand ($m_d$) curve is drawn with one path. To do this, \texttt{code.append} is used. See Section 12.1 on page 74 for more details.

5.8 Miscellany

\begin{tikzpicture}[font=\scriptsize]
\def\z{1}
\tzcoors
(0,0)(A)
($(A) + (45:\z)$)(B)
($(B) + (45:\z)$)(C)
($(A) + (-45:\z)$)(D)
($(D) + (45:\z)$)(E)
($(E) + (45:\z)$)(F)
($(D) + (-45:\z)$)(G)
($(G) + (45:\z)$)(H)
($(H) + (45:\z)$)(I);
\foreach \a in {A,...,I}
{\tzdot*(\a)\a}
\tzlines[red](B)(E)(H);
\end{tikzpicture}

\begin{tikzpicture}[scale=1,font=\scriptsize]
\tzaxes(4,4){alternatives}{preferences}
\tztickx{1/#a,2/#b,3/#c}
\tzplot[mark=ball](1,3)(2,2)(3,1){A}[0];
\tzplot[mark=*] (1,2)(2,1)(3,3){B}[0];
\tzplot[mark=+] (1,1)(2,3)(3,2){C}[0];
\end{tikzpicture}

\begin{tikzpicture}[scale=1,font=\scriptsize]
\tzaxes(4,4){alternatives}{preferences}
\tztickx{1/#a,2/#b,3/#c}
\foreach \x in {1,2,3}
{\foreach \y in {1,2,3}
{\tzdot*(\x,\y)}}
\tzlines(1,3)(2,2)(3,1){A}[x];
\tzlines(1,2)(2,1)(3,3){B}[x];
\tzlines(1,1)(2,3)(3,2){C}[x];
\end{tikzpicture}
Part III
Points, Lines, and Curves

6 Getting Ready

6.1 Styles: tzdotted, tzdashed, tzhelplines

The styles tzdotted, tzdashed, and tzhelplines are defined as follows:

```latex
\% styles: tzdotted, tzdashed, tzhelplines
\tikzset{
  tzdotted/.style={
    line cap=round,dash pattern=on 0pt off 1cm/(#1),
    tzdotted/.default=10
  },
  tzdashed/.style={
    dashed=none,dash pattern=on 5mm/(#1) off 5mm/(#1),
    tzdashed/.default=10
  },
  tzhelplines/.style={help lines,-,tzdotted}
}
```

The styles tzdotted and tzdashed print 10 dots and 10 dashes per 1cm, respectively, by default. The style tzhelplines uses tzdotted by default.

6.2 \tzhelplines, \tzhelplines*

\tzhelplines draws grid from the first coordinate to the second coordinate. If only one coordinate is specified, then the first coordinate is regarded as (0,0).

The starred version \tzhelplines* uses the grid as a bounding box.

```latex
\% syntax: minimum
\tzhelplines(<coor>)
\% syntax: full
\tzhelplines[<opt>](<coor1>)(<coor2>)
\% defaults
  [help lines,tzdotted=10](<m>)(()
\% (<m>): mandatory argument
```

Here, (<m>) stands for a mandatory argument.

```latex
\tzhelplines(4,3) \% works similarly to:
  \draw [help lines] (0,0) grid (4,3);  
\tzhelplines(1,1)(4,3) \% works similarly to:
  \draw [help lines] (1,1) grid (4,3);
```

By default, \tzhelplines prints grid with 10 dots per 1cm. \tzhelplines with the option value [tzdotted=<n>] prints <n> dots per 1cm. (That is, the default value is tzdotted=10.)
% \tzhelplines
\begin{tikzpicture}
\tzhelplines(2,2)
\draw [help lines] (3,0) grid (5,2);
\end{tikzpicture}

% tzdotted: (default: 10 dots per 1cm)
\begin{tikzpicture}
\tzhelplines[thick](1,2) \% 10 dots
\tzhelplines[thick,tzdotted=20](2,0)(3,2) \% 20 dots
\tzhelplines[thick,tzdotted=5](4,0)(5,2) \% 5 dots
\end{tikzpicture}

With the option value, [tzdotted=<n>/<d>], \tzhelplines prints <n> dots per <d>cm. Similarly for tzdashed.

% tzdotted, tzdashed
\begin{tikzpicture}
\tzhelplines[thick,tzdashed](4,2)
\tzhelplines[thick,step=.5](4,2)
\end{tikzpicture}

% scaled: 7 dots per .7cm (10 dots per hard 1cm)
\begin{tikzpicture}[scale=.7]
\tzhelplines[tzdashed](4,3)
\tzhelplines[step=.5](4,3)
\end{tikzpicture}

% scaled: 10/.7 means 10 dots per .7cm
\begin{tikzpicture}[scale=.7]
\tzhelplines[tzdashed=10/.7](4,3) \% 10 dots
\tzhelplines[step=5,tzdotted=10/.7](4,3) \% 5 dots
\end{tikzpicture}

6.3 \tzbbox: A bounding box
\tzbbox sets a bounding box.

% syntax
\tzbbox(<coor1>)(<coor2>)
% defaults
(0,0)(<n>)

\tzbbox(-1,-1)(4,3) \% is an abbreviation of:
\path [ use as bounding box ] (-1,-1) rectangle (4,3);

If only one coordinate is specified, the first coordinate is regarded as (0,0).
7 Dots

7.1 \tzcdot(*): A small circle

A dot is usually expressed by a small circle. \tzcdot prints a circle dot \(*\). The starred version \tzcdot* prints a filled circle dot \(*\). The radius of the circle is 1.2pt, by default.

Here, \(<\textit{m}>\) stands for a mandatory argument. All others are optional arguments.

**How to change the size** There are three ways to change the radius of a circle dot drawn by \tzcdot.

1. The simplest way is to use the last parenthesis option, like \tzcdot(0,0)(3pt).

\[
\begin{tikzpicture}
\tzhelplines(4,2)
\tzcdot(0,0)\tzcdot(1,1)
\tzcdot*[tzcdot=3pt](2,1)(2pt)
\tzcdot*[tzcdot=3pt](3,0)(3pt)
\end{tikzpicture}
\]

2. You can use the key-value option \[\text{tzcdot=<dim>}\], like \tzcdot[tzcdot=3pt](0,0), to change the radius of a circle dot. The tzcdot key is defined in the package. If both the tzcdot key-value and the last parenthesis option are used, the former wins.

\[
\begin{tikzpicture}
\tzhelplines(4,2)
\tzcdot*[tzcdot=3pt](1,1) \tzcdot*(2,1)(2pt) \tzcdot*(3,0)(3pt)
\end{tikzpicture}
\]

% \tzcdot(*)
\begin{tikzpicture}
tzhelplines(4,2)
tzcdot(0,0) \tzcdot*[tzcdot=4pt](1,1)
tzcdot*[tzcdot=3pt](2,1)(2pt) \tzcdot*[tzcdot=3pt](3,0)(3pt)
\end{tikzpicture}
3. Another way to change the radius is to use a macro, like \texttt{\settzcdotradius{3pt}}. It is effective within the \texttt{tikzpicture} environment unless changed by \texttt{\settzcdotradius} again.

```
\settzcdotradius
\begin{tikzpicture}
\tzhelplines(4,2)
\settzcdotradius(4pt)
\tzcdot(0,0)
\tzcdot(1,1)
\tzcdot+(2,1)(2pt) \tzcdot*=3,0
\end{tikzpicture}
```

**How to label**  You can add a label to a specified coordinate by adding the optional argument \{<label>\} immediately after \(<\texttt{coor}>\). You can also change the <label> position by the option \[<\texttt{angle}>\]. (Note that you cannot use the abbreviations such as \texttt{a}, \texttt{r}, \texttt{bl}, etc. to place label nodes.)

```
\tzcdot(0,0){A} % is an abbreviation of:
\draw (0,0) circle (1.2pt) node [label={A}] {{}};
\tzcdot(0,0){(2pt)A}{[blue]0} % is an abbreviation of:
\draw (0,0) circle (2pt) node [label={[blue]0:A}] {{}};
```

The label position does not depend on the size of circle dots.

**How to change colors**  With the first optional argument \[<opt>\], you can change the color of a dot. You can also change the color of a label, as shown in the following example.

```
% \tzcdot(*)
\begin{tikzpicture}
\draw [help lines] (0,0) grid (4,3);
\tzcdot(0,0){A}
\tzcdot*[red](1,0){\textbf{B}}[45]
\tzcdot*[red,fill=green](2,1){green}[blue]0)(10pt)
\tzcdot*[tzcdot=2*7pt]{3,2}{big}[center]
\tzcdot*[fill=red, text=blue](4,3){D}(2*3pt)
\end{tikzpicture}
```

**Shift**  Dots can be shifted by specifying the optional argument \texttt{<shift coor>} immediately before \(<\texttt{coor}>\). The empty shift option \texttt{<>} is not allowed.
7.2 \tzcdots(*): Multiple circle dots

The macro \tzcdots takes an arbitrary number of coordinates as arguments to print multiple circle dots with the radius 1.2pt, by default. You need to indicate when the iteration of an arbitrary number of coordinates ends, by typing a semicolon ;. Let us call this kind of macro a semicolon version macro.

Remark:

- DO NOT FORGET to enter ‘;’ at the end of iteration.
  - Tokens such as ‘!’ and ‘#’ other than the open parenthesis ‘(’ work like the semicolon ‘;’ to indicate the end of iteration. But it is highly recommended to use ‘;’ for consistency.
- Without the semicolon ;, the macro does nothing or a compile error occurs.

The starred version \tzcdots* prints multiple filled dots.
How to label  Each coordinate can be followed by the optional arguments {<label>} and [<angle>] to label dots. So the repeating pattern is the triple (<coor>){<label>}[<angle>].

% \tzcdots: label
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcdots(0,0){A}
(1,1);
(2,1){C}[0]
(3,2){D}[-90]
(4,0){E};
\tzcdots*(0,3)(1,2){B}(2,2){C}[45](3,3){4,3};
\end{tikzpicture}

How to change the size of dots  There are three ways of changing the radius of dots.

1. The simplest way is to use the last parenthesis optional argument, after the semicolon.

2. Another way is to use the \tzcdot key, like \tzcdots[tzcdot=3pt].... If both options are used the key-value option wins.

3. You can also use the macro \settzcdotradius. The effect remains within the \tikzpicture environment unless it is changed again.

% \tzcdots: size (radius)
\begin{tikzpicture}
\tzhelplines(4,3)
\settzcdotradius{3pt}
\tzcdots*(0,0)(1,1){3pt} % simplest
\tzcdots*(1,1)(2,1){1pt};(1pt)
\tzcdots*[tzcdot=5pt](2,2)(3,2){5pt};
\tzcdots*(3,3)(4,3){3pt};
\end{tikzpicture}

How to change colors  With the first optional argument [<opt>], you can change the color of dots. You can also change the color of all labels at once using the first optional argument, like \tzcdots[text=red].... as shown in the following example.

% \tzcdots: color
\begin{tikzpicture}
\tzhelplines(4,3)
\settzcdotradius{3pt}
\tzcdots*[red]
(0,0)(1,1){\textbf{Ben}}[blue]-90(2,1)(3,0);
\tzcdots*[thick,blue,fill=green,text=red]
(1,2){A}(2,2){Ben}[blue]-90(3,2){C}(4,2){D};(4pt)
\tzcdots*[blue]
(1,3){A}(2,3){B}(3,3){C}(4,3){D};
\end{tikzpicture}

Shift  You can move the coordinates of dots by specifying <shift coor> option immediately before the first coordinate.
7.3 \tzdot(*): A single node dot

The macro \tzdot prints a small circle node *, as a dot, with the \textit{diameter} (or \textit{minimum size}) of 2.4pt, by default.

The starred version \tzdot* prints a filled dot *.

\tzdot(*) accepts one mandatory argument, denoted by \texttt{(<m>)}. All others are optional.

The style \tzdot is defined as follows:

\begin{verbatim}
\% syntax: minimum
\tzdot(coor)
\% syntax: medium
\tzdot*[<<coor>>[<label>][<angle>][<dot size>]
\% syntax: full
\tzdot*[<node opt>]<shift coor<<coor>>[<label>]<[<label opt]angle>][<dot size>]
\% defaults
*[ tzdot=2.4pt ]<>(<m>)\}[](2.4pt)
\% the style tzdot is predefined
% (<m>): mandatory
\end{verbatim}

\textbf{7.3.1 Three Ways to change the size of node dots}

There are \textbf{Three Ways} to change the \textit{diameter} (or \textit{minimum size}) of a node dot drawn by \tzdot(*).

1. Use the predefined style \texttt{tzdot} in the first optional argument, like \texttt{\tzdot[tzdot=5pt](0,0)}, which gives the same result as \texttt{\tzdot[minimum size=5pt](0,0)}.

\begin{verbatim}
\tzdot(0,0) % works like:
  \path (0,0) node [tzdot=2.4pt] {}; % default size
  % or equivalently
  \node [tzdot,minimum size=2.4pt] at (0,0) {};
\end{verbatim}

2. The simplest way is to use the \textit{last} parenthesis optional argument, like \texttt{\tzdot(0,0)(5pt)}, which yields the same result as in \texttt{\tzdot[tzdot=5pt](0,0)}. If both options are used, the \texttt{tzdot} (or \texttt{minimum size}) option overwrites the last parenthesis option.
3. To change the size of all node dots drawn by `\tzdot*`, you can use the macro `\settztodotsize`. Its effect remains valid until the end of `tikzpicture` environment unless changed again.

```
\setztodotsize{8pt}
\begin{tikzpicture}
\helplines(4,2)
\tzdot(0,0)
\tzdot*[4pt] \tzdot*[3,0](6pt)
\end{tikzpicture}
```

7.3.2 How to label

You can add a label to a specified coordinate by specifying the optional argument `{<label>}` immediately after the coordinate `{<coor>}`.

You can also change the label position by the option `{<angle>}` following `{<label>}`.

```
\tzdot(0,0){A}(3pt) % works like:
\path (0,0) node [tzdot=3pt,label={[A]}] {};
\tzdot*[0](A){[red]0}(3pt) % works like:
\path (0,0) node [fill,tzdot=3pt,label={[red]0:A}] {};
```

```
\begin{tikzpicture}
\helplines(4,3)
\tzdot(A)
\tzdot*[B][0](8pt)
\tzdot*[C][45](4pt)
\tzdot*[D][-90](6pt)
\tzdot*[E][-90](12pt)
\end{tikzpicture}
```

Unlike \tzdot, the \tzdot’s label position depends on the size of a circle node. In TikZ jargon, `{<label>}` is in a label node for a main node that is a circle node with no text in it, so `{<label>}` moves accordingly as the main node dot gets bigger or smaller.

7.3.3 How to change colors and shapes

With the first optional argument `{<node opt>}`, you can change of the color or shape of dots. You can also change the label color using `{<label opt>}` as shown in the following example.

Remark:
- `{<node opt>}` is for options of main nodes, `{<label opt>}` is for options of label nodes.
- `{<label opt>}` is used in the form of `{<label opt>angle}`.
7.3.4 How to move: shift

Dots can be shifted by specifying the optional argument \texttt{<shift coor>} immediately before \texttt{(coor)}. The empty shift option \texttt{<>} is not allowed.

7.3.5 Comparison: \texttt{\tzdot} and \texttt{\tzcdot}

The most important difference between \texttt{\tzcdot} and \texttt{\tzdot} is that \texttt{\tzcdot} is affected by TikZ’s scaling factor, but \texttt{\tzdot} is not. This is critical when \texttt{xscale} is not equal to \texttt{yscale}. 

43
The following table further shows the differences between them.

<table>
<thead>
<tr>
<th>% concept</th>
<th>% single</th>
<th>% multi</th>
<th>% size control</th>
<th>% key=default size</th>
</tr>
</thead>
<tbody>
<tr>
<td>node [circle]</td>
<td>\tzdot</td>
<td>\tzdots</td>
<td>\settzdotsize</td>
<td>[tzdot=2.4pt] % diameter</td>
</tr>
<tr>
<td>circle</td>
<td>\tzcdot</td>
<td>\tzcdots</td>
<td>\settzcdotradius</td>
<td>[tzcdot=1.2pt] % radius</td>
</tr>
</tbody>
</table>

Remark:

- In TikZ, a ‘node’ is ‘not’ affected by ‘scaling’ unless the TikZ option `transform shape` is used together. \tzdot is also useful for labelling a large dot.
  - In \tzdot, <label> is a label in a *label node* for a node dot (as a *main node*). So if a main node dot gets larger or smaller, its label moves accordingly. (Unlike, the labels with \tzcdot or \tzcdots.)
  - The position of <label> in \tzcdot does not depend on the size of dots.

- The package tzplot takes \tzdot as a *standard dot*, not \tzcdot. So, you can apply the THREE WAYS (on page 41) to change the size of any standard dots.

### 7.4 \tzdots(\ast): Multiple node dots

\tzdots takes an arbitrary number of coordinates as arguments to print multiple circle node dots with the *diameter* (or *minimum size*) of 2.4pt, by default.

This is a *semicolon version* macro, with the repeating pattern \(<\text{coor}>\{<\text{label}>\}[<\text{angle}>]\), which means that you need to type a *semicolon* ‘;’ at the end of the coordinate repetition. The semicolon says, “The repetition ends here.”

Remark:

- DO NOT FORGET to enter ‘;’ at the end of iteration.
  - Tokens such as ‘!’ and ‘#’ other than the open parenthesis ‘(’ work like the semicolon ‘;’ to indicate the end of iteration. But it is highly recommended to use ‘;’ for consistency.

- Without the semicolon ;, the macro does nothing or a compile error occurs.

The starred version \tzdots* prints multiple filled node dots.

```latex
% syntax: minimum
\tzdots*\(<\text{coor}>\){<\text{coor}>\}..repeated..\(<\text{coor}>\) ;
% syntax: full
\tzdots*[<\text{node opt}>][<shift coor>\ (<\text{coor}>\{<\text{label}>\}[<\text{label opt}>][<angle>])
..repeated.. ()[] ; (<dot size>)
% defaults
*[tzdot=2.4pt]<< \(<m>\)[] ..repeated.. ()[] ; (2.4pt)
```
How to label Each coordinate can be followed by the optional arguments \{<label>\} and [<angle>] to label dots. So the triple (\{<coor>\}\{<label>\}[<angle>]) is the whole repeating pattern.

How to change the size of dots There are Three Ways of changing the diameter of node dots, as discussed in Section 7.3.1 on page 41.

1. The simplest way is to use the last parenthesis optional argument, after the semicolon.

2. Another way is to use the style tzdot, like \texttt{\textbackslash tzdots[tzdot=3pt]}.... If both options are used the \texttt{tzdot} option style wins.

3. You can also use the macro \texttt{\textbackslash settzdotsize}. The effect remains within the \texttt{tikzpicture} environment unless it is changed again.

How to change colors With the first optional argument [\texttt{<node opt>}] you can change the color of node dots. You can also change the color of each label by [\texttt{<label opt>}].
Remark:

- `\<node opt>` is the option of a main node and `\<label opt>` is the option of a label node.
- `\<label opt>` is used in the form of `\<[label opt]angle\>`, like `\[red]90`.
- You can control all labels together using `every label/.style` as in the following examples:

```latex
% \tzdots: every label/.style
\begin{tikzpicture}
\tzhelplines(4,3)
\settzdots{6pt}
\tzdots*\[red\]
(0,0)(1,1){\textbf{Ben}}[\[blue]-90](2,1)(3,0);
\tzdots*\[thick,blue,fill=green\]
(1,2){A}(2,2){Ben}[\[red]-90](3,2){C}(4,2){D};(8pt)
\tikzset{every label/.style=\{draw,text=red\}}
\tzdots*\[blue\]
(1,3){A}(2,3){B}(3,3){C}(4,3){D};
\end{tikzpicture}
```

Shift You can move the coordinates of dots by specifying `\<shift coor>` option immediately before the first coordinate. The empty shift option `<>` is not allowed.

```latex
% \tzdots: shift
\begin{tikzpicture}
\tzhelplines(4,3)
\settzdots{\red}
(0,0){Ace}[[font=\LARGE\ttfamily]-90]
(2,1){\textbf{Bob}}[[blue]135]
(3,2){$C_1$}$\ bullet\ N_o$[[align=center]0]
\end{tikzpicture}
```
8 Coordinates

8.1 \tzcoor and \tzcoor*

8.1.1 \tzcoor

For example, \tzcoor(0,0)(A) means that the coordinate (0,0) is named (A).

\tzcoor(0,0)(A) \textit{is an abbreviation of:}
\begin{itemize}
\item path (0,0) coordinate (A);
\item or \coordinate (A) at (0,0);
\end{itemize}

% syntax: minimum
\tzcoor(<coor>)(<name>)
% syntax: medium
\tzcoor(<coor>)(<name>)[<label>][<angle>]
% syntax: full
\tzcoor<shift coor>(<coor>)(<name>)[<label>][<label opt>angle]
% defaults
<>(<m>)(<m>)

Here, <m> stands for ‘mandatory.’ \tzcoor takes two mandatory arguments in parenthesis.

How to label You can put a label to a coordinate by specifying the optional arguments \{<label>\} and [<angle>] immediately after (name).

\tzcoor(0,0)(A){\$A\$}[0] \textit{works like:}
\begin{itemize}
\item path (0,0) coordinate [label=\{0:\$A\$\}] (A);
\end{itemize}

% \tzcoor
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoor(0,0)(A){\$A_1\$} % TikZ default: 90 or above
\tzcoor(2,1)(B){\$B_2\$\ending point}[\{align=left,red\}0]
\draw (A) -- (B);
\end{tikzpicture}

Shift You can move the coordinate by specifying the optional argument \textit{shift coor} before (<coor>). The empty shift option <> is not allowed.

% \tzcoor: shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoor<0,1>(0,0)(A){\$A_1\$} %
\tzcoor(2,1)(B){\$B_2\$\ending point}[\{align=left,red\}0]
\tzline(A)(B)
\end{tikzpicture}
The starred version \texttt{tzcoor*} works like \texttt{tzcoor} with one exception. It prints a ‘node dot’ of the size 2.4pt, by default, at a specified coordinate.

% syntax: minimum
\texttt{tzcoor*}(<coor>)(<coor name>)
% syntax: medium
\texttt{tzcoor*}(<coor>)(<coor name>){<label>}[<angle>]
% syntax: full
\texttt{tzcoor*}[<dot opt>]<shift coor>
   ( <coor> )( <name> ) { <label> } [ [ <label opt> ] <angle> ] (<dot size>)
% defaults
*[]<>(<m>)(<m>){} (2.4pt)

\texttt{tzcoor*}(0,0)(A) \% works like:
\begin{tikzpicture}
\draw (0,0) coordinate (A);
\end{tikzpicture}

\texttt{tzcoor*}(0,0)(A)\{A\}[right] \% works like:
\begin{tikzpicture}
\draw (0,0) coordinate (A);
\end{tikzpicture}

% \texttt{tzcoor*}
\begin{tikzpicture}
\draw[help lines] (4,3);
\draw[help lines] (0,3);
\end{tikzpicture}
\begin{tikzpicture}
\draw[help lines] (4,3);
\draw[help lines] (0,3);
\end{tikzpicture}

Changing the color and size of a dot You can change the color of a dot by specifying [<dot opt>], which is, in fact, Ti\textit{k}Z’s \texttt{node} option. To change the size of dots, you can apply the \textbf{THREE WAYS} (see Subsection 7.3.1 on page 41).

% \texttt{tzcoor*}: color, size
\begin{tikzpicture}
\draw[help lines] (4,3);
\end{tikzpicture}
\begin{tikzpicture}
\draw[help lines] (4,3);
\end{tikzpicture}

\texttt{tzcoor*}(1,1)(A) (A)
\texttt{tzcoor*}(2,1)(B) [0]
\texttt{tzcoor*}[red](1,2)(C)[180]
\texttt{tzcoor*}[fill=none,tzdot=6pt](3,2)(D)[45]
\texttt{tzcoor*}[blue,thick,fill=green](4,0)(E)[180][6pt]
\end{tikzpicture}

Shift The optional argument <shift coor> works just like in \texttt{tzcoor}. The empty shift option <> is not allowed.
8.2 \texttt{\textbackslash{tzcoors}} and \texttt{\textbackslash{tzcoors*}}: Semicolon versions

8.2.1 \texttt{\textbackslash{tzcoors}}

The macro \texttt{\textbackslash{tzcoors}} takes an arbitrary number of pairs of coordinates and their names as arguments. For example, \texttt{\textbackslash{tzcoors}(0,0)(A) (1,1)(B) (2,2)(C)}; means that the coordinate (0,0) is represented by the name (A), (1,1) by (B), and (2,2) by (C).

\begin{tikzpicture}
  \tzcoors(0,0)(A) (1,1)(B) (2,1)(C) (3,0)(D) ; % works like:
  \path (0,0) coordinate (A)
  (1,1) coordinate (B)
  (2,1) coordinate (C)
  (3,0) coordinate (D);
\end{tikzpicture}

\begin{tikzpicture}
  \tzcoors(0,0)(A) (1,1)(B) (2,1)(C){[0]} (3,0)(D){[90]} ; % works like:
  \path (0,0) coordinate (A)
  (1,1) coordinate (B)
  (2,1) coordinate [label=0:C] (C)
  (3,0) coordinate [label=90:D] (D);
\end{tikzpicture}

You can add a label to each specified coordinate by adding the optional arguments \{\texttt{\textbackslash{label}}\} and \{\texttt{\textbackslash{angle}}\} immediately after \texttt{(\textbackslash{name})}.

\begin{tikzpicture}
  \tzcoors(0,0)(A){Ace}[font=LARGE\texttt{\textbackslash{ttfamily}}]-90
  (2,1)(B){\texttt{\textbackslash{textbf}}(Bob)}[blue]135
  (3,2)(C){\texttt{\textbackslash{texttt}}(No)}[align=center]0;
  \draw (A) -- (B) -- (C);
\end{tikzpicture}

By the option \texttt{\textbackslash{\textbackslash{shift coor}}}, all specified coordinates are shifted. The empty shift option \texttt{<>} is not allowed.

By the option \texttt{\textbackslash{\textbackslash{shift coor}}}, all specified coordinates are shifted. The empty shift option \texttt{<>} is not allowed.
8.2.2 \texttt{tzcoors*}

The starred version \texttt{tzcoors*} takes an arbitrary number of coordinates as arguments to print node dots at the coordinates. Full repeating pattern is (\texttt{<(coor)> \texttt{(name)}>\texttt{<label>}}\texttt{[<angle]>}). It is required to type a \texttt{semicolon \; ;} to indicate when the iteration of coordinates ends.

\begin{verbatim}
\begin{tikzpicture}
\helplines(4,3)
\tzcoors*(0,0)(A) (1,1)(B) (2,1)(C){C} (3,0)(D);
\tzlines(A)(B)(C)(D);
\end{tikzpicture}
\end{verbatim}

You can label each dot by specifying the optional arguments \texttt{\{<label>\}} and \texttt{\[<angle>\]} after the pair (\texttt{<(coor)> \texttt{(name)}>}).

\begin{verbatim}
\begin{tikzpicture}
\helplines(4,3)
\tzcoors*(0,0)(A) (1,1)(B) (2,1)(C) {B} (3,3)(D){D}[0] ;
\end{tikzpicture}
\end{verbatim}

You can change the dot color by \texttt{[<dot opt>]} and the label color by \texttt{[<label opt>]]. You can apply the THREE WAYS (on page 41) to change the dot size. The simplest way of changing the dot size is to specify the \texttt{last} (even after the semicolon) parenthesis option \texttt{(<dot size>).}

\begin{verbatim}
\begin{tikzpicture}
\helplines(4,3)
\tzcoors*[red](0,0)(A) (1,1)(B) {B} (2,1)(C) (3,3)(D)[blue]0] ; (6pt)
\end{tikzpicture}
\end{verbatim}

By specifying the optional argument \texttt{<shift coor>} immediately before the first coordinate, you can move all specified coordinates. The empty shift option \texttt{<>} is not allowed.
8.3 \texttt{\textbackslash{}tzcoorsquick} and \texttt{\textbackslash{}tzcoorsquick*}

8.3.1 \texttt{\textbackslash{}tzcoorsquick}

You can see the coordinate array at a glance using \texttt{\textbackslash{}tzcoorsquick}, which displays specified names as text at the center of the coordinates, by default.

\begin{verbatim}
\begin{tikzpicture}
\tzhelplines(4,3)
\tztzcoorsquick(0,0)(A)(1,1)(Ben)(2,1)(Cate)(3,2)(Daniel);
\end{tikzpicture}
\end{verbatim}

A label can be suppressed by the empty braces \{\}. You can move the coordinates by specifying \texttt{<shift coor>} immediately before the first coordinate. The empty option \texttt{<>} is not allowed.

\begin{verbatim}
\begin{tikzpicture}
\tzhelplines(4,3)
\tztzcoorsquick(0,0)(A)(1,1)(Ben)(2,1)(Cate)(3,2)(Daniel);
\end{tikzpicture}
\end{verbatim}

8.3.2 \texttt{\textbackslash{}tzcoorsquick*}

The starred version \texttt{\textbackslash{}tzcoorsquick*} prints node dots on the coordinates and displays the names above (or 90 degree from) the dots, by default.
% syntax: minimum
\tzcoorsquick*(<coor>)(<name>)..repeated..(<coor>)(<name>);

% syntax: full
\tzcoorsquick*<dot opt><shift coor>(<coor>)(<name>)<label><angle>
..repeated.. (){}[];

% defaults
*[ tzdot=1.2pt ]<> (<m>)(<m>){}[] ..repeated.. (){}[];

\begin{tikzpicture}
\zhelplines(4,3)
\tzcoorsquick*(0,0)(A) (1,1)(Ben) (2,1)(Cate) (3,2)(Daniel);
\end{tikzpicture}

A label can be suppressed by the empty braces {}. You can change the dot size using the THREE WAYS (on page 41). You can shift the coordinate by specifying <shift coor> immediately before the first coordinate. The empty shift option <> is not allowed.

\begin{tikzpicture}
\zhelplines(4,3)
\tzcoorsquick*<0,1>(0,0)(A) (1,1)(Ben) (2,1)(Cate) (3,2)(Daniel);(6pt)
\end{tikzpicture}

\begin{tikzpicture}
\zhelplines(4,3)
\tzcoorsquick*[fill=none,blue,very thick](0,0)(A) (1,1)(B-1) (2,1)(C) (3,3)(D);[blue]0;(8pt)
\end{tikzpicture}

Remark: The first optional argument of \tzcoorsquick* is for only dots. You can use the TikZ option every label/.style={...} to control all the labels together. You can also control each labels using [label opt] for each coordinate.

\begin{tikzpicture}
\zhelplines(4,3)
\tzcoorsquick*[every label/.style={draw,red}](0,0)(A) (1,1)(B-1) (2,1)(C) (3,3)(D);[blue]0;(8pt)
\end{tikzpicture}

8.4 \tzgetxyval

\tzgetxyval extracts the values of x-coordinate and the y-coordinate in the unit of centimeter from a specified coordinate and saves the values in the user-defined macros, so that you can use them later. For example, \tzgetxyval(3,2){\xval}{\yval} results in \xval=3 and \yval=2.

\begin{tikzpicture}
\zhelplines(4,3)
\tzgetxyval[fill=none,blue,very thick](0,0)(A) (1,1)(B-1) (2,1)(C) (3,3)(D);[blue]0;(8pt)
\end{tikzpicture}
9 Plot Coordinates: \texttt{\textbackslash{}tzplot}: Semicolon Versions

9.1 \texttt{\textbackslash{}tzplot} and \texttt{\textbackslash{}tzplot*:} Syntax

\texttt{\textbackslash{}tzplot} takes an arbitrary number of coordinates as arguments. Each \(<\text{coor}>\) can be followed by the optional arguments \(<\text{label}>\) and \([\text{angle}>\] to label the coordinate. This is a \textit{semicolon version} and the whole repeating pattern is the triple \(<\text{coor}>\{\text{label}>\}[\text{angle}>\]. It is required to type a semicolon ‘;’ to indicate when the coordinate iteration ends.

The macro \texttt{\textbackslash{}tzplot} draws connected line segments that link specified coordinates.

\begin{verbatim}
% syntax: minimum
\tzplot(<coor>)(<coor>)..repeated..(<coor>) ;
% syntax: medium
\tzplot(<coor>)\{<label>\}[<angle>]..repeated..(<coor>)\{<label>\}[<angle>] ;
% syntax:full
\tzplot[<opt>]{<tension>}[<plot opt>]<shift coor>"<path name>"
   (<coor>)\{<label>\}[<label opt>angle]
..repeated..
(){}[] ; (<mark size> <code.append>
% defaults
[tzmark=2pt]{0}[smooth] <"= "(<m>){}[] ..repeated.. (){}[] ; (2pt) <
\end{verbatim}

The style \texttt{tzmark} is defined as follows:

\begin{verbatim}
% style: tzmark
\tikzset{
   tzmark/.style=
   {mark options=(solid,thin),mark size=#1},
   tzmark/.default=tzmarksize }
\end{verbatim}

\texttt{tzmarksize} is the \textit{radius} of a \texttt{mark} and the default is \texttt{2pt} as in Ti\textit{k}Z. The value of \texttt{tzmarksize} can be changed by the macro \texttt{settzmarksize}, like \texttt{settzmarksize(3pt)}.

The starred version \texttt{\textbackslash{}tzplot*} prints dot marks at specified coordinates, without drawing line segments connecting the coordinates, by default.

9.2 \texttt{\textbackslash{}tzplot*}: Dots and marks

The starred version \texttt{\textbackslash{}tzplot*} prints Ti\textit{k}Z marks (\texttt{*} by default) at specified coordinates. You can change the mark color and mark style using the first bracket optional argument.
Labels, marks, and mark size

You can also add labels to specified coordinates with the optional arguments \{<label>\} and \[<angle>\] immediately after each \(<\text{coor}>\).

\begin{tikzpicture}[scale=.8]
  \tzhelplines(4,3)
  \tzplot*(0,0)\{A\}[90](1,2)(2,2)(3,3){D}[0]; % works like:\draw [draw=none,mark=*] plot coordinates \{(0,0),(1,1),(2,2),(3,3)\};
  \node (0,0) \[label={[90]:D}\] {};
  \node (3,3) \[label={[0]:D}\] {};
  \tzplot*[mark=o,tzmark=6pt](0,3)(1,3)(2,2)(3,1);(10pt)
  \settzmarksize{4pt}
  \tzplot*[red, mark=heart](0,2)(1,2)(2,0)(3,2){D};
\end{tikzpicture}

There are three ways to change the mark size.

1. The simplest way is to use the parenthesis optional argument \(<\text{mark size}>\), \textit{immediately after the semicolon}.

2. You can use the style \texttt{tzmark}, like \texttt{tzmark=3pt}.

3. You can also use the macro \texttt{settzmarksize}, which is effective until the end of \texttt{tikzpicture} environment.

With \texttt{tzplot*}, you can draw line segments by giving the \texttt{TikZ}'s option \texttt{draw} in the first bracket optional argument, like \texttt{tzplot*[draw]}.

\begin{tikzpicture}[scale=1]
  \tzhelplines(4,3)
  \settzmarksize{3pt}
  \tzplot*[draw,mark=x](0,0)(1,1)(2,2)(3,3);
  \tzplot*[blue,mark=diamond*](0,0)(1,3)(2,2)(3,1);(10pt)
  \tzplot*[draw,dashed,red,mark=heart](0,2)(1,2)(2,0)(3,2);
\end{tikzpicture}

Remark: In \texttt{TikZ}, the \texttt{mark} shapes are affected by \texttt{scale}, \texttt{xscale}, and \texttt{yscale}.
\begin{tikzpicture}[yscale=.5]
\zhelplines(4,3)
\settzmarksize{3pt}
\tzplot*[draw,mark=x](0,0)(1,1)(2,1)(3,3);
\tzplot*[blue,mark=diamond*](0,3)(1,3)(2,2)(3,1);[10pt]
\tzplot*[draw,dashed,red,mark=heart](0,2)(1,2)(2,0)(3,2);
\end{tikzpicture}

\textbf{Shift}  You can move specified coordinates using the option \texttt{<shift coor>} before the first coordinate (to be precise, immediately before the option "\texttt{<path name>}" if it exists). The empty shift option \texttt{<>} is not allowed.

\begin{tikzpicture}
\zhelplines(4,3)
\tzplot*[draw](1,3){A}(2,3)(3,3){D}[0];
\tzplot*[draw,red]<-1,-1>(1,3){A}(2,3)(3,3){D}[0];
\tzplot*[draw,blue]<0,-3>(1,3){A}(2,3)(3,3){D}[0];
\end{tikzpicture}

\subsection{\texttt{tzplot}: Lines}

\texttt{tzplot} draws connected line segments connecting specified coordinates. (By default, \texttt{tension=0}.)

\begin{verbatim}
tzplot(0,0)(1,2)(2,2)(3,3); % works like:
\draw [tension=0] plot [smooth] coordinates { (0,0)(1,1)(2,2)(3,3) } ;
\end{verbatim}

\textbf{Options: draw, mark, mark options, etc.}  You can use the optional argument \texttt{[<opt>]} to change the style of lines and marks.

\begin{verbatim}
\begin{tikzpicture}[scale=.8]
\zhelplines(4,3)
\tzplot (0,0)(1,1)(2,1)(3,3);
\tzplot[dashed,mark=o](0,3)(1,3)(2,2)(3,1);
\tzplot[red,mark=ball,mark options={ball color=purple}](0,1)(1,2)(2,0)(3,2);
\end{tikzpicture}
\end{verbatim}

To close the path of \texttt{tzplot} you can use the option \texttt{smooth cycle} in the first bracket optional argument \texttt{[<opt>]} or in the second bracket optional argument \texttt{[plot opt]}.
Labels  You can label specified coordinates with the options \{<label>\} and [<angle>] immediately after each \{<coor>\).

Shift  You can also move the line segments by specifying the option \{<shift coor>\} before the first coordinate (to be precise, immediately before the option "<path name>" if it exists). The empty shift option <> is not allowed.

name path for intersections  To find the intersection points of two lines, you may want to name the paths first, like \{name path=<path name>\} in TiKZ. With \tzplot, you can do it by specifying the quote optional argument "<path name>" immediately before the first coordinate.

Extending the path  In order to extend a path, formed by \tzplot, from the last coordinate, you can write TiKZ code in the very last optional argument \{<code.append>\}, after the semicolon.
9.4 \texttt{\textbackslash{}tzplot}: Curves

With \texttt{\textbackslash{}tzplot}, the default value of \texttt{tension} is 0. You can draw a curve with \texttt{\textbackslash{}tzplot}, by specifying the optional argument \texttt{\textbackslash{}{tension}} before the coordinates or between the first and second bracket options (if they exist).

\begin{verbatim}
\texttt{\textbackslash{}tzplot\{blue,smooth cycle\}(1)(0,0)(1,2)(2,2)(3,3); \% works like:}
\texttt{\draw \{blue,tension=1\} plot \{smooth cycle\} coordinates \{(0,0)(1,1)(2,2)(3,3)\};}
\end{verbatim}

\begin{verbatim}
\begin{tikzpicture}[scale=.8]
tzhelplines(4,3)
tzplot\{tension=1\}(0,0)(1,1)(2,1)(3,3);
tzplot\{mark=o\}(0,3)(1,3)(2,2)(3,1); \% tension=0
\end{tikzpicture}
\end{verbatim}

To plot curves, the macro \texttt{\textbackslash{}tzplot\texttt{curve}} is provided. Basically, \texttt{\textbackslash{}tzplot\texttt{curve}} is the \texttt{tension=1} version of \texttt{\textbackslash{}tzplot}. (See Section 9.6 on page 58.)

9.5 \texttt{\textbackslash{}tzplot}: Bars and combs

With \texttt{\textbackslash{}tzplot}, you can draw bars or combs, using the Ti\texttt{kZ} options \texttt{ybar}, \texttt{xbar}, \texttt{ycomb}, and \texttt{xcomb}.

\begin{verbatim}
\begin{tikzpicture}[scale=.8]
tzhelplines(4,3)
tzplot\{draw\}(0,0)(1,1)(2,1)(3,3);
tzplot\{ybar\}(0,3)(1,3)(2,2)(3,1);
tzplot\{dashed,red,mark=heart\}(0,1)(1,2)(2,0)(3,2);\%(5pt)
\end{tikzpicture}
\end{verbatim}

\begin{verbatim}
\begin{tikzpicture}[scale=.8]
tzhelplines(4,3)
tzplot\{xbar\}(.5,2.5)(1,3)(2,2)(3,1);
tzplot\{xbar,bar width=2mm,fill,red!50\}(1,2.5)(1.5,2)(2,0)(3,1.5);\%(3pt)
\end{tikzpicture}
\end{verbatim}
Remark:

- Do not use `<shift coor>` for plotting bars or combs to avoid getting unexpected results. It gives you wrong bars because `<shift coor>` moves coordinates but not bars.
- It can be a mess when using the TikZ option `shift={(coor)}` with the type of mixed coordinates: native and named coordinates.

9.6 `\tzplotcurve(*)`

`\tzplotcurve` draws a curve connecting specified coordinates with `tension=1`, by default. Basically, it is equivalent to `\tzplot` with `[tension=1].

The starred version `\tzplotcurve*` draws a curve and displays marks *, by default. Basically, this is equivalent to `\tzpolot*[draw,tension=1].`

Since `\tzplotcurve` is a *semicolon version*, you need to enter a semicolon to indicate when the coordinate iteration ends. In repeating coordinates, each mandatory coordinate can have a label. So the whole repeating pattern is the triple `{<coor>}{<text>}[<pos>]. For example, `(A){here}[above]` represents `(A) node [above] {here} in TiKZ.
Options: lines, labels, colors, smooth cycle  Use the first bracket option to control the colors of lines or labels.

\tzplotcurve(0,0)(1,2)(2,2){A}[below](3,3){B}[right]; \% works like:
\draw [tension=1] plot [smooth] coordinates { (0,0)(1,1)(2,2)(3,3) }
(2,2) node [below] {A}
(3,3) node [right] {B} ;

You can change the color of all labels together by adding [text=<color>] to the first bracket option list.

\begin{tikzpicture}
% \tzplotcurve(*)
\tzhelplines(4,3)
\tzplotcurve*[red,text=blue]
(0,0)(1,2)(2,2)(3,3)(4,1){E}[0];
\tzplotcurve(1,.5)(2,3)(3,2)(2,1)(3,1);
\end{tikzpicture}

The simplest way to change the mark size is to specify \(<\text{mark size}>\) immediately after the semicolon ;. To close the path of \tzplotcurve, you can use the Ti\kZ option smooth cycle in the first bracket option or in the second bracket option.

\begin{tikzpicture}
% \tzplotcurve(*): smooth cycle, mark size
\begin{tikzpicture}
\tzhelplines(4,3)
\tzplotcurve*[red,text=blue][smooth cycle]
(0,0)(1,2)(2,2)(3,3)(4,1){E}[0];%(3pt)XX
\tzplotcurve*[smooth cycle](1,.5)(2,3)(3,2)(2,1)(3,1);
\end{tikzpicture}
\end{tikzpicture}

Tension  You can change the value of tension (tension=1 by default) by specifying the option \{<tension>\} before the coordinates or between the two bracket options if they exist.

\begin{tikzpicture}
% \tzplotcurve: tension
\begin{tikzpicture}
% \tzplotcurve
\tzhelplines(4,3)
\tzcoor(.5,.5)(A)
\tzplotcurve[blue]{2}(1,3)(A)(4,0);
\tzplotcurve[thick]{1,3}(A)(4,0); % default: tension=1
\tzplotcurve[red]{.5}(1,3)(A)(4,0);
\tzplotcurve[dashed]{0}(1,3)(A)(4,0);
\end{tikzpicture}
\end{tikzpicture}

Shift  Use the optional argument <shift coor> before the first coordinate (to be precise, immediately before *<path name>*", if exists). The empty shift option <> is not allowed.
Extending the path  In order to extend the path created by `\tzplotcurve(*)` from the last coordinate, you can directly write TikZ code in the very last optional argument `<code.append>`, after the semicolon.

You can also use `<--cycle>` to close the path with a straight line from the last coordinate to the first coordinate.

name path for intersection points  You can name the path of `\tzplotcurve` by specifying the option "<path name>" immediately before the first coordinate.
10 Nodes

10.1 \tznodemand \tznodestar

The macro \texttt{\tznodenode} allows you to put text at specified coordinates. You can also optionally name a node so you can refer to the \textit{node coordinate} later.

The starred version \texttt{\tznodenode*} is equivalent to \texttt{\tznodenode[draw]}, which draws the perimeter of the specified node. The default node shape is a \textit{rectangle}.

\begin{verbatim}
% syntax: minimal
\tznodemacro{<coor>}{<text>}
% syntax: full
\tznodemacro{<opt>}{<shift coor>}{<coor>}{<node name>}{<text>}{<node opt>}
% defaults
\[<\{\texttt{<node opt>}\}][]\]
\end{verbatim}

\begin{verbatim}
\tznodemacro{0,0}{text} % works like:
\path (0,0) node {text};
% or
\node at (0,0) {text};
\end{verbatim}

\begin{verbatim}
\tznodemacro{draw}{0,0}{A}{text}{above right} % works like:
\node [draw] (A) at (0,0) [above right] {text};
\end{verbatim}

The \texttt{\tznodemacro{*}} prints the perimeter of a node, which is a \textit{rectangular} by default.

\begin{verbatim}
% syntax: minimal
\tznodemacro*{<coor>}{<text>}
% syntax: full
\tznodemacro*{<opt>}{<shift coor>}{<coor>}{<name>}{<text>}{<node opt>}
% defaults
*\{draw\}\[\{\texttt{<node opt>}\}][]\]
\end{verbatim}

\begin{verbatim}
\tznodemacro*{0,0}{A}{text}{above right} % works like:
\node [draw] (A) at (0,0) [above right] {text};
\end{verbatim}

\textbf{Putting text} You can use \textit{TikZ} options in the first bracket optional argument \texttt{[<opt>]} or the second bracket option \texttt{[<node opt>]} to put text with different colors, fonts, and so on.

\begin{verbatim}
\begin{tikzpicture}
\tzhelplines(4,3)
\tznodemacro{1,1}{text}
\tznodemacro{2,2}{another}{align=left}
\tznodemacro{3,1}{text}
\tznodemacro*{blue}{3,0}{s\texttt{\textshape} Text}{r} % right
\tznodemacro*{4,0}{another}{align=center,\texttt{\itshape} text}
\end{tikzpicture}
\end{verbatim}
**Abbreviations**  You can use *abbreviations* such as a for above, l for left, ar for above right, bl for below left, and so on to indicate where the *text* of a *main node* is placed. (See also Section 1.2 on page 1.)

**Warning:** A *main node* is placed by the placement words, while a *label node* is placed by the placement words or the `<angle>` expression.

- You CANNOT use the abbreviations to place a *label node* for points or dots or coordinates.
- Instead, use `<angle>` or the original (unabridged) placement words such as right and below left to place `<label>`.
- Note that, on the other hand, you CANNOT use the `<angle>` option to place a *main node*.

```latex
% \tznode(*): main node, label node
\begin{tikzpicture}
\tzhelplines(4,3)
\tznode*{(1,1)}{A}[ar]
\tznode*[red]{{(1,1)}{B}}[b]
\tznode*{{(1,1)}{C}}[l]
\tznode*{inner sep=0pt,text=blue,scale=2}{{(3,2)}{Text}}[label=180:Left,pin=45:Above Right]
\tznode*[inner sep=0pt,text=red, scale=2]{{(3,2)}{Text}}[b=2cm] % below=2cm
\end{tikzpicture}
```

**Naming nodes**  You can name a node at a specified coordinate (<coor>) by specifying (<name>) immediately after the coordinate. You can use the node name as a node coordinate.

```latex
% \tznode(*): name nodes
\begin{tikzpicture}
\tzhelplines(4,3)
\tznode*(1,1){A}
\tznode*[fill=blue,text=yellow]{{(2,3)}{B}}
\tznode*[circle,text=red]{{(4,1)}{C}}
\tzline[->]{{(A)}{B}}
\tzto[->,bend left,dashed]{{(B)}{C}}
\end{tikzpicture}
```

**Shift**  You can move the coordinates by specifying the option `<shift coor>` immediately before the coordinate (<coor>). The empty shift option <> is not allowed.

```latex
% \tznode(*): shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tznode*<-1,0>{(1,1)}{A}
\tznode*[fill=blue,text=yellow]<1,-1>{{(2,3)}{B}}
\tznode*[circle,text=red]<-2,-1>{{(4,1)}{C}}
\tzline[->]{{(A)}{B}}
\tzto[->,bend left,dashed]{{(B)}{C}}
\end{tikzpicture}
```
10.2 \tznodedot(*)

\tznodedot names a node and prints a circle node dot. \tznodedot is basically the same as \tzdot, except for one thing. \tznodedot names a node.

The starred version \tznodedot* prints a filled circle node dot (of the size 2.4pt, by default), just like \tzdot*. But it optionally names a node.

% syntax: medium
\tznodedot(<coor>)(<node name>){<label>}[<angle>]
% syntax: full
\tznodedot*[<opt>]<shift coor>
\t\\ttzdottext{(coor)}{(node name)}{<label>}[<label opt>][angle] (<dot size>)
% defaults
\t*[]<>(<m>)(){}(2.4pt)

Since \tznodedot(*) prints a node dot, its <label> is placed by <angle>.

% \tznodedot(*): label, color
\begin{tikzpicture}
\tzhelplines(4,3)
\tznodedot(1,1)(A){A}[180]
\tznodedot*(2,3)(B){textbf{B}}
\tznodedot[rectangle](4,1)(C){\red{C}}
\t\tztline[-](A)(B)
\t\tztto[->,bend left,dashed](B)(C)
\end{tikzpicture}

You can apply the Three Ways (on page 41) to change the size of node dots. The simplest way is to use the last parenthesis option (\texttt{<dot size>}).

% \tznodedot(*): size
\begin{tikzpicture}
\tzhelplines(4,3)
\settztdotsize{5pt} %
\tznodedot(1,1)(A){A}[180]
\tznodedot*(2,3)(B){B}
\tznodedot[rectangle](4,1)(C){0}(10pt) %
\t\tztline[-](A)(B)
\t\tztto[->,bend left,dashed](B)(C)
\end{tikzpicture}

You can move the coordinates of dots by specifying the \texttt{<shift coor>} option immediately before the coordinate. The empty shift option <> is not allowed.

% \tznodedot(*): shift
\begin{tikzpicture}
\tzhelplines(4,3)
\settztdotsize{5pt}
\tznodedot(1,1)(A){A}[180]
\tznodedot*(2,3)(B){B}
\t\tznodedot[rectangle]<-1,-1>(4,1)(C){0}(10pt) %
\t\tztline[-](A)(B)
\t\tztto[->,bend left,dashed](B)(C)
\end{tikzpicture}
10.3 \tznodename and \tznodename*

\tznodename draws and names a rectangle node with text in it.

% syntax:
\tznodename[<opt>](<coor>)(<node name>){<text>}[<node opt>]
% defaults
{}<>(<m>)()\{text=black\}

% \tznodename
\begin{tikzpicture}
\tzhelplines(4,3)
\tznodename(0,0)(A)
\tznodename(4,2)(B)[circle]
\tzline[blue,thick](A)(B)
\tznodename[scale=1.5](2,1){Node frame}
\quad\{label=180:Left, pin=-45:pin\}
\end{tikzpicture}

% \tznodename: fill (opacity=1)
\begin{tikzpicture}
\tzhelplines(4,3)
\tznodename(0,0)(A)
\tznodename(4,2)(B)[circle]
\tzline[blue,thick](A)(B)
\tznodename[fill=green,scale=1.5](2,1){Node frame}
\quad\{label=180:Left, pin=-45:pin\}
\end{tikzpicture}

The starred version \tznodename* fills the rectangle with color (black!50 by default) with fill opacity=.3 but with text opacity=1, by default.

% syntax:
\tznodename*[<opt>]<shift coor>
\quad \{<coor>\}(<node name>){<text>}[<node opt>]\{<fill opacity>\}
% defaults
*{}<>(<m>)()\{fill=black!50,fill opacity=.3,text opacity=1\} <> (<>)(<>){draw=black,text=black}{.3}

Remark: \tznodename works very similar to \tznodename*, but their ‘starred versions’ work differently. While \tznodename* draws the perimeter of a node, \tznodename* fills a node with color.

% \tznodename*: opacity=.3 (by default)
\begin{tikzpicture}
\tzhelplines(4,3)
\tznodename(0,0)(A)
\tznodename(4,2)(B)[circle]
\tzline[blue,thick](A)(B)
\tznodename*[green,scale=1.5](2,1){Node frame}
\quad\{label=180:Left, pin=-45:pin\}
\end{tikzpicture}
You can change the fill opacity of `\tznodetframe*` by specifying the last curly brace option `{fill opacity}`). You can also move the node by specifying the `<shift coor>` option immediately before the coordinate. (The empty shift coor <> is not allowed.)

```
% \tznodetframe(*): opacity, shift
\begin{tikzpicture}
\tzhelplines[solid](4,3)
\tznodetbox(0,0)
\tznodetframe*(1,2)\{A\}(.1) % opacity
\tznodetframe\{fill=yellow\}(2,2)\{B\}
\tznodetframe*[\{red\}](3,2)\{Cat\}
\tznodetframe\{fill=green\}\{-1,-2\}(4,2)\{Dog\} % shift
\tzto[->,bend left=45](A.north)(C.90)
\tzto[->,bend right=45,dashed](B.-135)(D.west)
\end{tikzpicture}
```

**Remark:** In addition to using the option `{<fill opacity)>`, you can use a macro to change the default value.

- The default fill opacity can be changed by the macro `\settztfillopacity`.
- The default fill color is `black!50`. You can use the macro `\settztfillcolor` to change the default fill color.
- With `\tznodetframe`, you can change the color of the perimeter and text with the second bracket option `{<node opt>}`.

### 10.4 `\tznodetcircle` and `\tznodetcircle*`

`\tznodetcircle` works just like `\tznodetframe` but with a circle node.

```
% syntax
\tznodetcircle[<opt>]<shift coor><<coor>>\{<node name>\}\{<text>\}\{<node opt>\}
% defaults
[circle]<><\{<m>\}\{\text=black\}
```

`\tznodetcircle*` works just like `\tznodetframe*` but with a circle node.

```
% syntax
\tznodetcircle*[<opt>]<shift coor>
  \{<coor>\}\{<node name>\}\{<text>\}\{<node opt>\}\{<fill opacity>\}
% defaults
*\{[circle,fill=black!50,fill opacity=.3, text opacity=1]\} <> \{(<m>\}\{\{\text=black\}\{.3\}
```

```
% \tznodetcircle
\begin{tikzpicture}
\tzhelplines(4,3)
\tznodetcircle(0,0)\{A\}
\tznodetcircle(4,2)\{circle\}
\tzline[blue, thick](\{A\}\{B\})
\tznodetcircle*[scale=1.5]\{2,1\}\{Node frame\}
\end{tikzpicture}
```
You can change the fill opacity of \texttt{\tznodencircle*} by specifying the last curly brace option \{<fill opacity>\} or using \texttt{\setzfillopacity}. You can also move the node by specifying the \texttt{<shift coor>} option immediately before the coordinate. (The empty shift coor <> is not allowed.)

10.5 \texttt{\tznodellipse} and \texttt{\tznodellipse*}

\texttt{\tznodellipse} works just like \texttt{\tznodeframe} but with an ellipse node.

The starred version \texttt{\tznodellipse*} works just like \texttt{\tznodeframe*} but with an ellipse node.
11 Lines

11.1 \tzline: Connecting two points

\tzline connects two points with a straight line.

% syntax: minimum
\tzline(<coor1>)(<coor2>)

% syntax: medium
\tzline[<opt>](<coor1>){<text1>}[<node opt1>]
\tzline[<opt>](<coor2>){<text2>}[<node opt2>]

% syntax: full
\tzline[<opt>][shift coor="<path name>"]
\tzline[<opt>]<shift coor="<path name>" [code.append]

% defaults
[]<>{}[above]{}</code.append

\tzline(0,1)(2,1) \% works like:
\draw (0,1) -- (2,1);

11.1.1 Line styles

You can use the first optional argument [<opt>] to control the line styles.

\tzline[blue](0,1)(2,1) \% works like:
\draw [blue] (0,1) -- (2,1);

% \tzline
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline(0,0)(3,3)
\tzline[->,color=red](0,0)(3,3)
\tzline[dashed](1,2)(4,0)
\tzline[dotted](1,2)(4,0);
\end{tikzpicture}
11.1.2 Adding text

You can add text by specifying the optional arguments \{<text>\} and \[<node opt>\].

**Text next to the line**  With the options \{<text1>\}[<node opt1>\] in-between two coordinates, you can add text next to the line, with the option \[above,midway\] by default.

\begin{tikzpicture}
\tzhelplines(4,3)
\tzline[blue](0,0)(4,0){\textbf{my line}}[>]
\tzline[blue](0,0)(1,1){\textbf{my line}}[>]
\tzline[->](1,1)(4,1){\textbf{line A}}[>]
\end{tikzpicture}

You can use the abbreviations of TikZ basic placement options such as a for above, bl for below left, and so on. (For more details, see page 2.)

**Text at or around the last coordinate**  You can also add text at (by default) or around the second coordinate by specifying \{<text2>\} and \[<node opt2>\] immediately after the second coordinate.

\begin{tikzpicture}
\tzhelplines(4,3)
\tzline[blue](0,0)(4,4){\textbf{my line}}[>]
\end{tikzpicture}

11.1.3 Moving the line: shift

You can move the line generated by \tzline by specifying the option <shift coor> before the first coordinate (to be precise, immediately before the option "<path name>", which is put immediately before the first coordinate, if it exists). The empty shift coor <> is not allowed.

\begin{tikzpicture}
\tzhelplines(4,3)
\tzline[blue,>-](0,0)(4,4){\textbf{my line}}[>]
\tzline[blue]<1,0>(0,0)(4,4){\textbf{my line}}[>]
\tzline[dashed](1,1)(4,1){\textbf{line A}}[>]
\end{tikzpicture}
11.1.4 Extending the path

You can extend the path of \texttt{\textbackslash tzline} from the last coordinate, by writing Ti\textit{k}Z code in the last optional argument \texttt{<code.append>}. 

```latex
% \texttt{\textbackslash line: shift}
\begin{tikzpicture}
  \tzhelplines(4,3)
  \tzline[blue,->](0,0)(3,3)<arc(0:-60:1cm)>
  \tzline[dashed,->](1,2)(4,0)
  \draw[<to[bend left] ++(-4,1) node [a] \{ends here!\}>]
\end{tikzpicture}
```

11.1.5 Naming the path: Intersection points

When you specify the option \texttt{"<path name>"} immediately before the first coordinate, it works like \texttt{[name path=<path name>] in the option list of [opt]}. You can use this name of path to find intersection points.

```latex
% \texttt{\textbackslash line+: name path}
\begin{tikzpicture}
  \tzhelplines(4,3)
  \tzline[blue,->] "dem" (0,0)(3,3)\{demand\}[r]
  \tzline[dashed,->] "supply"(1,2)(4,0)\{supply\}[ar]
  \tzxpoint*{dem}{supp}
\end{tikzpicture}
```

11.2 \texttt{\textbackslash line+: Relative coordinates}

The \textit{plus version} \texttt{\textbackslash line+} takes the second coordinate \texttt{(<coor2>)} relative (with ++) to the first coordinate \texttt{(<coor1>)}. 

\textit{Everything else is the same as in \texttt{\textbackslash line}.}

```latex
% syntax: full
\tzline+[<opt>]<shift coor>"<path name>"
\tzline+[<coor1>]{<text1>}[<node opt1>]
\tzline+[<coor2>]{<text2>}[<node opt2>]<code.append>

% defaults
+[]""{<m>}{\{above\}{<m>}{}}\{}<>
```

\texttt{\textbackslash line+}(0,1)(2,1) \% works like: 
\texttt{\draw (0,1) -- ++ (2,1);}

\texttt{\textbackslash line+}[dashed]"AA"(0,1){\{A\}[red](2,1){\{B\}[right,blue]} \% works like: 
\texttt{\draw [dashed,name path=AA]
(0,1) -- node [red] {A} ++ (2,1) node [right,blue] {B};}
11.3 Styles: tzshorten and tzextend

The styles `tzshorten` and `tzextend` are defined as follows.

For example, `[tzshorten={2mm}{1mm}]` is equivalent to `[shorten <=2mm, shorten >=1mm]` in TiKZ. Simple `[tzshorten]` means that `[tzshorten={2pt}{2pt}]` by default.

The style `tzextend` is a negative `tzshorten`. For example, `tzextend{2mm}{1mm}` is equivalent to `tzshorten{-2mm}{-1mm}`.

11.4 \tzlines: Connecting multiple points: Semicolon version

`\tzlines` connects two or more points with connected line segments. Since `\tzlines` takes an arbitrary number of coordinates as arguments, it is a **semicolon version**. So, you need to enter a semicolon `;` to indicate when the coordinate iteration ends.
The whole repeating pattern in `\tzlines` is the triple `<coor>{<text>}[<node opt>]`.

\begin{tikzpicture}
\helplines(4,3)
\tzlines[red,thick,<->,text=blue]
(0,1){Up}
(1,2){Down}[a,sloped]
(3,1){Up}
(4,3){line A} ;
\tzlines[dashed](0,0)(1,1)(2,1)(3,0){line B}[r];
\end{tikzpicture}

**Line styles**  Use the first optional argument `<opt>` to control the style of the connected line drawn by `\tzlines`.

\begin{tikzpicture}
\helplines(4,3)
\tzlines[blue,<-](0,1)(1,2)(3,1)(4,3); \draw (1,1) -- (2,2) -- (3,1) -- (4,3);
\tzlines[dashed](0,0)(1,1)(2,1)(3,0); \draw (1,1) -- (2,2) -- (3,1) -- node {C} (4,3) node [right] {D};
\end{tikzpicture}

**Adding text**  You can add text in the midway, by default, of each line segment by specifying the options `{<text>}` and `[<node opt>]` immediately after each coordinate (except the last one). The options `{<text>}` and `[<node opt>]` after the last coordinate put `<text>` at or around the last coordinate.

\begin{tikzpicture}
\helplines(4,3)
\tzlines[red,thick,<-,text=blue]
(0,1){Up}
(1,2){Down}[a,sloped]
(3,1){Up}
(4,3){line A} ;
\tzlines[dashed](0,0)(1,1)(2,1)(3,0){line B}[r];
\end{tikzpicture}

**Shift**  You can move the connected line by specifying `<shift coor>` before the first coordinate or immediately before the option "<path name>" if it exists. (The empty shift option <> is not allowed.)
\begin{tikzpicture}
\helplines(4,3)
\tzlines[red,thick,\text{\textlangle},text=blue]
\begin{scope}[shift=(4,3)]
\tzlines[\text{\textlangle}=0,-1](0,1)
\tzlines(1,2)
\tzlines(3,1)
\tzlines(4,3)
\end{scope}
\tzlines[dashed](0,0)(1,1)(2,1)(3,0)
\end{tikzpicture}

Naming paths  You can name the path of \tzlines by specifying the option "\textless path name\textgreater" immediately before the first coordinate.

\begin{tikzpicture}
\helplines(4,3)
\tzlines[red,thick,\text{\textlangle},text=blue]
\begin{scope}[shift=(4,3)]
\tzlines[\text{\textlangle}=0,-1](0,1)\text{	extlangle}AA\textrangle(0,1)
\tzlines(1,2)
\tzlines(3,1)
\tzlines(4,3)
\end{scope}
\tzlines[dashed](0,0)(1,1)(2,1)(3,0)
\end{tikzpicture}

Extending the path  You can extend the path of \tzlines by writing Ti\TeX code in the last (even after the semicolon) optional argument \textless code.append\textgreater.

\begin{tikzpicture}
\helplines(4,3)
\tzlines[red,thick,\text{\textlangle},text=blue]
\begin{scope}[shift=(4,3)]
\tzlines(0,1)\text{	extlangle}Up\textgreater
\tzlines(1,2)\text{	extlangle}Down\textgreater
\tzlines(3,1)\text{	extlangle}Up\textgreater
\tzlines(4,3)\text{	extlangle}arc(0:90:1cm)\textgreater
\end{scope}
\tzlines[dashed](0,0)(1,1)(2,1)(3,0)
\end{tikzpicture}

You can close the path with a straight line by \textlangle cycle\textrangle.
11.5 \texttt{\textbackslash tzlines*: Relative coordinates: Semicolon version}

The \textit{plus version} \texttt{\textbackslash tzlines*} connects two or more points with connected line segments, but each coordinate (except the first one) is \textit{relative} to the previous coordinate.

\textit{Everything else is the same as in \texttt{\textbackslash tzlines}.} It is also required to enter a \textit{semicolon} to indicate when the coordinate iteration ends.

\begin{tikzpicture}
\helplines
\tzlines[red,thick,<->,text=blue]
(0,1){Up}
(1,2){Down}[a,sloped]
(3,1){Up}
(4,3){line A} ; %--cycle
\tzlines[dashed]
(0,0)(1,1)(2,1)(3,0){line B}[r] ; %--cycle
\end{tikzpicture}
12 Connecting Points

12.1 \tzto: Two points

\tzto connects two points with a straight or curved line, using TikZ’s to operation. So \tzto is more general than \tzline, which connects points only with a straight line.

\begin{verbatim}
% syntax: minimum
\tzto(<coor>)(<coor>)
% syntax: medium
\tzto[<opt>](<coor>){<text>}{<node opt>}(<coor>){<text>}{<node opt>}
% syntax: full
\tzto[<opt>][<shift coor>]<path name>" (coor){<text>}{<node opt>}(coor){<text>}{<node opt>}{code.append}
% defaults
[]<>"(m){above}{m}
\end{verbatim}

\tzto[dashed](1,1)(3,2) % works like:
\draw [dashed] (1,1) to (3,2);

**Line styles** You can control line styles with the first optional argument [<opt>].

\begin{verbatim}
% \tzto
\begin{tikzpicture}
\tdelimitersizes
\tzline(4,3)
\tzto[blue,thick,->](0,1)(3,3)
\tzto[out=0](0,0)(4,2)
\tzto[bend right,dashed](2,0)(4,1)
\end{tikzpicture}
\end{verbatim}

**Adding text** You can add text next to the line ([midway,above], by default) by specifying the options {<text>} and [<node opt>] in-between the two coordinates.

\begin{verbatim}
\tzto[->,bend right](1,1){A}[near start](3,2) % works like:
\draw [->,bend right] (1,1) to node [near start] {A} (3,2);
\end{verbatim}

\begin{verbatim}
\tzto[->,bend right](1,1){A}[near start](3,2){B}[right] % works like:
\draw [->,bend right] (1,1) to node [near start] {A} (3,2) node [right] {B};
\end{verbatim}

You can also add text at or around the last coordinate by the options {<text>} and [<node opt>] immediately after the last coordinate.

\begin{verbatim}
% \tzto: adding text
\begin{tikzpicture}
\tdelimitersizes
\tzline(4,3)
\tzto[blue,thick,->](0,1){line}[sloped](3,3)
\tzto[out=0](0,0){curve}[pos=.8](4,2){line B}[r]
\tzto[bend right,dashed](2,0)(4,1){line C}[red,r]
\end{tikzpicture}
\end{verbatim}
Shift  You can move the line by specifying the option `<shift coor>` before the first coordinate or immediately before the option "<path name>" if it exists. (The empty shift option <> is not allowed.)

```latex
\begin{tikzpicture}
\helplines(4,3)
\tzto[blue,thick,->]<0,1>(0,1){line}[sloped](3,3)
\tzto[out=0]<1,1>(0,0){curve}[pos=.8](4,2){line B}[r]
\tzto[bend right,dashed]<-1,0>(2,0)(4,1){line C}[red, r]
\end{tikzpicture}
```

Naming paths: Intersections  You can name the path of \tzto by specifying the option "<path name>" immediately before the first coordinate.

```latex
\begin{tikzpicture}
\helplines(4,3)
\tzto[bend right=45,blue,thick,->]"curveA"(1,3)(4,0)
\tzto[out=0]"curveB"(0,0)(4,2){line B}[r]
\% intersection point
\tzXpoint*[curveA]{curveB}
\end{tikzpicture}
```

Extending the path  You can extend the path of \tzto by writing TikZ code in the last optional argument `<code.append>`.

```latex
\begin{tikzpicture}
\helplines(4,3)
\tzto[bend right=45,blue,thick,->](1,3)(4,0)<--([turn]0:1cm) node [b] {extended}>
\tzto[out=0]
(0,0)(4,2)<--([turn]0:1cm) node [r] {extended}>
\end{tikzpicture}
```

12.2 \tzto+: Relative coordinates

The plus version \tzto+ uses the second coordinate relative to the first coordinate. *Everything else is the same as in \tzto.*

\begin{verbatim}
\tzto+(1,1)(3,2) \% works like:
\draw (1,1) to ++(3,2);
\end{verbatim}
12.3 \tztos: Multiple points: Semicolon version

\tztos takes an arbitrary number of coordinates as arguments to connect them by Ti\kZ’s to operation. Since this is a semicolon version, you need to enter a semicolon to indicate when the coordinate iteration ends.

% syntax
\tztos[<opt>]<shift coor>"<path name>"<coor1>..<text>..<node opt>..<code.append>
% defaults
[<opt>][<shift coor>]"<path name>"<coor1>..<text>..<node opt>..<code.append>

The quadruple <coor1>..<text>..<node opt> is the whole repeating pattern. Here, [<to opt>] is for the options of Ti\kZ’s to operation such as [bend right], [bend left], [bend left=<angle>], [out=<angle>,in=<angle>] and so on.

\tztos(0,0)(1,2)(3,1); \% works like:
\draw (0,0) to (1,2) to (3,1);

\tztos[blue] (0,0) [bend right] (1,2) [out=-60,in=45] (3,1); \% works like:
\draw[blue] (0,0) to [bend right] (1,2) to [out=-60,in=45] (3,1);

How to connect coordinates  You can use the options of Ti\kZ’s to operation to connect the coordinates with different types of curves.
Adding text You can add text next to lines or curves by specifying the options `{<text>}` and `[<node opt>]` in-between coordinates or after the option `[<to opt>]`, if it exists. You can also add text at or around the last coordinate by the last options `{<text>}` and `[<node opt>]`.

Shift You can move the line or curve of \tztos using the option `<shift coor>` before the first coordinate or immediately before the option "<path name>", if any. (The empty shift option <> is not allowed.)

name path for intersections You can name the path of \tztos by specifying "<path name>" immediately before the first mandatory coordinate.
Extending paths You can extend the path of \tztos from the last coordinate, by writing TikZ code in the last (after the semicolon) optional argument <code.append>.

12.4 \tztos+: Relative coordinates: Semicolon version

The plus version \tztos+ takes each coordinate (except the first coordinate) relative (with ++) to the previous coordinate.

Everything else is the same as in \tztos.

13 Filling Area

13.1 \tzpath: Semicolon version

\tzpath creates a path connecting an arbitrary number of coordinates, but it does not stroke the path. You can visualize the path with \tzpath[draw]. Since \tzpath is a semicolon version macro, you need to enter a semicolon ‘;’ to indicate where the coordinate iteration ends.
The main purpose of \tzpath is to fill an enclosed area with color or pattern. You can use \tzpath[fill] or \tzpath[pattern=<...>] to do it.

Path construction operation  (If you are not an experienced user of TikZ just skip this part.)

\tzpath is similar to \tztos, but it is more flexible in constructing paths. \tzpath allows you to choose how to construct a path using the [\textit{path style}] option in-between coordinates. Path extension operation can be selected from ‘--’, ‘to’, ‘|-’, ‘-|’, etc. You can use the first brace option \{\textit{path style}\} to change all the \textit{path style} in-between coordinates.

The default path style is ‘to’ and can also be changed by \settztpathstyle, like, for example, \settztpathstyle{--}. The effect remains valid until the end of \tikzpicture environment unless changed again.

You can extend the path of \tzpath by writing TikZ code in the last (after the semicolon) optional argument \texttt{<code.append>}. So \texttt{<--cycle>} closes the path with a straight line.
13.2 \texttt{\textbackslash tzpath*: Semicolon version}

The starred version \texttt{\textbackslash tzpath*} additionally fills the interior of \texttt{\textbackslash tzpath} with black!50 with \texttt{fill opacity}=.3 and \texttt{text opacity}=1, by default. \texttt{\textbackslash tzpath*} works like \texttt{\textbackslash tzpath[fill=black!50,fill opacity=.3,text opacity=1]}. You can change the defaults by \texttt{\textbackslash settzfillcolor} and \texttt{\textbackslash settzfillopacity}.

Filling the interior  You can optionally change the opacity of fill color using the Ti\textit{k}Z option \texttt{fill opacity}.

You can also change the fill opacity by specifying the last curly brace optional argument \{<\texttt{fill opacity}>\}, after the semicolon.
You can also use the macros \texttt{\settzfillcolor} and \texttt{\settzfillopacity} to change the defaults. The effect remains valid until the end of the \texttt{tikzpicture} environment, unless changed again.

```latex
\begin{tikzpicture}
\tzhelplines(4,3)
\settzfillcolor{red}
\tzpath*[blue](0,1)(1,2)[=](3,1)(4,3);
\tzpath*[< to [bend right=60] (0,1) >
\tzpath*[0,0][to[bend right]]
(1,1)
(2,1)[to[out=-135,in=45]]
(3,0); <--cycle>
\end{tikzpicture}
```

### 13.3 \texttt{\tzpath+} and \texttt{\tzpath**: Relative coordinates: Semicolon versions}

The \texttt{plus version} \texttt{\tzpath+} uses each coordinate (except for the first coordinate) relative (with ++) to the previous coordinate.

*Everything else is the same as in \texttt{\tzpath}.*

\texttt{\tzpath**} is simply the plus version of \texttt{\tzpath*}.

```latex
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline[red,very thick](0,3)(4,0)
\tzpath*[pattern=bricks,preaction={fill=brown}]
(0,0)(3,1)(-1,2)(-2,-1);
\tzpath*[pattern=north east lines,opacity=.5]<1,-1>(0,0)(3,1)(-1,2)(-2,-1);
\end{tikzpicture}
```

### 14 Curves

There are many ways to draw curves.

#### 14.1 Bézier curves

\texttt{\tzbezier} draws a Bézier curve with one or two control points.
14.1.1 \texttt{tzbezier}

% syntax: minimal
\tzbezier(<start-coor>)(cntl-coor>)(<last-coor>)
\tzbezier(<start-coor>)(cntl-coor>)(<cntl-coor>)(<last-coor>)
% syntax: full
\tzbezier{<draw opt>}(shift coor)"<name path>"{<start-coor>}(cntl-coor>)(<cntl-coor>)(<last-coor>)
{text}>[<node opt>]<code.append>
% defaults
[]<>"(
\begin{tikzpicture}
\tzhelplines(4,4)
\tzbezier[red](0,3)(1,2)(2,4)(4,3)
\tzbezier[red](0,3)(1,2)(2,4)(4,3);
\end{tikzpicture}

Control points You can specify one or two control points, (\texttt{<cntl-coor>}). So \texttt{tzbezier} accepts three or four coordinates as arguments.

% three coordinates: one control point
\tzbezier(0,1)(1,0)(4,3) % works like:
\draw (0,0) ..controls (1,0).. (4,3);

% four coordinates: two control points
\tzbezier(0,1)(1,0)(2,4)(4,3) % works like:
\draw (0,0) ..controls (1,0) and (2,4).. (4,3);

The style \texttt{tzshowcontrols} displays the control points by drawing dotted lines, by default. You can also change the dotted line style, like \texttt{tzshowcontrols={dashed,green}}.

% \texttt{tzbezier}: \texttt{tzshowcontrols}
\begin{tikzpicture}
\tzbezier[blue,tzshowcontrols](0,2)(1,.5)(4,0)
\tzdots*[blue](0,2)(1,.5)(4,0);
\tzbezier[red,tzshowcontrols={green,dashed}]
(0,3)(1,2)(2,4)(4,3)
\tzdots*[red](0,3)(1,2)(2,4)(4,3);
\end{tikzpicture}

Adding text You can add text next to the curve or at the last coordinate by specifying the options \texttt{(<text>)} and \texttt{[<node opt>]} immediately after the last coordinate.
Shift  You can move the curve of \texttt{\textbackslash tzbezier} by specifying the option \texttt{<shift coor>} before the first coordinate or immediately before the option "\texttt{<path name>}", if any. The empty shift option <> is not allowed.

name path  You can name the path of \texttt{\textbackslash tzbezier} by specifying the option "\texttt{<path name>}' immediately before the first coordinate.

Extending paths  You can extend the path of \texttt{\textbackslash tzbezier} from the last coordinate, by writing \texttt{TikZ} code in the last optional argument \texttt{<code.append>}. 

\begin{tikzpicture}
\begin{scope}
\tzhelplines(4,4)
\tzbezier[blue]{0,2}(1,5)(4,0){curve}[draw,black,r]
\tzbezier[red]{0,3}(1,2)(2,4)(4,3){bezier}[near end,a]
\end{scope}
\end{tikzpicture}

\begin{tikzpicture}
\begin{scope}
\tzhelplines(4,4)
\tzbezier[blue]{0,2}(1,5)(4,0){curve}[draw,black,r]
\tzbezier[blue,dashed]{1,1}{0,2}(1,5)(4,0){curve}[r]
\tzbezier[red]{0,3}(1,2)(2,4)(4,3){bezier}[near end,a]
\tzbezier[red,dashed]{0,-1}{0,3}(1,2)(2,4)(4,3){bezier}[blue,near end,a]
\end{scope}
\end{tikzpicture}

\begin{tikzpicture}
\begin{scope}
\tzhelplines(4,4)
\tzbezier[blue,dashed,thick]{1,1}{0,2}(1,5)(4,0){curve}[r]
\tzbezier[red,dashed,thick]{0,-1}{0,3}(1,2)(2,4)(4,3){bezier}[blue,near end,a]
\xpoint{curve}{bezier}{X}{X}{-90} \% <angle>
\end{scope}
\end{tikzpicture}

\begin{tikzpicture}
\begin{scope}
\tzhelplines(4,4)
\tzbezier[red]{0,3}(1,2)(2,4)(4,3){bezier}[near end,a]
\tzbezier[blue,->]{0,2}(1,5)(4,0)
\node [above] at (B) {};
\end{scope}
\end{tikzpicture}

\begin{tikzpicture}
\begin{scope}
\tzhelplines(4,4)
\tzbezier[red]{0,3}(1,2)(2,4)(4,3){bezier}[near end,a]
\node [below] at (A) {};
\end{scope}
\end{tikzpicture}
### 14.1.2 `\tzbezier+`: Relative coordinates

For the *plus version* `\tzbezier+(A)(B)(C)`, (B) and (C) are relative to (A), as in TikZ.

For `\tzbezier+(A)(B)(C)(D)`, (B) and (D) are relative to (A), and (C) is relative to (D), as in TikZ.

\begin{verbatim}
% three coordinates
\tzbezier+(0,1)(1,-1)(4,3) \% works like:
\draw (0,1) ..controls +(1,-1).. ($(0,1)+(4,3)$);
\end{verbatim}

\begin{verbatim}
% four coordinates
\tzbezier+(0,1)(1,-1)(-2,1)(4,3) \% works like:
\draw (0,1) ..controls +(1,-1) and +(-2,1).. ($(0,1)+(4,3)$);
\end{verbatim}

\begin{verbatim}
% `\tzbezier+
\begin{tikzpicture}
\tzhelplines(4,4)
\tzbezier+(0,3)(1,-1)(-2,1)(4,0){bezier}[near end,a]
  < to [bend right] ++(-1,-1) node [below] {A} >
\tzbezier+[blue,->](0,2)(1,-1.5)(4,-2)
  < |- ++(-2,1) node [above] {B} >
\end{tikzpicture}
\end{verbatim}

### 14.2 Parabolas

#### 14.2.1 `\tzparabola`

`\tzparabola` draws a parabola going through specified coordinates. It accepts two or three coordinates. In the case of three coordinates, the parabola bends at the second coordinate.

\begin{verbatim}
% two coordinates
\tzparabola(0,0)(3,2) \% works like:
\draw (0,0) parabola (3,2);
\end{verbatim}

\begin{verbatim}
% three coordinates
\tzparabola(0,0)(1,1)(3,2) \% works like:
\draw (0,0) parabola bend (1,1) (3,2);
\end{verbatim}

\begin{verbatim}
% syntax: minimal
\tzparabola(<coor>)(<coor>)
\tzparabola(<coor>)(<coor>)(<coor>)

% syntax: full
\tzparabola[<opt1>]<shift coor>"<path name>"
  <(coor)>(<coor>)(<coor>){<text>}[<node opt>]<code.append>

% default
[]<>""(<>)(<>){<>}
\end{verbatim}
Parabolas  \tzplot draws the graph of a quadratic function $f(x) = ax^2 + bx + c$ for appropriate values of $a$, $b$, and $c$.

Adding text  You can add text at or around the last coordinate by specifying the options \{<text>\} and \[<node opt>\] immediately after the last coordinate.

Shift  You can move the parabola by specifying the option \<shift coor\> before the first coordinate or immediately before the option \"<path name>\", if any. The empty shift option <> is not allowed.

name path  You can name the path of \tzparabola by specifying the option \"<path name>\" immediately before the first coordinate.
Extending paths  You can extend the path of \texttt{tzparabola} from the last coordinate by writing Ti\textit{k}Z code in the last optional argument \texttt{<code.append>.

\begin{tikzpicture}
\draw[blue,->] (0,0) parabola bend +(2,2) (4,0);
\end{tikzpicture}

14.2.2 \texttt{tzparabola+}: Relative coordinates

For the \textit{plus version} \texttt{tzparabola+}, the second and the third coordinates are relative to the first coordinate.\newline
\textit{Everything else is the same as in \texttt{tzparabola}.}

\begin{tikzpicture}
\draw (0,0) parabola bend +30 (4,0);
\end{tikzpicture}

14.3 \texttt{tzplotcurve}

See Section 9.6 on page 58, for more details on \texttt{tzplotcurve}. 
14.4 \tzto, \tztos

See Section 12.1 on page 74, for more details on \tzto.
See Section 12.3 on page 76, for more details on \tztos.

14.5 \tzfn

With \tzfn, you can plot functions such as \( f(x) = \frac{1}{2}(x - 1)^3 + 1 \), \( g(x) = \sin x \), \( h(x) = \sqrt{x} - 1 \), and so on. See Section 20.1 on page 110, for more details on \tzfn.

15 Polygons and Circles

15.1 \tzpolygon: Semicolon version

\tzpolygon connects an arbitrary number of coordinates to draw a polygon, a closed figure. \tzpolygon is equivalent to a closed \tzlines. Since \tzpolygon is a *semicolon version*, you need to enter a *semicolon* to indicate when the coordinate repetition ends.

\begin{tikzpicture}
\helplines (4,3)
\tzpolygon[fill,blue,auto]
(0,1)(Side A)[sloped,red]
(1,3)(B)
(2,3)
(3,2)(Side D)[swap,sloped]
(1,0);
\end{tikzpicture}

You can also move the polygon by specifying the option `<shift coor>` before the first coordinate. The empty shift option `<>` is not allowed.
15.2 \texttt{\texttt{polygon}*}: Semicolon version

\texttt{\texttt{polygon}*} paints the interior of the polygon with black!50 with fill opacity=.3 but with text opacity=1, by default.

\begin{tikzpicture}
\helplines(4,3)
\polygon[blue,auto]
(0,1)(Side A)[sloped,red]
(1,3)(B)
(2,3)
(3,2)(Side D)[swap,sloped]
(1,0);
\polygon[auto,dashed]<1,-.5>
(0,1)(1,3)(2,3)(C)(3,2)(1,0);
\end{tikzpicture}

You can change the fill opacity by specifying the the last curly brace option \{<fill opacity>\} immediately after the semicolon.

\begin{tikzpicture}
\helplines(4,3)
\polygon[draw=blue,auto]
(0,1)(Side A)[sloped,red]
(1,3)(B)
(2,3)
(3,2)(Side D)[swap,sloped]
(1,0);
\polygon[green,auto,dashed,text=black]<1,-.5>
(0,1)(1,3)(2,3)(C)(3,2)(1,0); .7
\end{tikzpicture}

You can also change the defaults using \texttt{\texttt{settzfillcolor}} and \texttt{\texttt{sttzfillopacity}}.

15.3 \texttt{\texttt{polygon}+} and \texttt{\texttt{polygon}++}: Relative coordinates: Semicolon version

The plus version \texttt{\texttt{polygon}+} uses each coordinate (except the first one) relative (with ++) to the previous coordinate.

\texttt{\texttt{polygon}++} is just a plus version of \texttt{\texttt{polygon}*.}
\tzpolygon*(1,1)(2,2)(3,1)(4,3); % works like:
\draw (1,1) -- ++(2,2) -- ++(3,1) -- ++(4,3) -- cycle;

\tzpolygon+[dashed]"AA"(1,1)(2,2){A}(3,1){B}[below]; % works like:
\draw [dashed,name path=AA] (1,1) -- ++(2,2) -- ++(3,1) node {A} -- ++(4,3) node [below] {B} -- cycle;

% \tzpolygon: shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzpolygon*[blue,auto]
(0,1)(Side A)[sloped,red]
(1,2){B}
(1,0)
(1,-1)(Side D)[swap,sloped]
(-2,-2);
\tzpolygon*[green,auto,dashed,text=black]<1,-.5>(0,1)(1,2)(1,0)(C)(1,-1)(-2,-2); \end{tikzpicture}

\section{15.4 \tzframe and \tzframe*}
\tzframe accepts two coordinates draws a rectangle.

% syntax: minimum
\tzframe(\<coor\>)(\<coor\>)
% syntax: full
\tzframe[\<opt\>](\<shift coor\>\"<path name>"\<coor1\>\<coor2\>\langlecode.append\>
% defaults
[]"==\langle\n\rangle\langle\>

\tzrectangle is an alias of \tzframe.

\tzframe(0,1)(3,2) % works like:
\draw (0,1) rectangle (3,2);

The plus version \tzframe+ uses the second coordinate as the coordinate relative (with ++) to the first. \tzrectangle+ is an alias of \tzframe+.

\tzframe+(0,1)(3,2) % works like:
\draw (0,1) rectangle ++(3,2);
The starred version \(\texttt{tzframe*}\) fills the interior with black!50 with \texttt{fill opacity=.3}, by default. (\(\texttt{tzrectangle*}\) is an alias of \(\texttt{tzframe*}\).)

\(\texttt{tzframe+}\) has also its starred version \(\texttt{tzframe**+}\). (\(\texttt{tzrectangle**+}\) is an alias of \(\texttt{tzframe**+}\).)

With the starred versions, you can change the fill opacity using the last option \{<fill opacity>\}.

You can move \(\texttt{tzframe}\) and its variants by specifying the option \texttt{<shift coor>} immediately before the first mandatory coordinate. The empty shift option \texttt{<>} is not allowed.

You can extend the path by the option \texttt{<code.append>}. 

% \texttt{\textbackslash tzframe, \textbackslash tzframe+}
\begin{tikzpicture}
\zhelplines(4,3)
\tzframe(0,0)(3,2)
\tzframe*[blue,rounded corners=2mm](1,3)(1,-2)
\end{tikzpicture}

%%

% syntax
\texttt{\textbackslash tzframe*}[<opt>]<shift coor>"<path name>"
\begin{tikzpicture}
\zhelplines*(4,4)
\tzframe*[fill=black!50,fill opacity=.3]<"(m)(m)"(<m>)<.3>
\end{tikzpicture}

% defaults
*[[fill=black!50,fill opacity=.3]<>"(<m>)(<m>)\{.3\}<\]]

% \texttt{\textbackslash tzframe}(+)\texttt{\textbackslash tzrectangle}(+)
\begin{tikzpicture}
\zhelplines*(4,4)
\tzframe*(0,0)(3,2)
\tzframe*[fill=green,rounded corners=2mm](1,3)(1,-2){.7} %%
\end{tikzpicture}

% \texttt{\textbackslash tzframe*}(+)\texttt{\textbackslash tzrectangle*}
\begin{tikzpicture}
\zhelplines*(4,4)
\tzcoors(2,2)(A)(3,1)(B);
\tzframe*[blue,dashed]<1,1>(0,0)(3,2)
\tzframe*[blue,dashed]<1,1>(0,0)(3,2)
\tzframe*[blue,rounded corners=2mm](A)(B)
\tzframe*[red,rounded corners=2mm]<-.5,-.5>(A)(B)
\tzframe*[green,rounded corners=2mm]<1,1>(A)(B)
\end{tikzpicture}

% \texttt{\textbackslash tzframe*}, \texttt{\textbackslash tzrectangle*}: <code.append>
\begin{tikzpicture}
\zhelplines*(4,4)
\tzcoors(2,2)(A)(3,1)(B);
\tzframe*[blue,even odd rule]<.5,0.5>(0,0)(3,2)
\tzframe*[red,even odd rule]<.5,0.5>(0,0)(3,2)
\end{tikzpicture}
15.5 \texttt{\texttt{tzcircle}} and \texttt{\texttt{tzcircle*}}

\texttt{\texttt{tzcircle}} draws a circle around a specified coordinate with a specified radius. The coordinate and the radius are mandatory.

% syntax
\texttt{\texttt{tzcircle}[<opt>]<shift coor>"<path name>"<(coor)>(<radius>)<code.append>
% defaults
[]<"<(m)>(m)">

\texttt{\texttt{tzcircle}(0,0)(1cm) \% works like:
\begin{tikzpicture}
\draw (0,0) circle (1cm);
\end{tikzpicture}}

The starred version \texttt{\texttt{tzcircle*}} fills the interior with \texttt{black!50} with \texttt{fill opacity=.3}, by default. You can change the fill opacity using the last option \texttt{<fill opacity>}.

% syntax
\texttt{\texttt{tzcircle*}[<opt>]<shift coor>"<path name>"}
\begin{tikzpicture}
\draw (0,0) circle (1cm);
\end{tikzpicture}
% defaults
*<fill\texttt{\texttt{=black!50,fill opacity=.3}}<"<(m)>(m)">.3<>

\begin{tikzpicture}
\draw (2,2) circle (1cm);
\draw[blue, dashed] (3,1) .. controls (2,1) .. (1);
\draw[green] (3,.5) circle (.5cm);
\end{tikzpicture}

You can move the circle by specifying the option \texttt{<shift coor>} immediately before the center coordinate. The empty shift option \texttt{<>} is not allowed.

% \texttt{\texttt{tzcircle*}}: shift
\begin{tikzpicture}
\draw (2,2) circle (1cm);
\draw[blue, dashed] (3,1) .. controls (2,1) .. (1);
\draw[green] (3,.5) circle (.5cm);
\end{tikzpicture}

You can extend the path by the option \texttt{<code.append>}.  

\begin{tikzpicture}
\draw (2,2) circle (1cm);
\draw[blue, dashed] (3,1) .. controls (2,1) .. (1);
\draw[green] (3,.5) circle (.5cm);
\end{tikzpicture}
\begin{tikzpicture}
\helplines(4,4)
\coord(2,2)(A)(3,1)(B);
\circle*[blue,even odd rule](2,2)(1)
\circle(2,2) circle (1.5)
\circle[fill=green,even odd rule](3,1)(1)
\circle(3,1) circle (.7)
\end{tikzpicture}

15.6 \texttt{\textbackslash tzellipse} and \texttt{\textbackslash tzellipse}*  

\texttt{\textbackslash tzellipse} draws an ellipse around a specified coordinate with the specified x-radius and y-radius.

% syntax
\texttt{\textbackslash tzellipse}[^{<opt>}]<shift coor>"<path name>"<coor> (<x and y radius>)<code.append>
% defaults
\texttt{[]}<>(<>)<>()

\texttt{\textbackslash tzoval} is an alias of \texttt{\textbackslash tzellipse}.

\texttt{\textbackslash tzellipse}(0,0)(1 and .5) % works like:
\texttt{\textbackslash draw} (0,0) ellipse (1 and .5);

The starred version \texttt{\textbackslash tzellipse}* fills the interior with black!50 with fill opacity=.3, by default. You can change the fill opacity using the last option \{<fill opacity>\}.

% syntax
\texttt{\textbackslash tzellipse}*[^{<opt>}]<shift coor>"<path name>"
\texttt{\textbackslash (coor) (<x and y radius>) (<fill opacity>)}<code.append>
% defaults
*[^{fill=black!50,fill opacity=.3}]<>()<()><()>{.3}<>

% \texttt{\textbackslash tzellipse(*)}
\begin{tikzpicture}
\helplines(4,3)
\ellipse(2,2)(1.5 and 1)
\ellipse[fill=red](3,1)(1cm and .5cm)
\ellipse*[blue](2,1)(1 and 1.5){.5}
\ellipse*[green](3,.5)(1cm and .5cm)
\end{tikzpicture}

You can move the ellipse by specifying the option \texttt{<shift coor>} immediately before the mandatory coordinate. The empty shift option <> is not allowed.
You can extend the path by the option `<code.append>`. Note also that \tzoval* is an alias of \tzellipse*.

16 Arcs and Wedges

16.1 \tzarc(\): Centered arcs

16.1.1 Arcs

\tzarc draws an arc around a specified center coordinate.

% syntax: minimum
\tzarc(<coor>)(<ang1:ang2:radius>)
% syntax: full
\tzarc[<opt>][<shift coor>]<path name>*
  (<coor>)(<ang1:ang2:radius>{<text>}[<node opt>]<code.append>
% defaults
{}"""(m){m}{}<>}

\tzarc(1,1)(30:120:1) % works like:
  \draw (1,1) ++(30:1) arc (30:120:1);

The swap version \tzarc' switches its drawing direction from counterclockwise to clockwise and vice versa.

\tzarc'(1,1)(30:120:1) % works like:
  \draw (1,1) ++(30:1) arc (30:120-360:1);
You can add text along the arc by specifying the options \{<text>\} and \[<node opt>\] immediately after the two mandatory arguments.

You can move arcs by specifying the option \texttt{<shift coor>} just before the center coordinate.

You can also extend the path of \texttt{tzarc} by specifying the last option \texttt{<code.append>} with \textit{TikZ} code written in it. For example, \texttt{<--cycle>} makes a closed path.

16.1.2 Elliptical arcs

\texttt{tzarc} draws an elliptical arc if you specify \texttt{x-radius} and \texttt{y-radius}.
16.2 \tzarcfrom(')': Arcs as in TikZ

\tzarcfrom draws an arc starting from a specified point, like \texttt{TikZ} does.

% syntax: minimum
\tzarcfrom(<coor>)(<angA:angB:radius>)
% syntax: full
\tzarcfrom[<opt>]<shift coor>"<path name>"
    (<coor>)(<angA:angB:radius>)<text>[<node opt>]<code.append>
% defaults
[]<>"{}<>"
16.3 \texttt{\textbackslash tzarcsfrom}: Connected arcs: Semicolon version

The macro \texttt{\textbackslash tzarcsfrom} (i.e. \texttt{\textbackslash tzarcs + from}) accepts an arbitrary number of parenthesis arguments in the form of \texttt{(<angA:AngB:radius>)} following the start coordinate. You need to enter a \texttt{semicolon} to indicate when the repetition ends.

\begin{verbatim}
% syntax: minimum
\texttt{\textbackslash tzarcsfrom}(<start coor>)
 \texttt{\textbackslash (angA:angB:radius)}>..repeated..\texttt{\textbackslash (angA:angB:radius)} ;
% syntax: full
\texttt{\textbackslash tzarcsfrom}[<opt>]<shift coor>"<path name>"
 (\texttt{<start coor>})(\texttt{<angA:angB:radius>})\{\texttt{<text>}}[\texttt{<node opt>}]..repeated..\}{}[] ; \texttt{\textbackslash code.append}
% defaults
[]<>""<(m)\{\}..repeated..()\}{} ; <\texttt{code.append}>
\end{verbatim}

\begin{verbatim}
% \texttt{\textbackslash tzarcsfrom}: adding text, \texttt{\textbackslash code.append}
\begin{tikzpicture}
\zhelplines(4,4)
\tzcoor*(3,2)(A)
\tzarcsfrom[->,auto](A)
 (0:180:1)(1)[midway]
 (180:360:1.2)(2)[midway]
 (0:180:1.4){3}[midway,swap]
 (180:360:1.6){4}[midway,swap];
 < node [right,blue] {End!} >
\end{tikzpicture}
\end{verbatim}

\begin{verbatim}
% \texttt{\textbackslash tzarcsfrom}: shift
% flag: step 1
\begin{tikzpicture}
\edef\x{atan(2/3)}
\tzarcsfrom[red,->](0,0)
 (-\x:-\x-180:1){red 1}[midway];
\tzarcsfrom[blue,->](0,0)
 (180-\x:-\x:1){blue 1}[midway];
\tzarcsfrom[dashed,-><.5,5>(0,0)
 (180-\x:-\x:1){blue 1}[midway];
\end{tikzpicture}
\end{verbatim}

\begin{verbatim}
% flag: step 2
\begin{tikzpicture}
\edef\x{atan(2/3)}
\tzarcsfrom[red](0,0)
 (-\x:-\x-180:1){midway}
 (-\x+180:-\x:2){midway};
\tzarcsfrom[blue](0,0)
 (180-\x:-\x:1){midway}
 (-\x:-\x-180:2){midway};
\end{tikzpicture}
\end{verbatim}
16.4 Wedges

16.4.1 \tzwedge('

\tzwedge draws a wedge around a specified center coordinate. \tzwedge works similarly to \tzarc, but it forms a closed path from the center coordinate.

The macro \tzwedge' is the swap version of \tzwedge. It switches the drawing direction from counterclockwise to clockwise and vice versa.
16.4.2 \tzwedge\('\)

The starred version \tzwedge\* works just like \tzwedge, but it fills wedges with black!50 with fill opacity=.3 and fill opacity=1, by default. You can change the defaults with \settztfillcolor and \settztfillopacity. You can also change the fill opacity by specifying the last optional argument \{<fill opacity>\}.

\begin{tikzpicture}
\tzcoor*(2,1)(A)(3pt)
\zhelplines(4,3)
\tzarc[->,green](A)(30:120:1.5)
\tzarc'[->,green](A)(0:135:1.3)
\tzwedge*[fill](A)(-60:240:1)
\end{tikzpicture}

\begin{tikzpicture}
\tzcoor*(2,1)(A)
\zhelplines(4,3)
\tzarc[->,lightgray]<.1,.1>(A)(30:120:1.3) % shift
\tzarc'[->,lightgray](A)(0:135:1.3)
\tzwedge*[fill]<2,0>(A)(-60:240:1) % shift
\end{tikzpicture}

\tzwedge\*' is the swap version of \tzwedge\*. 

\begin{tikzpicture}
\begin{scope}[->,>=stealth]
\tzcoor*(2,1)(A)
\zhelplines(3,3)
\tzarc[->,very thick,blue](1,1)(30:120:1.5)
\tzarc[-,very thick,blue,fill=red](1,1)(30:120:1)
\tzwedge'[dashed](1,1)(30:120:1.5){clockwise}[pos=.45]
\tzwedge*[dashed,fill=green](1,1)(-90:180:1)
\tzdot*(1,1)
\end{scope}
\end{tikzpicture}
% \texttt{\texttt{tzwedge}(\texttt{'}): shift, fill opacity}
\begin{tikzpicture}
\tzdot*(1,1)
\zhelplines(3,3)
\tzwedge[blue,very thick](1,1)(30:120:1.5)
\tzwedge[red,very thick]<.5,.5>(1,1)(30:120:1.5)
\tzwedge'[dashed](1,1)(0:135:1.5)
\tzwedge*[fill=blue](1,1)(300:240:1){B}[b]
\tzwedge*[blue]<2,0>(1,1)(300:240:1){B}[b]{.7}
\end{tikzpicture}
Part IV
Plotting Graphs

17 Axes

17.1 Draw axes

17.1.1 \taxes

Basically, \taxes(<x1,y1>)(<x2,y2>) draws the x axis from <x1> to <x2> and the y axis from <y1> to <y2>. The coordinate (<x1,y1>) represents the origin and (<x2,y2>) represents the opposite corner of the rectangle formed by the two coordinates.

\taxes takes only one coordinate (<x2,y2>) as a mandatory argument, in which case the coordinate (<x1,y1>) is considered as (0,0).

% syntax: minimal
\taxes(<x2,y2>{<x-axis name>}{<y-axis name>}
% syntax: full
\taxes[opt]{<x-shift,y-shift>}(<x1,y1>)(<x2,y2>)
{<x-axis name>}{<node opt>}{<y-axis name>}{<node opt>}

% defaults
[-1]<0,0>(0,0){cm}{right}{above}
% arguments
[#1]: line style, arrow type (for x-axis & y-axis)
[#2]: axes shift coor  % axes intersect at (#2)
(#3): (x1,y1)  % origin: if omitted, regarded as (0,0)
(#4): (x2,y2)  % opposite corner: mandatory
(#5): x-axis name
[#6]: x-axis name option % node option
(#7): y-axis name
[#8]: y-axis name option % node option

Here, (cm) stands for a mandatory argument.

% \taxes
\begin{tikzpicture}[scale=.7]
\tzhelplines(3,3)
\taxes(3,3){x}{y}
% \taxes: shift
\end{tikzpicture}

Shift  By default, the x and y axes intersect at (0,0). Specifying the option <x-shift,y-shift> moves the axes to intersect at (<x-shift,y-shift>).

% \taxes: shift
\begin{tikzpicture}[scale=.5]
\tzhelplines(-1,-1)(7,7)
\tshoworigin*
\taxes[thick,blue]<1,2>(7,7){x,y}
\taxes[-,dashed](7,7){$X$}[below]{$Y$}[left]
\taxes<6,1>{6,1}({$a$}[left]{$b$}[left]}
\end{tikzpicture}
17.1.2 \texttt{\textbackslash{}tzaxes*}

The starred version \texttt{\textbackslash{}tzaxes*} sets the current state to a bounding box when the macro \texttt{\textbackslash{}tzaxes} execution is complete. It is recommended for you to use \texttt{\textbackslash{}tzaxes*} as the first graphics command in \texttt{tikzpicture} environment or before any larger graphics.

\begin{tikzpicture}[scale=.5]
\tzaxes*(8,5){\(x\)}{\(f(x)\)} % bounding box
\tzhelplines(-2,-1)(10,8)
\tzto[out=90,in=-135,dashed]{(-2,8)}{(12,-2)}
\tzbezier[blue]{(-1,-1)(3,-2)(7,12)(10,10)}
\end{tikzpicture}

17.2 \texttt{\textbackslash{}tzaxisx} and \texttt{\textbackslash{}tzaxisy}

\texttt{\textbackslash{}tzaxisx} draws only the \(x\) axis.

% syntax
\texttt{\textbackslash{}tzaxisx}[<opt>]<y-shift>{<from>}{<to>}{<x-axis name>}[<node opt>]
% defaults
[-]00<(<m>){(<m>}{[right]
% arguments:
[#]1: line style, arrow type (for x-axis)
<#2>: y-shift of x-axis
{#3}: x-axis starts from \texttt{\%\% mandatory
{#4}: x-axis runs to \texttt{\%\% mandatory
{#5}: x-axis name
[#6]: x-axis name option \texttt{\%\% node option

\texttt{\textbackslash{}tzaxisy} draws only the \(y\) axis.

% syntax
\texttt{\textbackslash{}tzaxisy}[<opt>]<x-shift>{<from>}{<to>}{<y-axis name>}[<node opt>]
% defaults
[-]00<(<m>){(<m>}{[above]
% arguments:
[#]1: line style, arrow type (for y-axis)
<#2>: x-shift of y axis
{#3}: y-axis starts from \texttt{\%\% mandatory
{#4}: y-axis runs to \texttt{\%\% mandatory
{#5}: y-axis name
[#6]: y-axis name option \texttt{\%\% node option

\begin{tikzpicture}[scale=.5]
\tzshworigin
\tzhelplines(5,5)
\tzaxis[blue]{-1}(5){\(xx\)}
\tzaxis[red] {-1}(5){\(yy\)}{green}
\end{tikzpicture}
\begin{tikzpicture}[scale=.5]
\helplines(7,7)
\showorigin\axes(7,7)\begin{scope}[xscale=.5,yscale=-.5]
\axisx{-1}{7}{$xx'$} [near end,below]
\axisy{-1}{7}{$yy'$} [midway,sloped,auto]
\axisx[-,dashed,red]<2>{-1}{7}{$xx$}
\axisy[-,dashed,blue]<3>{-1}{7}{$yy$}
\end{scope}
\end{tikzpicture}

\section{Display the origin}
\showorigin prints ‘0’ (approximately) at the bottom left of the origin \((0,0)\), by default.

All arguments of \showorigin are optional.

\begin{tikzpicture}[scale=.45]
\helplines\showorigin\axes(8,6)\showorigin<2,1>{O_2}{red}\end{tikzpicture}

You can change the text by specifying the curly brace option \{<text>\}, like, for example, \showorigin{$O$}. You can also change the coordinate of origin by the option \{<origin>\}. Specifying the option \{shift coor\} also moves the origin.

\begin{tikzpicture}[scale=.45]
\helplines\showorigin\axes<-$1,-1,2,1$(8,6)\showorigin<2,1>{O_2}{red}\end{tikzpicture}

\section{\showorigin*}
\showorigin* prints a node dot at the origin with no text by default. Internally the dot is processed by \dot*. All arguments are optional.

\begin{tikzpicture}[scale=.45]
\helplines\showorigin*<-$0,0$(8,6)\showorigin*[->]<$0,0$>{O_2}\end{tikzpicture}
You can add text with the option \{<text>\}. The default size of the dot is 2.4pt, and it can be changed with the last option \{<dot size>\}. You can change the dot style using the first optional argument \[<dot opt>\]. You can also move the dot by specifying the option \{<shift coor>\}.

\begin{tikzpicture}
\helplines(-2,-1)(6,4)
\showorigin*[tikzpicture]
\axes(-2,-1)(6,4)
\end{tikzpicture}

Remark: For \showorigin*, text for the origin and the dot are placed independently. In other words, the position of node text does not depend on the size of a node dot. (In fact, the node text for the origin should look good with the ‘ticks labels’, so it was not designed as a label for the node dot. This also means that the origin text cannot be positioned by an \{angle\}.)

\begin{tikzpicture}[scale=.5]
\helplines(-1,-1)(7,7)
\showorigin*[blue]{o_1}[red](5pt)
\axes(-1,-1)(7,7){x}{y}
\axes<6,4>(6,4)(1,1){x'}{y'}[left]{y'}[below]
\showorigin*[fill=none]<6,4>{O'}[blue,ar](3pt)
\end{tikzpicture}

17.4 \texttt{tzaL(')}: L-type axes
\tzaL is similar to \texttt{tza}, but it draws only the ‘L’ type axes with \{(x1,y1)\} as the origin and \{(x2,y2)\} as the opposite corner of the rectangle. Those two coordinates are mandatory.

\begin{tikzpicture}[scale=.5]
\helplines(-2,1)(6,5)
\showorigin*[blue]{o_1}[red](5pt)
\axes(-2,-1)(6,5){x}{y}
\zticks{1,...,5}{1,...,4}
\end{tikzpicture}
The swap version \texttt{\textbackslash tzaxesL'} swaps $(<x_1,y_1>)$ and $(<x_2,y_2>)$. That is, $\texttt{\textbackslash tzaxesL'}(A)(B)$ is equivalent to $\texttt{\textbackslash tzaxesL}(B)(A)$.

The option \texttt{\textbackslash shift coor} moves the whole L-type axes. The empty option \texttt{<>} is not allowed.

\begin{verbatim}
% \texttt{\textbackslash tzaxesL, \textbackslash tzaxesL'}
\begin{tikzpicture}[scale=.5]
\zhelplines(8,7)
\tzshoworigin
\tzaxes(8,7)\{\$x\}$\{\$y\}$
\tzaxesL[red,thick](2,2)(6,5)\{\$a\}$\{\$b\$
\tzaxesL'[blue,dashed,\textarrow](2,2)(6,5)\{\$c\}$\{\$d\$
\tzaxesL'[\textarrow](7,1)(4,7)\{\$m\}$\{\textdraw\textcircle\}
\end{tikzpicture}
\end{verbatim}

\begin{verbatim}
% \texttt{\textbackslash tzaxesL, \textbackslash tzaxesL': shift}
\begin{tikzpicture}[scale=.5]
\zhelplines(8,7)
\tzshoworigin
\tzaxes(8,7)\{\$x\}$\{\$y\}$
\tzaxesL[red,thick](2,2)(6,5)\{\$a\}$\{\$b\$
\tzaxesL[red,thick]<.5,.5>(2,2)(6,5)\{\$a'\}$\{\$b'\$
\tzaxesL'[blue,dashed,\textarrow]<-1,-1>(2,2)(6,5)\{\$c'\}$\{\$d'\$
\end{tikzpicture}
\end{verbatim}

18 Ticks

18.1 \texttt{\textbackslash tzticks}: Tick labels

By default, \texttt{\textbackslash tzticks} prints tick labels and draws zero length tick marks, i.e. from (0pt) to (0pt).

\begin{verbatim}
% syntax: minimal
\tzticks{<x-ticks pos>}{<y-ticks pos>}
% syntax: medium
\tzticks{<x-from:x-to>}{<x-ticks pos/labels>}[<node opt>]
{<y-from:y-to>}{<y-ticks pos/labels>}[<node opt>]
% syntax: full
\tzticks{<opt>}{<x-shift,y-shift>}
{<x-from:x-to>}{<x-ticks pos/labels>}[<node opt>]
{<y-from:y-to>}{<y-ticks pos/labels>}[<node opt>]
% defaults
[]<0,0>(Opt:0pt){\m}\{text height=1.25ex,text depth=.25ex,below}(Opt:0pt){\[]}[left]
\end{verbatim}

Tick labels Internally, \texttt{\textbackslash tzticks} uses TikZ's \texttt{foreach} operation. So you need to provide comma separated lists to print tick labels. If only one comma separated list is specified, it is for $x$ tick labels.
You can change the numbered labels to a different format with slashes and other text, as follows: \texttt{<number>/<other text>}.

**Tick marks** To prints tick marks you need to specify \texttt{(x-from:x-to)} for \textit{x} ticks and/or \texttt{(y-from:y-to)} for \textit{y} ticks. (The default is \texttt{(0pt:0pt)}.)

**Shift** The position of tick labels does not depend on the length of the tick marks. To change the position, use the option \texttt{<x-shift,y-shift>}, where \textit{x-shift} is for \textit{x}-ticks and \textit{y-shift} is for \textit{y}-ticks.
18.2 \tzticks*: Tick marks

The starred version \tzticks* always ignores all tick labels and draws tick marks from 0pt to 3pt, by default.

% syntax: minimal
\tzticks*{<x-ticks pos>}{<y-ticks pos>}
% syntax: medium
\tzticks*{<x-from:x-to>}{<x-ticks pos>}{<y-from:y-to>}{<y-ticks pos>}
% syntax: full
\tzticks*[<opt>]<x-shift,y-shift>{<x-ticks pos>}{<y-from:y-to>}{<y-ticks pos>}
% defaults
[]{<0,0>(0pt:3pt){<m>}(0pt:3pt){}}
% starred(*) version always suppresses tick labels

% \tzticks*
\begin{tikzpicture}[scale=.4,font=\scriptsize]
\zhelplines\tzticks*\tshoworigin\taxes(-1,-1)(10,10)
\tzticks*[draw=red,thick]
\{1,...,7,8/\alpha\} % label ignored
\{0pt:3cm\}\{2,...,6,7/\beta\} % label ignored
\end{tikzpicture}

% \tzticks(*)
\begin{tikzpicture}[scale=.5]
\zhelplines\tzticks\tshoworigin\taxes(-1,-1)(10,8)
\{0,0.2,...,8\}
\{0,0.2,...,7\} % default (Opt:3pt)
\tzticks*[draw=red,thick]
\{opt:10pt\}\{1,...,8\}
\{opt:10pt\}\{1,...,7\}
\tzticks\{1,...,8\}
\{2,...,7\} % default (Opt:Opt)
\end{tikzpicture}

18.3 \tzticksx(*) and \tzticksy(*)

You can handle x ticks and y ticks independently.

X ticks \tzticksx only prints x-tick labels but not tick marks, by default. To prints tick marks you need to specify \texttt{(x-from:x-to)}.

% syntax
\tzticksx[<opt>]<y-shift>{<from>:<to>}{<x-tick pos/labels>}{<node opt>}
% defaults
[<0,0>(Opt:0pt){<m>}[text height=1.25ex,text depth=.25ex,below]
\zticksx* only prints x-tick marks from 0pt to 3pt, by default, suppressing tick labels.

% syntax:
\zticksx*[<opt>]<y-shift>({from:<to>}){<xtick pos>}
% defaults
*[]<>(0pt:3pt){<m>}
% starred(*) version always suppresses tick labels

Y ticks \zticksy only prints y-tick labels but not tick marks, by default. To print tick marks you need to specify ({<x-from>:<x-to>}).

\zticksy* only prints y-ticks from 0pt to 3pt by default, suppressing tick labels.

% syntax:
\zticksy*[<opt>]<y-shift>({from:<to>}){<y-ticks pos/labels}>[<node opt>]
% defaults
[]<>(0pt:0pt){<m>}
% starred(*) version suppresses tick labels

\begin{tikzpicture}[scale=.4,font=\scriptsize]
\zhelplines
\tzshoworigin
\tzaxes(-1,-1)(10,10)
\tzaxes[dashed]<2,1>(-1,-1)(10,10)
\zticksx*[draw=red,thick]
(-5pt:1cm){1,...,7,8/$\alpha$}
\zticksy*[draw=blue,thick]
(0pt:3cm){2,...,6,7/$\beta$} % labels ignored
\end{tikzpicture}

Shift The options \texttt{<y-shift>} and \texttt{<x-shift>} move x-ticks and y-ticks, respectively.

\begin{tikzpicture}[scale=.4,font=\scriptsize]
\zhelplines
\tzshoworigin
\tzaxes(-1,-1)(10,10)
\tzaxes[dashed]<2,1>(-1,-1)(10,10)
\zticksx*[draw=red,thick]
<-1>{-5pt:10pt}{1,...,7,8/$\alpha$} % labels ignored
\zticksy*[draw=blue,thick]
<2>(0pt:3cm){2,...,6,7/$\beta$}
\end{tikzpicture}
19 Projections

19.1 \tzproj(*)

\tzproj accepts a mandatory coordinate and draws perpendicular lines onto each axis from the coordinate. The lines are dotted, by default.

% syntax: minimum
\tzproj(<coor>)
% syntax: medium
\tzproj*[<coor>]{<x-text>}{<node opt>}{<y-text>}{<node opt>}
% syntax:
\tzproj*[<opt>]<x-shift,y-shift>(){<coor>}
\{<x-text>}{<node opt>}{<y-text>}{<node opt>}(<dot size>)
% defaults
*[dotted]<0,0>(<m>){}
[text height=1.25ex,text depth=.25ex,below]{}[left](2.4pt)

\tzproj* additionally prints a ‘black node dot’ of the size 2.4pt, by default. Internally, the node dot is processed by \tzdot*. The first option <opt> does not control the node dot.

Dot size  You can only control the size of dots by the last optional argument (<dot size>) or by the THREE WAYS on page 41. If you want to control fill or color of dots, use \tzdot* separately.

%\tzproj(*)
\begin{tikzpicture}[scale=.5]
\tzaxes(8,6)
\tzproj[dashed,blue](2,3)
\tzcoors(30:7)(A)(50:6)(B);
\tzproj*[A]
\tzproj*[dashed,text=blue](B)(5pt)
\end{tikzpicture}

Adding text  You can also add text around the projection point on each axis by the option \{<text>\}. The position and color of the text is controlled by the option \[<node option>\]. The default position is (approximately) \[below\] for the x axis and \[left\] for the y axis.

% \tzproj(*): adding text
\begin{tikzpicture}[scale=.5]
\tzhelplines(8,6)
\tzaxes(8,6)
\tzproj[dashed,blue](2,3){$x$}{$y$}
\tzcoors(30:7)(A)(50:6)(B);
\tzproj*[text=blue](A){$a_1$}{$a_2$}
\tzproj*[dashed](B){$x^*$}{$y^*$}[green]{*$y^*$}[red]
\end{tikzpicture}

Projection shift  Specifying the option \[x-shift,y-shift\] moves the projection point and text on each axis.
19.2 \texttt{\textbackslash tzproj(*)} and \texttt{\textbackslash tzprojy(*)}

\texttt{\textbackslash tzprojx(*)} draws a dotted line, which is perpendicular to the x axis.
\texttt{\textbackslash tzprojx*} additionally prints a ‘black node dot’ of the size 2.4pt, by default.

\begin{verbatim}
\begin{tikzpicture}
  \helplines(-1,-2)(6,6)
  \showorigin
  \axes(-1,-1)(6,6)
  \proj*(<x-shift,y-shift>(<coor>){<x-text>}[<node opt>](<dot size>)
\end{tikzpicture}
\end{verbatim}

\texttt{\textbackslash tzprojy(*)} draws a dotted line, which is perpendicular to the y axis.
\texttt{\textbackslash tzprojy*} additionally prints a ‘black node dot’ of the size 2.4pt, by default.

\begin{verbatim}
\begin{tikzpicture}
  \helplines(-1,-2)(6,6)
  \showorigin
  \axes(-1,-1)(6,6)
  \projy*(<x-shift,y-shift>(<coor>){<y-text>}[<node opt>](<dot size>)
\end{tikzpicture}
\end{verbatim}

You can only control the size of dots by the last option \texttt{(<dot size>)}. If you want to control fill or color of dots, use \texttt{\textbackslash tzdot*} separately. You can also add text around the projection point on each axis by specifying the option \texttt{(<x-text>)} or \texttt{(<y-text>)} followed by the option \texttt{[<node option>]}.

\begin{verbatim}
\begin{tikzpicture}
  \helplines(-1,-2)(6,6)
  \showorigin
  \axes(-1,-1)(6,6)
  \projx*(<x-shift,y-shift>(<coor>){<x-text>}[<node opt>](<dot size>)
\end{tikzpicture}
\end{verbatim}

Specifying the option \texttt{<x-shift,y-shift>} with \texttt{\textbackslash tzprojx(*)} and \texttt{\textbackslash tzprojy(*)} moves the projection point and text accordingly.
20 Plot Functions

20.1 \tzfn: Plot functions

\tzfn plots a function of \(x\).

\begin{tikzpicture}[scale=.5,font=\scriptsize]
\zhelplines(-1,-2)(6,6)
\zshoworigin
\zaxes(-1,-1)(6,6)
\zaxes[blue]<2,1>(-1,-1)(6,6)
\tzprojx*[green,thick,solid]<2,1>(3,4){$x=3$}[blue]
\tzprojy*[thick]<2,1>(5,2)
\end{tikzpicture}

\tzfn takes two mandatory arguments: \{\text{<fn of } x\}\} and \{\text{<domain>}\}. The domain should be of the form \{\text{<from num:to num>}\}, like \{1:5\}.

\begin{tikzpicture}
\def\Dx{7-2/3*x}
\tzfn{\Dx}{0:7}\{D\}\[r\]
\tzfn{1+x}{0:7}\{S\}\[blue,r\]
\end{tikzpicture}

% \tzfn
\begin{tikzpicture}[scale=.5]
\tzaxes(8,8)
\def\Dx{7-2/3*x}
\tzfn{\Dx}{0:7}\{$D$}[r]
\tzfn{1+x}{0:7}\{$S$}[blue,\text{\node opt}]\[code.append\]
\end{tikzpicture}

You can also use the predefined functions of \text{TikZ} such as \text{sin}, \text{cos}, \text{ln}, \text{log10}, \text{log2}, \text{exp}, \text{sqrt}, and so on. (See \text{TikZ} manual.)
20.1.3 Name paths: name path

You can name the path of \texttt{tznfn} by specifying the option "\texttt{<path name>}" immediately before the mandatory argument \{\texttt{<fn of \textbackslash x>}\}. You can use the path name to find intersection points.

\begin{verbatim}
\texttt{tznfn"mypath"Fx[1:5]} \% works like:
\texttt{draw [samples=200,,domain=1:5,name path=mypath] plot (\textbackslash x,(\textbackslash Fx))};
\end{verbatim}

Remark: Suppose that the function’s expression \texttt{<fn of \textbackslash x>} consists only of a macro name, say \texttt{Fx}. Then

- The macro name \texttt{Fx} (\textit{without the backslash}) is automatically assigned to \texttt{<path name>}, unless you give another name.
- That is, \texttt{tznfn\textbackslash Fx} is equivalent to \texttt{tznfn"Fx"\textbackslash Fx}. (\textit{You don’t need to type the same thing twice}.)

\begin{verbatim}
\texttt{tznfn\textbackslash Fx[1:5]} \% works like:
\texttt{draw [samples=200,,domain=1:5,name path=Fx] plot (\textbackslash x,(\textbackslash Fx))};
\end{verbatim}

20.1.4 Move graphs: shift

You can move the graph of \texttt{tznfn} by specifying the option \texttt{<shift coor>} before the mandatory argument \{\texttt{<fn of \textbackslash x>}\} or immediately before the option "\texttt{<path name>}", if it exists. The empty shift option \texttt{<>()} is not allowed.
20.1.5 Extend paths: \texttt{<code.append>}, \texttt{tzfnAtBegin}, \texttt{tzfnAtEnd}

\texttt{<code.append>} You can extend the path created by \texttt{tzfn} from the end of the graph, by writing TikZ code in the last optional argument \texttt{<code.append>}. Internally it adds the TikZ code to the path after the options \texttt{<text>} and \texttt{<node opt>}. 

\texttt{tzfnAtEnd} You can also extend the path with \texttt{tzfnAtEnd}. Internally it adds TikZ code immediately before the options \texttt{<text>} and \texttt{<node opt>}. But you have to use \texttt{tzfnAtEnd (immediately) before each \texttt{tzfn}}.
\section{tzLFn: Plot linear functions}

\texttt{tz LF n} draws a linear function: \( y = ax + b \).

- When you know two coordinates on a line, \texttt{tz LF n(\texttt{coor 1})(\texttt{coor 2})} draws the line.
- When you know one coordinate and the slope of a line, \texttt{tz LF n(\texttt{coor 1}){\texttt{slope}}} draws the line.
- If you specify all the three arguments \((\texttt{coor 1})(\texttt{coor 2})\{\texttt{slope}}\), then the slope is ignored.

\begin{verbatim}
% syntax
\texttt{tz LF n[<opt>]<shift coor>"<path name>"
   (\texttt{coor 1})(\texttt{coor 2}){\texttt{slope}}[<domain>]{\texttt{text}}[<node opt>]\texttt{code.append}
% defaults
\[<>"(\texttt{m})()\]
\end{verbatim}

\texttt{tz LF n} accepts two mandatory arguments: \((\texttt{coor 1})\) and \[\texttt{domain}].

- The domain should be of the form \[\texttt{from num:to num}\].
- If just one coordinate is specified without a slope, the slope is regarded as 1, by default.

For example, \texttt{tz LF n(1,1)(2,3){0:4}} draws a line passing through two points: \((1,1)\) and \((2,3)\), over \(0 \leq x \leq 4\). \texttt{tz LF n(1,1){.5}{0:4}} draws a line passing through a point \((1,1)\) with the slope \(.5\), over \(0 \leq x \leq 4\).

You can add text at the end of the line of \texttt{tz LF n} by the options \{\texttt{text}\} and \{\texttt{node opt}\}. You can also name the path of \texttt{tz LF n} by specifying the option "\texttt{path name}" immediately before the mandatory coordinate.

\begin{verbatim}
% \texttt{tz LF n}
\begin{tikzpicture}
\texttt{helplines(1,1)(A)(1,2)(B)(3,1)(C);}
\texttt{tz LF n[red]"Fx"(A){.5}{0:4}{$f(x)$}[a]}
\texttt{tz LF n[blue]"Gx"(B)(C){0:4}{$g(x)$}[r]}
\texttt{tz X point*[fill=none]{Fx}{Gx}(E){$E$}[3pt]}
\end{tikzpicture}
\end{verbatim}

You can move the line of \texttt{tz LF n} by specifying the option \texttt{shift coor} immediately before the option "\texttt{path name}". (The empty shift option <> is not allowed.) You can also expand the path of \texttt{tz LF n} by writing Ti\texttt{kZ} code in the last optional argument \texttt{code.append}.
20.3 Horizontal lines

20.3.1 \tzhnat

\tzhnat draws a horizontal line at a specified value of \(y\).

\begin{tikzpicture}
\helplines(4,4)
\coors*(1,1)(A)(1,2)(B)(3,1)(C);
\LFn[blue]"Gx"(B)(C)[0:4]{$g(x)$}[r]
\LFn[dashed,red,->]<1,1>"Gx"(B)(C)[0:4]{$g(x)$}[r]
< arc (0:140:2) node [below] {End!} >
\end{tikzpicture}

You can name the path of \tzhnat by the option "\texttt{<path name>}". You can move the line by the option \texttt{<shift coor>}. You can also expand the path from the end of the line by writing TikZ code in the last optional argument \texttt{<code.append>}. 

\begin{tikzpicture}
\helplines(4,4)
\fnat{0}{<line A>}[blue,thick]
\fnat{1}{1:3}<line B>[red,thick]
\fnat{2}{0:3} node {line C}
\end{tikzpicture}

% \tzFn: shift, expanding path
\begin{tikzpicture}
\helplines(4,4)
\coors*(1,1)(A)(1,2)(B)(3,1)(C);
\LFn[blue]"Gx"(B)(C)[0:4]{$g(x)$}[r]
\LFn[dashed,red,->]<1,1>"Gx"(B)(C)[0:4]{$g(x)$}[r]
< arc (0:140:2) node [below] {End!} >
\end{tikzpicture}
\begin{tikzpicture}
\coordinate (A) at (0,1);
\coordinate (B) at (0,2);
\draw (A) -- (B);
\end{tikzpicture}

\begin{tikzpicture}
\coordinate (A) at (0,1);
\coordinate (B) at (0,2);
\draw (A) -- (B);
\end{tikzpicture}

\subsection{20.3.2 \texttt{tzhfn}}

\texttt{tzhfn} accepts a coordinate as a mandatory argument and draws a horizontal line at the \texttt{y} value of the coordinate. For example, \texttt{tzhfn(\texttt{<x>},3)}, ignoring \texttt{<x>}, is equivalent to \texttt{tzhfnat{3}}.

\textit{Everything else is the same as in \texttt{tzhfnat}.}

\begin{tikzpicture}
\coordinate (A) at (0,1);
\coordinate (B) at (0,2);
\draw (A) -- (B);
\end{tikzpicture}

\subsection{20.4 Vertical lines}

\subsection{20.4.1 \texttt{tzvfnat}}

\texttt{tzvfnat} draws a vertical line at a specified value of \texttt{x}.

\begin{tikzpicture}
\coordinate (A) at (0,1);
\coordinate (B) at (0,2);
\draw (A) -- (B);
\end{tikzpicture}

\begin{tikzpicture}
\coordinate (A) at (0,1);
\coordinate (B) at (0,2);
\draw (A) -- (B);
\end{tikzpicture}
You can name the path of \tzvfnat by the option "<path name>". You can move the line by the option <shift coor>. You can also expand the path from the end of the line by writing Ti\kZ code in the last optional argument <code.append>.

In the previous example, \tzhelplines* is used to fix a bounding box. (See Section 6.2 on page 35, for more details.)

20.4.2 \tzvfn

\tzvfn accepts a coordinate as a mandatory argument and draws a horizontal line at the x value of the coordinate. For example, \tzvfn(3,<y>), ignoring <y>, is equivalent to \tzvfnat{3}.

Everything else is the same as in \tzvfnat.
21 Intersections

21.1 $\texttt{tzXpoint}(\star)$: Intersection points

$\texttt{tzXpoint}$ finds intersection points of two paths and saves them as coordinate names for later use.

\begin{minipage}[t]{0.8\textwidth}
\begin{alltt}
\begin{verbatim}
% syntax: minimal
\texttt{tzXpoint}{\texttt{<path>}}{\texttt{<path>}}{\texttt{<coor name>}}
% syntax: medium
\texttt{tzXpoint}{\texttt{<path>}}{\texttt{<path>}}{\texttt{<coor name>}}{\texttt{<label>}}{\texttt{<angle>}}
% syntax: full
\texttt{tzXpoint}{\texttt{<opt>}}{\texttt{<path>}}{\texttt{<path>}}{\texttt{<coor name>}}{\texttt{<nth>}}{\texttt{<label>}}{\texttt{<label opt>}}{\texttt{angle}}{\texttt{(dot size)}}
% defaults
\end{verbatim}
\end{alltt}
\end{minipage}

For example, $\texttt{tzXpoint}\{\texttt{path1}\}\{\texttt{path2}\}\{\texttt{A}\}$ determines an intersection of the two paths and names the point (A) or (A-1). (By default, the name is (intersection) as in TikZ.) If there are two or more intersection points, they are named as follows: (A)=(A-1), (A-2), (A-3), etc.

You can determine which intersection point is named directly by specifying the option [nth]. If you select the second intersection point to be named (A) out of multiple intersections, they are named as follows: (A-1), (A)=(A-2), (A-3), (A-4), etc. You can label intersection points by specifying the option {<label>} and [<angle>].

\begin{minipage}[t]{0.8\textwidth}
\begin{verbatim}
\begin{tikzpicture}[scale=.5]
\draw[help lines] (8,8) grid (0,0);
\draw[red,bend right] "AA" (1,8) (8,1);
\draw[blue,bend right] "BB" (0,2) (8,6);
\node[dot=2.4pt] (A) at (AA) {A};
\end{tikzpicture}
\end{verbatim}
\end{minipage}

$\texttt{tzXpoint}\star$ The starred version $\texttt{tzXpoint}\star$ simply adds a node dot to $\texttt{tzXpoint}$. The default dot size is 2.4pt and it can be changed by the last option (<dot size>) or the THREE WAYS (on page 41).

\begin{minipage}[t]{0.8\textwidth}
\begin{alltt}
\begin{verbatim}
% syntax: minimum
\texttt{tzXpoint}\star{\texttt{<path>}}{\texttt{<path>}}
% syntax: medium
\texttt{tzXpoint}\star{\texttt{<path>}}{\texttt{<path>}}{\texttt{<coor name>}}{\texttt{<label>}}{\texttt{<angle>}}
% syntax: full
\texttt{tzXpoint}\star{\texttt{<opt>}}{\texttt{<path>}}{\texttt{<path>}}{\texttt{<coor name>}}{\texttt{<nth>}}{\texttt{<label>}}{\texttt{<label opt>}}{\texttt{angle}}{\texttt{(dot size)}}
% defaults
\end{verbatim}
\end{alltt}
\end{minipage}
21.2 Vertical intersection points

21.2.1 \tzvXpointat(*)

\tzvXpointat determines vertical intersection points of a path at a specified value of $x$. So it takes \{<path>\} and \{<x-val>\} as mandatory arguments.

Remark: Internally, \tzvXpointat depends on the current bounding box, which generally does not cause a problem because it is used after paths to be intersected are formed. In case of any problem of no intersection point, you may want to fix a bounding box using \tzbbox or Ti\hskip1ptkZ’s \useasboundingbox.

The starred version \tzvXpointat* additionally prints a node dot of the size 2.4pt, by default, at the (first) intersection point.
\begin{tikzpicture}
\helplines(4,3)
\def\Fx{.5*(x-2)^2}
\fn{Fx[0:4]} % name path=Fx
\fnat[dashed]{1}
\tfnat(Fx){1}(A)
\tfnat(Fx){3.2}(B)[-45]
\end{tikzpicture}

21.2.2 \texttt{\textbackslash tzvXpoint(*)}

\texttt{\textbackslash tzvXpoint} accepts \{<path>\} and \{<coor>\} as mandatory arguments to find vertical intersection points of a path at the \(x\) value of the coordinate, ignoring the \(y\) value. For example, \texttt{\textbackslash tzvXpoint(mypath)(3,<y>)}, ignoring \(<y>\), is equivalent to \texttt{\textbackslash tzvXpointat(mypath){3}}.

\textit{Everything else is the same as in \texttt{\textbackslash tzvXpointat}.}

\begin{tikzpicture}
\coors(1,0)(A)(3.2,0)(B);
\fnat[dashed]{1,0}
\tfnat{Fx}(A){A}(Ax)
\tfnat{Fx}{B}[Ax][45]
\tfnat{Fx}{B}[B][-45]
\end{tikzpicture}

21.3 Horizontal intersection points

21.3.1 \texttt{\textbackslash tzhXpointat(*)}

\texttt{\textbackslash tzhXpointat} determines horizontal intersection points of a path at a specified value of \(y\). So it takes \{<path>\} and \{<y-val>\} as mandatory arguments.
The starred version \texttt{\textbackslash xpoint\bigstar} additionally prints a node dot of the size 2.4pt, by default, at the intersection point.

\begin{tikzpicture}
\draw[help lines] (4,3);
\def\Fx{.5*(x-2)^2}
\tzfn\Fx[0:4] \% name path=Fx
\tzXpointat{Fx}(1)(A)
\tzdot(A){A}[0]
\tzfnat[dashed]{{1.5}}(X){X}[45]
\tzXpointat*{Fx}{{1.5}}(X){X}[45]
\tzdot*{(X-2)}{Y}[135]
\end{tikzpicture}

21.3.2 \texttt{\textbackslash xpoint\bigstar}

\texttt{\textbackslash xpoint} accepts \{<path>\} and \{<coor>\} as mandatory arguments to find horizontal intersection points of a path at the y value of the coordinate, ignoring the x value. For example, \texttt{\textbackslash xpoint\{mypath\}{{x},3}}, ignoring \texttt{x}, is equivalent to \texttt{\textbackslash xpointat\{mypath\}{{3}}}.

\textit{Everything else is the same as in \textbackslash xpointat.}

\begin{tikzpicture}
\tzXpoint\{opath\}{<path>}{<coor>}{<coor name>}{<label>}{<angle>}
\% defaults
[]
\end{tikzpicture}

The starred version \texttt{\textbackslash xpoint\bigstar} just adds a node dot to \texttt{\textbackslash xpoint}.
prints, at the (first) intersection point, a node dot of the size 2.4pt, by default.

\begin{tikzpicture}
\def\Fx{.5*(x-2)^2}
\tzfn\Fx[0:4] % name path=Fx
\tzcoors(0,1)(A)(0,1.5)(B);
\tzhelplines*(4,3)
\tzhelplines*(0,4) % name path=Fx
\tzdot(A){A}[0]
\tzcoors(0,1.5)(X)(X)[45]
\tzdot*(X-2){Y}[135]
\end{tikzpicture}

22 Secant and Tangent Lines

22.1 Secant lines

\texttt{\textbackslash tzsecantat} draws a line segment or a secant line of a curve, on the behind layer by default. You need to specify a path name and two values of \texttt{x}. With \texttt{\textbackslash settzsecantlayer}, you can change the layer, like \texttt{\textbackslash settzsecantlayer{main}}.

\begin{tikzpicture}[scale=.5]
\tzaxes(-1,-1)(8,6){$x$}{$y$}
\tzbezier+"curve"(.5,1)(1,6)(-1,-4)(7,5)
\tzsecantat[thick,red]{curve}{1}{3}
\tzsecantat[thick,red]{curve}{1}{4}
\tzsecantat[thick,red]{curve}{1}{7}
\end{tikzpicture}

\textbf{Domain} The domain should be of the form \texttt{[from num:to num]}. Without specifying the optional domain, \texttt{\textbackslash tzsecantat} draws a line segment connecting two points on the (curved) path.

\begin{tikzpicture}[scale=.5]
\tzaxes(-1,-1)(8,6){$x$}{$y$}
\tzbezier+"curve"(.5,1)(1,6)(-1,-4)(7,5)
\tzsecantat[thick,red]{curve}{1}{3}
\tzsecantat[thick,red]{curve}{1}{4}
\tzsecantat[thick,red]{curve}{1}{7}
\end{tikzpicture}

You can optionally specify the \texttt{[domain]}. 

121
Shift  You can move \texttt{\textbackslash tzsecantat} by specifying the option \texttt{<shift coor>}. 

Naming paths  By specifying the option \texttt{"<path name>"} you can name a path of \texttt{\textbackslash tzsecantat}. 

<code.append> You can extend the path of \texttt{\textbackslash tzsecantat} by writing TikZ code in the last optional argument \texttt{<code.append>}. 

---

% \texttt{\textbackslash tzsecantat: domain} 
\begin{tikzpicture}[scale=.5]  
\tzhelplines(8,6)  
\tzaxes(-1,-1)(8,6){\$x\$}{\$y\$}  
\tzbezier*\"curve"(.5,1)(1,6)(-1,4)(7,5){ar}  
\tzsecantat[blue]{curve}{1}(3){0:5}{secant line}[a]  
\tzsecantat[thick,red]{curve}{1}(4)  
\tzsecantat[blue,dashed]{curve}{1}(7){0:8}  
\end{tikzpicture}

% \texttt{\textbackslash tzsecantat: shift, name path (intersection)} 
\begin{tikzpicture}[scale=.5]  
\tzhelplines(8,6)  
\tzaxes(-1,-1)(8,6){\$x\$}{\$y\$}  
\tzbezier*\"curve"(.5,1)(1,6)(-1,4)(7,5){ar}  
\tzsecantat[thick,red]{curve}{1}(4)  
\tzsecantat[blue]{curve}{1}(3){0:5}{secant line}[a]  
\tzsecantat[blue]<1,-1>\{shift\}{1}(3){0:5}{shifted}[r]  
\tzvXpointat*{shift}{3}{X}{-45}  
\end{tikzpicture}

% \texttt{\textbackslash tzsecantat: <code.append>} 
\begin{tikzpicture}[scale=.5]  
\tzhelplines(8,6)  
\tzaxes(-1,-1)(8,6){\$x\$}{\$y\$}  
\tzbezier*\"curve"(.5,1)(1,6)(-1,4)(7,5){ar}  
\tzsecantat[blue]{curve}{1}(3){0:5}{secant line}[a]  
\tzsecantat[blue,->]<1,-1>\{shift\}{1}(3){0:5}  
\tzvXpointat*{shift}{3}{X}{-45}  
\end{tikzpicture}
22.1.2 \texttt{tzhsecant}

\texttt{tzhsecant} uses two coordinates instead of two values of $x$ to draw a line segment or a secant line of a curve, on the behind layer by default. You need to specify a path name and two coordinates, then \texttt{tzhsecant} uses the $x$ values of the two coordinates.

\textit{Everything else is the same as in \texttt{tzhsecant}.}

You can change the layer with \texttt{setztzhsecantlayer}.

% syntax: minimum
\begin{tikzpicture}[scale=.5]
\tzhelplines
\tzaxes(-1,-1)(8,6){$x$}{$y$}
\tzbezier*"curve"(.5,1)(1,6)(-1,-4)(7,5){curve}[ar]
\tzcoor(1,0)(K)
\tzsecant[blue]{curve}(K)(3,0)[0:5]{secant line}[a]
\tzsecant[thick,red]{curve}(K)(4,0)
\tzsecant[blue,dashed]{curve}(K)(7,0)[0:8]
\end{tikzpicture}

The domain should be of the form \texttt{<from num:to num>}. Without specifying the optional domain, \texttt{tzhsecant} draws a line segment connecting two points on the (curved) path.

You can move the secant line and extend the path.

% \texttt{tzhsecant: shift, <code.append>}
\begin{tikzpicture}[scale=.5]
\tzhelplines(8,6)
\tzaxes(-1,-1)(8,6){$x$}{$y$}
\tzbezier*"curve"(.5,1)(1,6)(-1,-4)(7,5){curve}[ar]
\tzcoor(1,0)(K)
\tzsecant[blue]{curve}(K)(3,0)[0:5]{secant line}[a]
\tzvXpointat*{shift}{3}{X}{-45}
\end{tikzpicture}

22.2 Tangent lines

22.2.1 \texttt{tztangentat}

\texttt{tztangentat} draws a tangent line to a curve at a specified value of $x$. By default, the tangent line is drawn on the behind layer, which can be changed by \texttt{setztztangentlayer}, like \texttt{setztztangentlayer{main}}.

\textbf{Remark:} To calculate the slope at $x$, $x$ varies over the interval $(x-\varepsilon_1, x+\varepsilon_2)$ and $\varepsilon_1 = \varepsilon_2 = 0.01$, by default. So the slope of tangent line is only \textit{approximate}. 

123
Domain \texttt{\ztangentat} takes three mandatory arguments: \{<path>\}, \{<x-val>\}, and \{<domain>\}. The mandatory argument \{<domain>\} should be of the form \{<from num:to num>\}.

Shift You can move the tangent line by specifying the option \texttt{<shift coor>}.

Naming paths By specifying the option \texttt{"<path name>"}, you can name the path of \texttt{\ztangentat}.
Variations  The slope of the tangent line is approximate, so sometimes you may want to change the variation interval to get better results. You can change $\varepsilon_1$ and $\varepsilon_2$ by specifying the option \( (\varepsilon_1,\varepsilon_2) \) immediately after the mandatory argument \( \{x-val\} \). Or you can change the variations by the macro \settztangentepsilon\, like \settztangentepsilon{\varepsilon_1}{\varepsilon_2}. The effect remains until the end of \tikzpicture environment unless changed again.

\begin{tikzpicture}[scale=.5,font=\footnotesize]
\begin{scope}[scale=2,thick,red]
\draw [blue] plot [smooth] coordinates { (0,0) (1,1) (2,4) };
\end{scope}
\end{tikzpicture}

% \settztangentepsilon: variations, shift
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
% \settztangentepsilon{\varepsilon_1}{\varepsilon_2}\quad \text{sets the variations to } \varepsilon_1 \text{ and } \varepsilon_2.
22.2.2 \settztangent

\settztangent uses a coordinate instead of a value of x to draw a tangent line. For example, \settztangent{curve}(4,\langle y \rangle) is equivalent to \settztangentat{curve}(4) for any \langle y \rangle.

Everything else is the same as in \settztangentat.

You can extend the path of \settztangentat by writing Ti\TeX code in the last optional argument <code.append>.

\begin{tikzpicture}[scale=.5,font=\footnotesize]
\begin{scope}[scale=2,thick,red]
\draw [blue] plot [smooth] coordinates { (0,0) (1,1) (2,4) };
\end{scope}
\end{tikzpicture}

You can extend the path of \settztangentat by writing Ti\TeX code in the last optional argument <code.append>.

\begin{tikzpicture}[scale=.5,font=\footnotesize]
\begin{scope}[scale=2,thick,red]
\draw [blue] plot [smooth] coordinates { (0,0) (1,1) (2,4) };
\end{scope}
\end{tikzpicture}
\ztangent accepts three mandatory arguments: \{\textit{path}\}, \{\textit{coor}\}, and \{\textit{domain}\}.

You can shift the tangent line and extend its path.

You can control the interval of variations of $x$ by the option \{(\textit{epsilon1},\textit{epsilon2})\} or the macro \texttt{\settztangentepsilon}.
23 Miscellany

23.1 \tzbrace()

\tzbrace takes two coordinates as mandatory arguments to draw a calligraphic brace connecting them.

% syntax: minimum
\tzbrace(<coor>)(<coor>)
% syntax: medium
\tzbrace(<coor>)(<coor>{<text>}[<node opt>]
% syntax: full
\tzbrace[<draw opt>]{<raise>}{<decoration opt>}{<shift coor>}
  (<coor>)(<coor>{<text>}[<node opt>]
% defaults
[]{5pt}[amplitude=5pt]<(<m>)(<m>)}{}

The raise value of a brace is 5pt by default and the value can be changed by the first curly brace optional argument {<raise>}

The amplitude of a brace is 5pt by default. You can control the amplitude by writing the option amplitude=<dim> in the second bracket option [<decoration opt>].

\tzbrace[thick](0,0)(3,1) % works like:
\draw[thick,decorate,decoration={calligraphic brace, amplitude=5pt, raise=5pt}]
  (0,0) to (3,1);

The swap version \tzbrace' swaps the coordinates. So it prints a mirror image of \tzbrace.
For example, \tzbrace'(0,0)(3,1) is equivalent to \tzbrace(3,1)(0,0).

\begin{tikzpicture}[sloped]
\zhelplines(4,2)
\tzline(0,0)(3,1)
\tzbrace(0,0)(3,1){AAA}[above=10pt,blue]
\tzbrace'(0,0)(3,1){BBB}[below=10pt]
\end{tikzpicture}

You can change the style of the decorating brace by the second bracket optional argument [<decoration opt>].

You can also move a brace by specifying the option <shift coor> immediately before the the first mandatory coordinate. The empty shift option <> is not allowed.

\begin{tikzpicture}[sloped]
\zhelplines(4,3)
\tzline(0,0)(3,1)
\tzbrace[very thick,pen colour=blue]
  (0,0)(3,1){AAA}[a=15pt]
\tzbrace[red,very thick](10pt)
  (0,0)(3,1){BBB}[b=20pt]
\end{tikzpicture}

\begin{tikzpicture}[sloped]
\zhelplines(4,3)
\tzline(0,0)(3,1)
\tzbrace[very thick,pen colour=blue]
  (0,0)(3,1){AAA}[a=15pt]
\tzbrace[red,very thick](10pt)
  (0,0)(3,1){BBB}[b=20pt]
\end{tikzpicture}
\begin{tikzpicture}[sloped]
\zhelplines(4,3)
\tzline(0,0)(3,1)
\tzbrace[very thick,pen colour=blue][amplitude=10pt](0,0)(3,1)
\tzbrace[red,very thick][brace,amplitude=10pt](0,0)(3,1)
\end{tikzpicture}
Version history

- v1.0.1 (2021/03/20)
  - revised the document with typo corrections
  - added the option `<code.append>` to `<tzframe>, `<tzcircle>, and `<tzeellipse`
  - added aliases: `<let tzrectangle tzframe` and `<let tzoval tzeellipse`

- v1.0 (2021/02/28) uploaded to CTAN

- v0.999a (2021/02/27)
  - writing document
  - some `<tz<...>AtBegin` and `<tz<...>AtEnd` not documented

- v0.999 (2021/02/24)
  - fixed the title. “Plot Graphs with Ti<em>k</em>Z Abbreviations”
  - changed the default `<tzpathstyle>` and `<tzlinkstyle>` from ‘--’ to ‘to’ (no harm)
  - added `<tzpathlayer>` and `<settzpathlayer>`, for later use. (currently not used)
  - removed explanation of the option `<+or++>` in the document (considering to remove later)
  - changed the delimiter `<path style>` to `[<path style>]` in `tzpath` and `tzlink(<s)` (critical change)
  - added `text opacity=1` everywhere `fill opacity=.3` is used

Acknowledgement

I am grateful to Kangsoo Kim of KTUG (Korean TeX Users Group) who has written many packages including `oblivoir.cls` for helping to implement the idea of the semicolon version with `expl3`.

References

Index

【Symbols】
\<m> 35, 37, 41, 100

【A】
abbreviations 2, 8

【B】
bounding box 19, 35, 101

【C】
clockwise 93, 97
counterclockwise 93, 97
current bounding box 23, 114, 115

【L】
label node 2, 7, 42, 44, 46, 62

【M】
main node 2, 7, 8, 42, 44, 46, 62

【N】
node coordinate 61, 62

【P】
plus version 69, 73, 75, 78, 81, 84, 86

【S】
semicolon version 11, 39, 44, 78
\settzcdotradius 38, 40
\settzdots 45
\settzfillcolor 65, 80
\settzfillopacity 65, 80
\settzmarksize 53
\settzpathstyle 79
\settzseccntlayer 121, 123
\settztangentepsilon 125
\settztangentlayer 123
swap version 93, 95, 97, 98, 104

【T】
THREE WAYS 41
\tzarc 93
\tzarc' 93
\tzarcfrom 95
\tzarcfrom' 95
\tzarcsfrom 96
\tzaxes 18
\tzaxes* 19, 101, 114
\tzaxesL 103
\tzaxesL' 104
\tzaxisx 20, 101
\tzaxisy 20, 101
\zbbox 36, 114
\tbezier 6, 10, 81
\tbrace 127
\tbrace' 127
\tzcdot 3, 37
\tzcdot* 3, 37
\tzcdots 14, 39
\tzcdots* 14
\tcircle 91
\tcircle* 91
\tcoor 5, 8, 47
\tcoor* 5, 9, 48
\tcoors 15, 49
\tcoors* 15, 50
\tcoorsquick 15, 51
\tcoorsquick* 16, 51
tzdashed 35
tzdotted 35
tzdashed 41
tzdotted 41, 45
\tzdotted* 4, 41
\tzdots 14, 44
\tzdots* 14, 44
tzextend 70
\tzfn 23, 87, 110
\tzfnAtBegin 112
\tzfnAtEnd 112
\tzframe 89
\tzframe* 90
\tzframe** 90
\tzframe+ 89
\tzgetxyval 52
\tzhelplines 35
\tzhelplines* 35, 114
\zhfn 23, 115
\zhfnat 23, 114
\zhXpoint 26, 120
\zhXpoint* 26, 120
\zhXpointat 26, 119
\zhXpointat* 26, 120
\tzLFn 24, 113
\tzline 9, 67
\tzline+ 69
\tzlines 11, 70
\tzlines+ 73
tzmark 53
\tzmarksize 53
\tnode 7, 61
\tnode* 7, 61
\tnodecircle 65
\tnodecircle* 65
\tnodedot 63
\tnodedot* 63
\tnodeellipse 66
\tnodeellipse* 66
\tnodeframe 64

130