Abstract

This is a \LaTeX{} package that provides Ti\kern0.05em\hbox{\kern0.05em Z\kern0.05em} based macros to make it easy to draw graphs. The macros provided in the tzplot package are just abbreviations for Ti\kern0.05em\hbox{\kern0.05em Z\kern0.05em} codes, which can be complicated, but using the package, hopefully, makes drawing easier, especially when drawing repeatedly. The macros have been chosen and developed with an emphasis on drawing graphs in economics.

Keywords: lines, dots, curves, axes, functions, projection, ticks, intersections, tangent lines

Contents

Part I Prelimiaries

1 Introduction

1.1 About tzplot.sty ........................................... 1
1.2 Preoccupied style names ................................... 2
1.3 How to read this document .................................. 3

2 Changes

2.1 What’s New in version 2.1 .................................. 3
2.2 What’s New in version 2.0 .................................. 3
2.2.1 New macros ............................................. 3
2.2.2 Extending paths: \texttt{\textbackslash tz<...>AtBegin} and \texttt{\textbackslash tz<...>AtEnd} ................. 4
2.2.3 New coordinates ......................................... 4
2.2.4 Error messages ......................................... 4
2.2.5 Abridged strings to place labels for coordinates, dots, and points ........................................ 5
2.2.6 New styles for middle arrow tips ......................... 5
2.3 Remarks ................................................... 5

Part II Getting Started

3 An Intuitive Introduction I: Basics

3.1 Lines: Basics: \texttt{\textbackslash tzline}(0,0)(3,1) ............. 7
3.2 Dots: Basics ............................................. 7
3.2.1 A circle dot: \texttt{\textbackslash tzcdot}(0,0) ................... 7
3.2.2 A circle node dot: \texttt{\textbackslash tzdot}(0,0) ............. 8
3.2.3 Difference between \texttt{\textbackslash tzcdot} and \texttt{\textbackslash tzdot} .................. 8
3.3 Coordinates: Basics: \texttt{\textbackslash tzcoor}(0,0)(A) ............ 9
3.4 Curves: Basics .......................................... 10
3.4.1 \texttt{\textbackslash tzto}(0,0)(4,2) ....................... 10
3.4.2 \texttt{tzbezier} .................................................. 10
3.4.3 \texttt{tzparabola} .............................................. 10
3.5 Adding text: Nodes and placement .................................. 11
3.5.1 \texttt{tznodex(3,1)[text][right]} ............................... 11
3.5.2 Review: Main nodes and label nodes in TikZ .................... 11
3.5.3 Abbreviations of TikZ basic placement option styles: \texttt{a}, \texttt{r}, \texttt{ar}, \texttt{bl}, etc. .................. 12
3.6 Labeling dots and coordinates .................................... 12
3.6.1 \texttt{tdot}, \texttt{tzcdot} ................................... 12
3.6.2 \texttt{tcotx} ....................................................... 12
3.6.3 \texttt{tcoors*} .................................................. 13
3.7 Adding text next to lines or curves ................................ 13
3.7.1 \texttt{tzline} ..................................................... 13
3.7.2 \texttt{tzto} ....................................................... 14
3.7.3 \texttt{tzbpanion} ............................................... 14
3.7.4 \texttt{tzparabola} ............................................. 15

4 An Intuitive Introduction II: Repetition of Coordinates .......... 15
4.1 Linking many coordinates: Semicolon versions .................. 15
4.1.1 \texttt{tztix}: Connected line segments ......................... 15
4.1.2 \texttt{tzpolygon}, \texttt{tzpolygon*}: Closed paths ............. 16
4.1.3 \texttt{tzpath*}: Filling area .................................. 17
4.2 Many dots: Semicolon versions .................................. 18
4.2.1 \texttt{tzcdots(*)} ............................................. 18
4.2.2 \texttt{tzdots(*)} .............................................. 18
4.3 Many coordinates: Semicolon versions ........................... 19
4.3.1 \texttt{tzcoors}, \texttt{tzcoors*} ................................ 19
4.3.2 \texttt{tzcoorsquick} ........................................... 20
4.3.3 \texttt{tzcoorsquick*} ......................................... 20
4.4 Plot coordinates: Semicolon versions ............................ 21
4.4.1 \texttt{tzplot*}: Mark dots with [mark=*] ....................... 21
4.4.2 \texttt{tzplot}: Lines with [tension=0] ......................... 21
4.4.3 \texttt{tzplot*}[draw]: Lines with dots ....................... 22
4.4.4 \texttt{tzplotcurve}: Curves with [smooth,tension=1] ........... 22

5 An Intuitive Introduction III: Plotting Functions ............... 23
5.1 Axes .................................................................. 23
5.1.1 \texttt{tztaxes} .................................................. 23
5.1.2 \texttt{tztaxes*} .................................................. 23
5.1.3 \texttt{tzshoworigin}, \texttt{tzshoworigin*} ..................... 24
5.1.4 \texttt{tzaxisx}, \texttt{tzaxisy} .................................. 24
5.2 Ticks .................................................................. 25
5.2.1 \texttt{tzticks} .................................................... 25
5.2.2 \texttt{tzticks*} .................................................. 25
5.2.3 \texttt{tzticksx(*)}, \texttt{tzticksy(*)} ......................... 26
5.3 Projections on the axes .......................................... 26
5.3.1 \texttt{tzproy(*)}, \texttt{tzproy(*)} ............................ 26
5.3.2 \texttt{tzproy(*)} .................................................. 27
5.4 Plot functions ..................................................... 27
5.4.1 \texttt{tzfn} ....................................................... 27
5.4.2 \texttt{tzfnat}, \texttt{tzhfn}: Horizontal lines ................. 28
5.4.3 \texttt{tzvfnat}, \texttt{tzhfn}: Vertical lines .................. 28
5.4.4 \texttt{tzhfn}: Linear functions ................................ 29
5.5 Intersection points ................................................ 29
5.5.1 Naming paths ................................................... 29
5.5.2 \texttt{tzhXpoint(*)}: Intersection points of two paths ....... 29
5.5.3 \texttt{tzvXpointat(*)}, \texttt{tzhXpoint(*)}: Vertical intersection points ........................................... 30
5.5.4 \texttt{tzXpointat(*)}, \texttt{tzhXpoint(*)}: Horizontal intersection points ...................... 30
10.6 \tzplotcurve(*) ................................................. 64

11 Nodes ......................................................... 67
11.1 \tznod and \tznod* .............................................. 67
11.2 \tznodes and \tznodes*: Semicolon versions ............... 69
11.3 \tznodedot(*) .................................................. 71
11.4 \tznodedots(*): Semicolon versions ....................... 72
11.5 \tznodelframe and \tznodelframe* ......................... 74
11.6 \tznodelcircle and \tznodelcircle* ....................... 75
11.7 \tznodelellipse and \tznodelellipse* ..................... 76

12 Lines .......................................................... 77
12.1 \tzline: Connecting two points .................................. 77
12.1.1 Line styles .................................................. 77
12.1.2 Adding text .................................................. 78
12.1.3 Moving lines: \textit{shift} ..................................... 79
12.1.4 Extending paths .............................................. 79
12.1.5 Naming paths: Intersection points ...................... 79
12.2 \tzline+: Relative coordinates ............................. 80
12.3 Styles: \tzshorten and \tztextend ............................ 81
12.4 \tzlines: Connecting multiple points: Semicolon version . 81
12.5 \tzlines+: Relative coordinates: Semicolon version .... 84

13 Connecting Points ............................................. 86
13.1 \tzto: Two points ................................................ 86
13.2 \tzto+: Relative coordinates .................................. 88
13.3 \tztos: Multiple points: Semicolon version ................ 88
13.4 \tztos+: Relative coordinates: Semicolon version .......... 91
13.5 \tzlink: Two points .............................................. 91
13.6 \tzlinks: All in one: Semicolon versions ................... 94
13.6.1 \tzlinks: Standard version ............................... 94
13.6.2 \tzlinks+, \tzlinks*, \tzlinks++: Variants .................. 96
13.6.3 Putting text, shift, intersections, and extending paths ... 97

14 Filling Area ...................................................... 99
14.1 \tzpath: Semicolon version ....................................... 99
14.2 \tzpath*: Semicolon version .................................... 101
14.3 \tzpath+ and \tzpath**: Relative coordinates: Semicolon versions 102

15 Curves .......................................................... 103
15.1 Bézier curves .................................................... 103
15.1.1 \tzbezier ...................................................... 103
15.1.2 \tzbezier+: Relative coordinates ......................... 105
15.2 Parabolas ....................................................... 106
15.2.1 \tzparabola ..................................................... 106
15.2.2 \tzparabola+: Relative coordinates ..................... 108
15.3 Edges ........................................................... 109
15.3.1 \tzedge(+) ..................................................... 109
15.3.2 \tzedges(+): Semicolon versions ......................... 110
15.4 More curves ..................................................... 112
15.4.1 \tzplotcurve, \tzplot .................................... 112
15.4.2 \tzto, \tztos ................................................ 112
15.4.3 \tzlink, \tzlinks ............................................ 112
15.4.4 \tzfn ........................................................ 113

16 Polygons and Circles ........................................... 114
16.1 Polygons: \tzpolygon: Semicolon versions ................. 114
16.1.1 \tzpolygon ................................. 114
16.1.2 \tzpolygon* ............................... 115
16.1.3 \tzpolygon+, \tzpolygon++: Relative coordinates: Semicolon versions 115
16.2 Rectangles ................................. 116
16.2.1 \tzframe and its variants ............... 116
16.2.2 \tzrectanglering(*) .................... 117
16.3 Circles and rings .......................... 119
16.3.1 \tzcircle(*) .............................. 119
16.3.2 \tzring(*) ............................... 120
16.4 Ellipses ................................. 122
16.4.1 \tzellipse(*) ............................. 122
16.4.2 \tzellipsering(*) ...................... 123
17 Arcs, Wedges, and Angle Marks 124
17.1 \tzarc('): Centered arcs .................. 124
17.1.1 Arcs ..................................... 124
17.1.2 Elliptical arcs ........................... 125
17.2 \tzarcfrom('): Arcs as in Ti\textlZ \ldots 126
17.3 \tzarcfrom: Connected arcs: Semicolon version 127
17.4 Wedges ..................................... 128
17.4.1 \tzwedge(') ............................... 128
17.4.2 \tzwedge*(') .............................. 129
17.5 Angle marks ............................... 130
17.5.1 \tpointangle: Angles between points .... 130
17.5.2 \tzanglemark('): Angle marks .......... 131
17.5.3 \tzanglemark*('): Fill angle marks .... 133
17.5.4 \tzrightanglemark: Right angle marks 134
17.5.5 \tzrightanglemark*: Fill right angle marks 135
17.6 \tzdistance: Distances and changes ....... 136

18 Axes ........................................ 138
18.1 Draw axes .................................. 138
18.1.1 \tzaxes ................................ 138
18.1.2 \tzaxes*: ................................. 139
18.2 \txaxisx and \txaxisy ....................... 140
18.3 Display the origin ......................... 141
18.3.1 \txshoworigin ......................... 141
18.3.2 \txshoworigin* ....................... 142
18.4 \txaxesL('): L-type axes .................. 143

19 Ticks ....................................... 144
19.1 \txticks: Tick labels ...................... 144
19.2 \txticks*: Tick marks ...................... 146
19.3 \txticksx(*) and \txticksy(*) ............ 146

20 Projections ................................ 148
20.1 \txproj(*): Projections on the axes ...... 148
20.2 \txprojx(*) and \txprojy(*) .............. 149
20.3 \txproj*: Semicolon versions .......... 150
20.4 \txprojsx(*) and \txprojsy(*): Semicolon versions 151

21 Plot Functions ............................. 152
21.1 \txfn and \txfn': Plot functions and inverse functions 152
21.1.1 \txfn .................. 152
25.4.6 \texttt{\textbackslash tnormal*(')} ............................................. 188

26 Miscellany ................................................................. 190
  26.1 Middle arrows ......................................................... 190
      26.1.1 Controllable middle arrow tips: \texttt{--\textgreater \textless --} and \texttt{\textbackslash settzmidarrow} ........... 190
      26.1.2 Fixed middle arrow tip styles: \texttt{--o--}, \texttt{--x--}, \texttt{--/--} .................. 191
  26.2 \texttt{\textbackslash tzbrace(')} ..................................... 192
  26.3 \texttt{\textbackslash tzsnake(+)}: Snake lines (Experimental) ..................... 194

Version history .......................................................... 196

Acknowledgement .......................................................... 196

Reference ................................................................. 196

Index ............................................................... 197
Part I
Prelimiaries

1 Introduction

1.1 About tzplot.sty

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

What you can do with tzplot 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 Ti\kZ code.

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

How to load 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 Ti\kZ libraries:

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

In the version 2, more libraries are added to the list of preloaded libraries:

arrows.meta, bending, % (for middle arrow tips)
decorations.markings, % (for decoration)
decorations.pathmorphing, % (for snaked lines)
fpu, % (for angle computation)
spy % (for later use)

More comments 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 Ti\kZ tutorial. To make good use of this package, you need to familiarize yourself with Ti\kZ.
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.

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

- `tzdot` `tzmark` `tzdotted`
- `tzdashed` `tzhelplines` `tznodet`
- `tzshorten` `tzextend` `tzshowcontrols`

**Abbreviated styles** Following the manual of TikZ, this package also predefines abbreviations (or aliases) of TikZ’s basic placement options for main nodes as follows:

```latex
\tikzset{%
  a/.style={above=#1},
  b/.style={below=#1},
  c/.style={centered=#1}, % centered, not center
  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},
}
```

By these abbreviations (or aliases), we mean that, with the `\tzplot` package, we can use these alias styles in the `tikzpicture` environment as follows:

```latex
\draw (0,0) -- (1,0) node [ar] {line}; \% [ar] = [above right]
```

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

```latex
\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:

```latex
\begin{tikzpicture}
  <tzplot macros>
  <tikz macros>
    \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 Changes

2.1 What’s New in version 2.1

New macros

- \tzdistance: distance between two points, $\Delta x$, and $\Delta y$
- \tzslope': opposite direction of \tzslope
- \tzslope*: opposite direction of \tzslope

- \tnormalat: a normal line from a point on a path
- \tnormalat': opposite direction of \tnormalat
- \tnormal*: works like \tzslope, but rotated 90 degrees
- \tnormal*': works like \tzslope, but rotated -90 degrees
- \tnormal: a normal line from a point on a path
- \tnormal': opposite direction of \tnormal
- \tnormal*: works like \tzslope, but rotated 90 degrees
- \tnormal*': works like \tzslope, but rotated -90 degrees
- \settznormallayer: controls the layer of \tnormalat and its variants
- \settznormalepsilon

Redesigned macros

- \taxisx: added the option "<name path>" to find the intercepts easily (by default, \axisx)
- \taxisy: added the option "<name path>" to find the intercepts easily (by default, \axisy)
- \taxis: added the option "<path name>" with the default "axes"
- \taxisL: added the option "<path name>" with the default "axesL"

2.2 What’s New in version 2.0

2.2.1 New macros

New macros have been added.

- \tnfn: (swap version) inverse function of \tnfn
- \tnfnf, \tnfnf': function of $y$
- \tnfdefLF, \tnfdefLFnoy: to define linear functions
- \tLfn('), \tLfnf('): linear functions
- \tLfnxpoint(*): linear function intersection points
- \tnfnmin('), \tnfnmax('): envelope curves

- \tnfnarea(*), \settznfnarealayer, \tnfnarealine('): to fill the area under graphs
2.2.2 Extending paths: \tz<...>AtBegin and \tz<...>AtEnd

- \tztoAtBegin, \tztoAtEnd, \tztosAtBegin, \tztosAtEnd (version 1)
- \tzlineAtBegin, \tzlineAtEnd, \tzlinesAtBegin, \tzlinesAtEnd
- \tzlinkAtBegin, \tzlinkAtEnd, \tzlinksAtBegin, \tzlinksAtEnd
- \tzbezierAtBegin, \tzbezierAtEnd
- \tzparabolaAtBegin, \tzparabolaAtEnd
- \tzvfn(at)AtBegin, \tzvfn(at)AtEnd
- \tzhfn(at)AtBegin, \tzhfn(at)AtEnd
- \tzfnAtBegin, \tzfnAtEnd (version 1)
- \tzfnofyAtBegin, \tzfnofyAtEnd
- \tzLFnAtBegin, \tzLFnAtEnd
- \tzfnminAtBegin, \tzfnminAtEnd, \tzfnmaxAtBegin, \tzfnmaxAtEnd
- \tzplotAtBegin, \tzplotAtEnd, \tzplotcurveAtBegin, \tzplotcurveAtEnd

2.2.3 New coordinates

- (tzAAtMid): angle arc midpoint, depending on \tzanglemark
- (tzRAtAvertex): right angle mark vertex, depending on \tzrightanglemark

2.2.4 Error messages

Some macros, called *semicolon versions*, accept any number of coordinates. You MUST indicate when the coordinate iteration ends with a *semicolon* ;. Without the semicolon, an error occurs with the *error message*:

\textbf{! Package tzplot Error: You may have forgotten a semicolon here or above!}

Knowing two coordinates, you can use \tzLFn and related macros to graph a linear function through the two points without writing an explicit function. If you inadvertently try \textit{infinite} slopes, you will get an error with the *error message*:

\textbf{! Package tzplot Error: Perhaps you are trying an 'infinite slope' here or above!}
2.2.5 Abridged strings to place labels for coordinates, dots, and points

In TikZ, a label to a main node is placed by the \texttt{label} option. The syntax of the \texttt{label} option is \texttt{label={\{<label opt>\}<angle>:\{<label>\}}}. In TikZ, the position of labels is specified by \texttt{angles}. The positioning words like \texttt{above}, \texttt{below}, \texttt{below right}, and so on can be used and they are interpreted in TikZ as the corresponding angles.

\textit{Just to avoid frequent coding errors}, from the version 2, the \texttt{tzplot} package provides the \texttt{abridged strings} \texttt{a}, \texttt{b}, \texttt{c}, \texttt{br}, and so on that you can use instead of angles. With the \texttt{tzplot} package, the user input \texttt{a} is replaced by \texttt{above}, \texttt{b} by \texttt{below}, \texttt{c} by \texttt{center (not centered for the main node option)}, \texttt{br} by \texttt{below right}, and similarly for other abridged strings.

\textbf{Remark:} This is just a \textit{string replacement} that is not related to the function of TikZ. By this we mean that we \texttt{cannot use} these abridged strings to place labels, instead of angels, in the \texttt{tikzpicture} environment without using the related \texttt{\textbackslash{tz}<...>} macros.

The macros related to this issue are as follows:
- \texttt{dots: \texttt{tzdot(s)}, \texttt{tzcdots(s)}, \texttt{tznodedot(s)}}
- \texttt{coordinates: \texttt{tzcoor(s)}, \texttt{tzcoorsquick}}
- \texttt{intersection points: \texttt{tzXpoint, tzvXpoint, tzhXpoint, tzLFnXpoint}}
- \texttt{plot coordinates: \texttt{tzplot}, \texttt{tzplotcurve}}
- and their starred versions.

2.2.6 New styles for middle arrow tips

Some styles for drawing the the \textit{middle arrow tips} on a path are predefined.
- \texttt{-->--}: (controllable) middle arrow tip
  - \texttt{\textbackslash{settzmidarrow}} controls the positions and styles of middle arrow tips
- \texttt{--o--}: the circle middle arrow tip
- \texttt{--x--}: the cross middle arrow tip
- \texttt{--/---}: the diagonal middle arrow tip

You can use these styles to draw middle arrow tips as follows:

\begin{verbatim}
\tzlines[-->--,red](0,0)(1,0)(3,1); \hspace{1cm} % default=0.5
\tzlines[-->--=0.7](0,0)(1,0)(3,3); \hspace{1cm} % work like \draw[-->--=0.7]
\draw[-->--,red](0,0) -- (1,0) -- (3,1);
\draw[-->--=0.7](0,0) -- (1,0) -- (3,1);
\end{verbatim}

2.3 Remarks

Some macros have been modified in order to add new features. This does not cause any harm to existing users.
- \texttt{\textbackslash{tznode}}: to add new option \texttt{\textbackslash{node.code}}
  - This allows you to use full features (including \texttt{foreach}) of the TikZ’s \texttt{node} operation.
- \texttt{\textbackslash{tzframe}, \texttt{\textbackslash{tzcircle}, \texttt{\textbackslash{tzellipse}}}}: to add new option \texttt{\textbackslash{code.append}}
  - Now you can use \texttt{even odd rule} to draw rings with these macros.

Some macros are “experimental” and their syntax may change in the future.
**Styles of middle arrow tips in the istgame package**  The package *istgame* to draw game trees predefined the styles of the middle arrow tips including \(\text{-}\text{-}\), \(\text{-} \text{o}\rightarrow\), and \(\text{-} \text{x}-\).

- The style \(\text{-}\text{-}\) defined in *istgame* and \(\text{-}\rightarrow\rightarrow\) defined in *tzplot* are a little different from each other in the default values.
  - Still you may want to use the style \(\rightarrow\) instead of \(\rightarrow\rightarrow\). In that case, you can do like this:
    \[
    \text{	exttt{tikzset{->/.style=\{-->\}}}}
    \]
  - *However,* it is important to understand that changing the style to \(\rightarrow\) may override the style of \(\rightarrow\) defined in other package, depending on which package is loaded first.
  - The best way to use \(\rightarrow\) instead of \(\rightarrow\rightarrow\) is to upload the *istgame* and follow the instruction of the manual.

- The styles \(\text{-} \text{o}\rightarrow\) and \(\text{-} \text{x}-\) differ in definition and usage from the styles \(\text{-} \text{o}\) and \(\text{-} \text{x}-\) of the *istgame* package.
Part II
Getting Started

3 An Intuitive Introduction I: Basics

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

3.1 Lines: Basics: $\texttt{tzline}(0,0)(3,1)$

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

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

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

\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}

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

3.2 Dots: Basics

3.2.1 A circle dot: $\texttt{tzcdot}(0,0)$

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

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

You can change of the size of a dot by specifying the \texttt{radius} of the circle, like, for example, $\texttt{tzcdot}(0,0)(3pt)$. 7
3.2.2 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 (\textit{diameter} or \textit{minimum size}) is 2.4pt.

\begin{quote}
\texttt{\tzdot: node dot}
\begin{tikzpicture}
\draw [green] (0,0) node [\tzdot,minimum size=6pt] {}; \\
% tzdot style is predefined
\end{tikzpicture}
\end{quote}

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

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

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

3.3 Coordinates: Basics: \texttt{tzcoor(0,0)(A)}

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

\texttt{\begin{tikzpicture}
\zhelplines(4,3)
\tzcoor(0,0)(A)
\tzcoor(60:3cm)(B)
\tzline[->](A)(B)
\tzdot(A)(5pt)
\tzcoor(0,1)(C)
\tzcoor*[fill=none](4,2)(D)
\tzline[dashed](C)(D)
\end{tikzpicture}}

The starred version \texttt{tzcoor*} prints a filled node dot at a specified coordinate.

\texttt{\begin{tikzpicture}
\zhelplines(4,3)
\tzcoor(0,0)(A)
\tzcoor(60:3cm)(B)
\tzline[->](A)(B)
\tzdot(A)(5pt)
\tzcoor*(0,1)(C)
\tzcoor*[fill=none](4,2)(D)
\tzline[dashed](C)(D)
\end{tikzpicture}}
3.4 Curves: Basics

3.4.1 \tzto(0,0)(4,2)

\tzto connects two points with a line or a curve using the to operation of Ti\texttt{KZ}.

\begin{tikzpicture}
\helplines(4,3)
\tzto(0,0)(4,2)
\tzto[bend right,dashed](0,1)(3,2)
\tzto[out=90,in=-90,->,blue](0,2)(4,1)
\end{tikzpicture}

\tzto[bend right](0,1)(3,2) \ %\ works\ like:
\draw[bend right] (0,1) to (3,2);

3.4.2 \tzbezier

\tzbezier draws a Bézier curve with one or two control points from the first coordinate to the last coordinate. The style \texttt{tzshowcontrols} predefined in the package reveals the control point(s).

\begin{tikzpicture}
\helplines(4,3)
\tzbezier[blue,thick](0,0)(2,0)(4,2)
\tzbezier[->,tzshowcontrols](0,2)(1,3)(3,0)(4,1)
\end{tikzpicture}

\tzbezier[blue](0,0)(2,0)(4,2) \ %\ works\ like:
\draw[blue] (0,0) .. controls (2,0) .. (4,2);
\tzbezier(0,2)(1,3)(3,0)(4,1) \ %\ works\ like:
\draw (0,2) .. controls (1,3) and (3,0) .. (4,1);

3.4.3 \tzparabola

\tzparabola draws a parabola controlled by several options of Ti\texttt{KZ}'s \texttt{parabola} operation. The macro \texttt{tzparabola} accepts two or three coordinate arguments to draw a parabola and the parabola bends at the second coordinate if it exists.

\tzparabola(0,0)(2,4) \ %\ works\ like:
\draw (0,0) parabola (2,4);
\tzparabola(2,0)(3,3)(4,1) \ %\ works\ like:
\draw (2,0) parabola bend (3,3) (4,1);
3.5 Adding text: Nodes and placement

3.5.1 `\textnode{3,1}{text}[right]`

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

```latex
\begin{tikzpicture}
\tzhelplines(4,3)
\textnode(0,0){A}
\textnode*(1,1){text}
\textdot*(2,2)
\textnode(2,2){Text}[above right,blue]
\end{tikzpicture}
```

In this package, macros related to ‘dots’ or ‘coordinates’ (like `\textdot`, `\textcoor`), can optionally have ‘label nodes,’ while macros related to ‘lines’ and ‘curves’ (like `\textline`, `\textto`, `\textbezier` and `\textparabola`) optionally have ‘main nodes’ or ‘text nodes.’
• There is one exception: \texttt{\texttt{tzshoworigin}. \texttt{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 18.3 on page 141 for more details.)

### 3.5.3 Abbreviations of Ti\texttt{z} basic placement option styles: \texttt{a, r, ar, bl, etc.}

You can use abbreviations (or aliases) \texttt{a} for \texttt{above}, \texttt{c} for \texttt{centered}, \texttt{r} for \texttt{right}, \texttt{bl} for \texttt{below left}, and so on to place main nodes. (Again, you cannot use angels to place main nodes.)

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

### 3.6 Labeling dots and coordinates

#### 3.6.1 \texttt{tzdot, \texttt{tzcdot}}

To add a label to a dot generated by \texttt{tzdot} or \texttt{tzcdot}, you should specify, \emph{right after a coordinate}, \texttt{\{<label>\}} followed by \texttt{[<angle>]} (90 degree or above by default in Ti\texttt{z}).

Remember that the order of the arguments is \texttt{(<coor>{<label>}{<angle>})}. To change the size of a dot, you need to specify the last option \texttt{(<dimension>)} after all the other arguments.

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

\texttt{tzcdot\*[1,2]{C}[180]\(2pt\)} \texttt{% works like:}
\texttt{\draw[fill] (1,2) circle (2pt) node [label={180:C}] {};};

#### 3.6.2 \texttt{tzcoor}

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

\textbf{Remark:} Remember the order of arguments is \texttt{\(<coor>\)(<name>){<label>}[<angle>\].

\begin{tikzpicture}
\tzcoor\(1,2\){C}[180]\(2pt\) \% works like:
\draw[fill] (1,2) circle (2pt) node [label={180:C}] {};
\end{tikzpicture}

\texttt{% syntax: simplified}
\texttt{\tzcoor\(<coor>\)(<coor name>){<label>}[<label opt>[<angle>]]}
\texttt{% defaults}
\texttt{(<>)(<>){}[]}
\texttt{% <m> means 'mandatory'}

You can see the full syntax of \texttt{tzcoor} in Section 9.1 on page 52.
3.6.3 \texttt{\textbackslash{tzc}oor*}

The starred version \texttt{\textbackslash{tzc}oor*} designates a coordinate and prints a node dot with a label around the designated point like \texttt{\textbackslash{tz}dot*} does.

% syntax: simplified
\texttt{\textbackslash{tzc}oor*([<dot opt>])(<coor>)(<coor name>){<label>}[<label opt>]<angle>][<dot size>]

3.7 Adding text next to lines or curves

3.7.1 \texttt{\textbackslash{tz}line}

\texttt{\textbackslash{tz}line} accepts two mandatory coordinates. To add text to a line segment, just specify the optional arguments \texttt{<text>} and \texttt{[<node opt>]} \textit{in-between} the two coordinates. The \texttt{[<node opt>]} is \texttt{[above,midway]}, by default.
The optional argument \{<text>\} 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}
\tzhelplines(4,3)
\tzline(0,3)(4,0){demand}
\tzline(0,0)(3,3){supply}[r]
\end{tikzpicture}

3.7.2 \texttt{\tzto}

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

\begin{tikzpicture}
\tzhelplines(4,3)
\tzto(0,0){route A}(3,0)
\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}

\texttt{\tzto}(0,0){route A}(3,0) \texttt{\% works like:}
\draw (0,3) to node \{above\} {route A} (3,0);

The optional argument \{<text>\} 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}
\tzhelplines(4,3)
\tzto[bend right=15](0,3)(4,0){demand}
\tzto[bend right=10](0,0)(3,3){supply}[r]
\end{tikzpicture}

3.7.3 \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 \{<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}
3.7.4 \tzparabola

You can add text to a parabola drawn by \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}
\zhelplines(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}

4 An Intuitive Introduction II: Repetition of Coordinates

4.1 Linking many coordinates: Semicolon versions

4.1.1 \tzlines: Connected line segments

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

Remark: : Without the semicolon \;, an error occurs with the the error message:

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

With the optional argument \{<text>\} followed by \[<node opt>\] 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 \[midway\]).
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 `[<node opt>]`.

**Remember** the repeating pattern is the triple `{<coor>}{<text>}[<node opt>]` in that order. **DO NOT FORGET** to indicate when the repetition ends by typing a semicolon. So it will look like `{<coor>}{<text>}[<node opt>] ..repeated.. `{<coor>}{<text>}[<node opt>]`.

### 4.1.2 `\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 16.1.1 on page 114 for more details.)

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

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

The entire repetition will look like `{<coor>}{<text>}[<node opt>] ..repeated.. `{<coor>}{<text>}[<node opt>]`.

**DO NOT FORGET** to indicate when the repetition ends by typing a semicolon.
\begin{tikzpicture}[sloped,auto]
\zhelplines(4,3)
\tzonegon[dashed]
 (0,0){up} (1,2){down} (3,0){up} (4,1){Weight}[r] ;
\end{tikzpicture}

4.1.3 \texttt{\textbackslash tspath*: Filling area}

\texttt{\textbackslash tspath} accepts an arbitrary number of coordinates to form a path, like \texttt{\textbackslash tzone} 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}
\zhelplines(4,3)
\tzonepath[draw](1,3)(0,2)(2,0)
 % semicolon
(3,1)(4,2)(2,3);
\end{tikzpicture}

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

\begin{tikzpicture}
\zhelplines(4,3)
\tzonepath[blue](0,3)(4,0)
\tzonepath[pattern=crosshatch](1,3)(0,2)(2,0)(3,1)(4,2)(2,3);
\end{tikzpicture}

\begin{tikzpicture}
\zhelplines(4,3)
\tzonepath[blue](0,3)(4,0)
\tzonepath[pattern=bricks,preaction={fill=brown}](0,0) (1,2) (3,0) (4,1);
\end{tikzpicture}

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

The starred version \texttt{\textbackslash tspath*} takes the default options \texttt{fill=black!50, fill opacity=.3, and text opacity=1} to fill the area.

% syntax: simplified
\texttt{\textbackslash tspath*}[<opt>][<coor>][<label>][<angle>]..repeated..(<coor>){<label>}[<angle>];
% defaults
 *<fill=black!50,fill opacity=.3,text opacity=1>(<m>) repeated;
 % <m> means mandatory

The macros \texttt{\textbackslash tspath} and \texttt{\textbackslash tspath*} are much more flexible. See Section 14.1 on page 99 for more details.
How to change the fill opacity with \texttt{tzpath*} is not discussed in this introduction, but one example is given below. (See Section 14.2 on page 101 for more details.)

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

4.2 Many dots: Semicolon versions

4.2.1 \texttt{tzcdots(*)}

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

\begin{tikzpicture}
\zhelplines(4,3)
\tzcdots(0,0)(1,1)(2,1)(3,2)(4,3);
\tzcdots*[red](0,3)(1,3)(2,3)(3,3)(4,2); \textit{semicolon}
\end{tikzpicture}

Each coordinate can be labeled by specifying the optional argument \texttt{\{<label>\}} followed by [\texttt{\langle angle\rangle}]. You can also change the size (radius) of the dots by specifying the last parenthesis option \texttt{\langle dot radius\rangle} after the semicolon.

\begin{tikzpicture}
\zhelplines(4,3)
\tzcdots(0,0)(1,1){B}(2,1)(3,2){D}\{-90\}(4,3);(5pt)
\tzcdots*[fill=red](0,3){A}(1,3){B}(2,3){C}\{-90\}(3,3)(4,2){E}[blue]0;\langle3pt\rangle \textit{radius}
\end{tikzpicture}

4.2.2 \texttt{tzdots(*)}

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

\begin{tikzpicture}
\zhelplines(4,3)
\tzdots(0,0)(1,1){B}(2,1)(3,2){D}\{-90\}(4,3);(5pt)
\tzdots*[fill=red](0,3){A}(1,3){B}(2,3){C}\{-90\}(3,3)(4,2){E}[blue]0;\langle3pt\rangle \textit{radius}
\end{tikzpicture}

% syntax: minimum
\texttt{tzdots*\langle coor\rangle\langle coor\rangle..repeated..\langle coor\rangle;}

18
Each coordinate can be labeled by specifying the optional argument \{<label>\} followed by \[<angle>\]. You can also change the size (diameter) of the dots by specifying the last parenthesis argument \(<\text{dot size}>\) after the semicolon.

\begin{tikzpicture}
thelplines(4,3)
dots(0,0)(1,1)(2,1)(3,2)(4,3);
dots*[red](0,3)(1,3)(2,3)(3,3)(4,2); % semicolon
\end{tikzpicture}

The options \{<label>\} and \[<angle>\] 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 \((<\text{coor}>)\)(<name>)\{<label>\}[<angle>]. The first two parenthesis arguments are mandatory and others are optional. The pattern is repeated until \texttt{tzcoors} meets a semicolon ;.
The starred version \texttt{tzcoors*} does one more thing: to print node dots.

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

4.3.2 \texttt{tzcoorsquick}

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

\begin{tikzpicture}
\helplines(4,3)
\coorss(0,0)(A)
(1,1)(Best)
(2,1)(Case)
(3,3)(Done)
(4,2)(End);
\end{tikzpicture}

4.3.3 \texttt{tzcoorsquick*}

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

\begin{tikzpicture}
\helplines(4,3)
\coorss*(0,0)(A)
(1,1)(Best)
(2,1)(Case)
(3,3)(Done)
(4,2)(End){END!}[blue];
\end{tikzpicture}
4.4 plot coordinates: Semicolon versions

4.4.1 \texttt{\textbackslash tzplot*: Mark dots with [mark=*]}

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

\begin{tikzpicture}
  \draw[help lines] (4,3);
  \tzplot*(0,0)
  (1,1)
  (2,1)
  (3,3)
  (4,2); \% semicolon
\end{tikzpicture}

\begin{tikzpicture}
  \draw[help lines] (4,3);
  \tzplot*(0,0)(1,1)(2,1);
  \draw [mark=*] plot coordinates {(0,0)(1,1)(2,1)};
\end{tikzpicture}

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

\begin{tikzpicture}
  \draw[help lines] (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}

4.4.2 \texttt{\textbackslash tzplot: Lines with [tension=0]}

\texttt{\textbackslash 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}
  \draw[help lines] (4,3);
  \tzplot (0,0)
  (1,1)
  (2,1)
  (3,3){Done}
  (4,2){END!}[[blue]0];
\end{tikzpicture}

\begin{tikzpicture}
  \draw[help lines] (4,3);
  \tzplot(0,0)(1,1)(2,1);
  \draw plot coordinates {(0,0)(1,1)(2,1)};
\end{tikzpicture}
4.4.3 \texttt{\textbackslash tzplot*[draw]}: Lines with dots

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

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

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

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

\begin{tikzpicture}
\tzhelplines(4,3)
\tzplotcurve(0,0)
(1,1)
(2,1)
(3,3)
(4,2){END!}[[blue]0];
\end{tikzpicture}

% syntax: simplified
\texttt{\textbackslash tzplotcurve}[<opt>]{<tension>}"<path name>"
\begin{itemize}
\item[] \texttt{(<coor>)\{<label>\}[<angle>].repeated..\{<coor>\}\{<label>\}[<angle>]};
\end{itemize}
% defaults
[smooth,tension=1]{1}"<path name>".repeated..(){};
% <m> means mandatory

\texttt{\textbackslash tzplotcurve}(0,0)(1,1)(2,1); % works like:
\texttt{\textbackslash 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}
\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]{.55}{(1,3)(A)(4,0)}; % TikZ default
\tzplotcurve[dashed]{}{0}{(1,3)(A)(4,0)};
\end{tikzpicture}
5 An Intuitive Introduction III: Plotting Functions

5.1 Axes

5.1.1 \tzaxes

\begin{tikzpicture}[scale=.5]
\tzhelplines(-1,-1)(8,5)
\tzaxes(-1,-1)(8,5) % basics
\end{tikzpicture}

\tzaxes draws the x-axis from \(x1\) to \(x2\) and the y-axis from \(y1\) to \(y2\).

If \((x1,y1)\) 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, [right] for x-axis and [above] for y-axis).

\begin{tikzpicture}[scale=.5]
\tzhelplines(-1,-1)(8,5)
\tzaxes[draw=blue](8,5){\$x\$}{\$y\$} % basics
\end{tikzpicture}

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 \((0,0)\).

\begin{tikzpicture}[scale=.5]
\tzhelplines(-1,-1)(8,5)
\tzaxes(-1,-1)(8,5){\$Q\$}{\$P\$}[l]
\tzaxes[dashed,-]\(2,1\)(-1,-1)(8,5) % shift
\end{tikzpicture}

5.1.2 \tzaxes*

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

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

\begin{tikzpicture}[scale=.5]
  \tzhelplines(-1,-1)(8,5)
  \tzhshoworigin
  \ttxaxes(-1,-1)(8,5)
\end{tikzpicture}

\texttt{tzhshoworigin*} prints a node dot with the size of 2.4pt (by default) at the origin.

\begin{tikzpicture}[scale=.5]
  \tzhelplines(-1,-1)(8,5)
  \tzhshoworigin*
  \ttxaxes(-1,-1)(8,5)
\end{tikzpicture}

\texttt{tzhshoworigin*}(<coor>){<text>}[<node opt>] prints a node dot and text around (<coor>), by default (0,0). (Notice that the place where the <text> is printed at is not by \texttt{angle}. Instead, you can use the abbreviations of Ti\texttt{kZ} basic placement options such as \texttt{a}, \texttt{l}, \texttt{br}, etc. See Section 1.2, for more details.)

\begin{tikzpicture}[scale=.5]
  \tzhelplines(-1,-1)(8,5)
  \tzhshoworigin*{<0,1>}
  \ttxaxes(-1,-1)(8,5)
  \ttxaxes[blue]<7,4>(8,5)(-1,-1)
  \tzhshoworigin*(7,4){<0,2>}[ar]
\end{tikzpicture}

Notice that, in the previous example, the two axes intersect at (7,4) by the shift option \texttt{<7,4>}.

5.1.4 \texttt{tzaxisx, tzaxisy}

\texttt{tzaxisx} draws an x-axis from <x1> to <x2>.

% syntax: simplified
\texttt{\texttt{tzaxis}}\texttt{[<opt>]}<y-shift>{<x1>}{<x2>}{<text>}[<node opt>]
% defaults
[->,>=stealth]<0>{<m>}{<m>}{}[right]

\texttt{tzaxisy} works similarly for the y-axis, except for the axis label position: above by default.
5.2 Ticks

5.2.1 \zticks

\zticks{\langle x\text{-tick places}\rangle}{\langle y\text{-tick places}\rangle} 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, \zticks uses the \texttt{foreach} operation of Ti\textit{K}Z.)

5.2.2 \zticks\*

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

You can change the length of tick marks, for example, like (-2pt:3pt) and (-5pt:10pt) as shown in the following example.
5.2.3 \texttt{\textbackslash{tzticksx}}(*), \texttt{\textbackslash{tzticksy}}(*)

\texttt{\textbackslash{tzticksx}} and \texttt{\textbackslash{tzticksy}} prints x-tick labels and y-tick labels, respectively.

\texttt{\textbackslash{tzticksx}} and \texttt{\textbackslash{tzticksy}}* suppress tick labels for their corresponding axes, like \texttt{\textbackslash{tzticksx}}*.

You can see more details on \texttt{\textbackslash{tzticks}} and its friends in Chapter 19 on page 144.

5.3 Projections on the axes

5.3.1 \texttt{\textbackslash{tzprojx}}(*), \texttt{\textbackslash{tzprojy}}(*)

\texttt{\textbackslash{tzprojx}} draws a dotted line (by default) from a specified coordinate to its projection point on the x-axis and prints text around ([below] by default) the projection point.

\texttt{\textbackslash{tzprojy}} works similarly but for the projection point on the y-axis.

\texttt{\textbackslash{tzprojx}}* does one more thing. It prints a node dot (with \texttt{\textbackslash{tzdot}}*) at a specified coordinate. \texttt{\textbackslash{tzprojy}}* works similarly but for the projection point on the y-axis.
\begin{tikzpicture}
\tzhelplines(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}

5.3.2 \texttt{\textbackslash tzproj\textdagger}

\texttt{\textbackslash tzproj} combines \texttt{\textbackslash tzprojx} and \texttt{\textbackslash tzprojy}. And \texttt{\textbackslash tzproj\textdagger} combines \texttt{\textbackslash tzprojx\textdagger} and \texttt{\textbackslash tzprojy\textdagger}.

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

5.4 Plot functions

5.4.1 \texttt{\textbackslash tzfn}

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

% syntax: simplified
\texttt{\textbackslash tzfn\texttt{\}[\texttt{\node dot}]}
% defaults
[]""{\texttt{\textbackslash m}}{\texttt{\textbackslash m}}{\texttt{\textbackslash m}}
% \texttt{\textbackslash m}: mandatory

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

\begin{tikzpicture}
\tzfn[dashed]{((x-2)^2+1)[1:4]{g(x)}}{r} % works like:
\draw [dashed] plot [domain=1:4] \( \{x,((x-2)^2+1)) \) node [right] {\( g(x) \)};
\end{tikzpicture}

You can name a path formed by \texttt{\textbackslash tzfn} by specifying the optional argument \"\texttt{\textbackslash path name}\" right before the mandatory curly brace argument \{\texttt{\textbackslash fn of} \texttt{\textbackslash x}\}. The name of a path is used to find intersection points.
Remark: If the curly brace mandatory argument consists of only a macro name like {\Foo}, the macro name Foo (without the backslash) is automatically assigned to the name of the path. (See Section 21.1.4 on page 154, for more details.)

5.4.2 \texttt{\textbackslash tzhfnat}, \texttt{\textbackslash tzhfn}: Horizontal lines

\texttt{\textbackslash tzhfnat} accepts a value of \( y \) and draws a horizontal line (the graph of a constant function) at \( y \) from left to right of the current bounding box, by default, unless you specify the optional argument \texttt{[<from:to>]}.% syntax \begin{tikzpicture} \tzhelplines(4,3) \tzhfnat[0] \tzhfnat[dashed]{[1:4]Two}{[1,at start]} \tzcoors{(0,2)}{(0,3)}{(0,4)}; \tzhfn[blue]{(0,3)}{(Three)[b,near end]} \tzhfn[->]{(0,3)}{(Four)[r]} \end{tikzpicture}

\texttt{\textbackslash 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.

5.4.3 \texttt{\textbackslash tzfvenat}, \texttt{\textbackslash tzfven}: Vertical lines

\texttt{\textbackslash tzfvenat} draws a vertical line at \( x \) from bottom to top of the current bounding box by default.

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

\texttt{\textbackslash tzfven(<coor>)} accepts a coordinate, instead of the value of \( x \). It draws a vertical line at the value of \( x \) coordinate of \(<\texttt{coor}>\), ignoring the value of \( y \) coordinate.
5.4.4 \texttt{tzLFn}: Linear functions

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

5.5 Intersection points

5.5.1 Naming paths

In Ti\kZ, you can find \textit{intersection} points when two \textit{named paths} intersect. The name of a path is usually given by the option \texttt{[name path=<path name>]} in Ti\kZ.

With the package \texttt{tzplot}, you can name a path by specifying an optional argument within quotation marks such as \texttt{"<path name>"}. (Of course, you can also name a path in usual Ti\kZ \textit{way}, like \texttt{[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:

\begin{verbatim}
\tzline[<opt>]{<path name>}(<coor>)...
\tzlines... {<path name>}(<coor>)...
\tzto... {<path name>}(<coor>)...
\tzfn... {<path name}>{<fn of \x>}...
\tzLFn... {<path name>}(<coor>)...
\end{verbatim}

\texttt{\tzline[dashed]"foo"(1,1)(3,3)} % works like \begin{verbatim}
\draw [dashed,name path=foo](1,1) -- (3,3);
\end{verbatim}

In most cases, the quote optional arguments for naming paths are placed \textit{immediately before the first mandatory argument} of the \texttt{tzplot} macros.

5.5.2 \texttt{tzXpoint(*)}: Intersection points of two paths

For example, \texttt{\tzXpoint{path1}{path2}(A)} finds intersection points of \texttt{path1} and \texttt{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}
\draw[help lines] (4,3) -- (0,0) -- (3,2);
\draw (0,3) -- (3,0) -- (4,3);
\node at (X) [label=above:E] {};
\node at (X) [label=above:E] {};
\end{tikzpicture}

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

\begin{tikzpicture}[scale=.7,font=\footnotesize]
\draw[help lines] (5,5) -- (0,0) -- (5,5);
\draw[<->,blue] (4-\x) -- (\IC -\x);
\node at (\IC) [label=above:A] {};
\node at (\IC) [label=above:A] {};
\end{tikzpicture}

Remark: You have to expect Ti\textit{k}Z to take a few seconds (or less) to find intersection points.

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

The macro \texttt{tXpoint} uses \(<\text{coor}>\), while \texttt{tXpointat} uses the value of \(x\). Here the \(y\) coordinate of \(<\text{coor}>\) is not important. \texttt{tXpoint} is useful when you do not know the exact value of \(x\) coordinate of \(<\text{coor}>\). The starred version \texttt{tXpoint*} additionally prints a node dot at the (first) intersection point.

\begin{tikzpicture}[scale=.7,font=\scriptsize]
\draw[help lines] (5,5) -- (0,0) -- (5,5);
\node at (\IC) [label=above:A] {};
\node at (\IC) [label=above:A] {};
\end{tikzpicture}

5.5.4 \texttt{tXpointat(*)}, \texttt{tXpoint(*)}: Horizontal intersection points

\texttt{tXpointat\{<path>\}\{<y>\}(A)} works like \texttt{tXpointat\{<path>\}\{<x>\}(A)}, but it uses the value of \(y\) instead of \(x\). The starred version \texttt{tXpointat*} additionally prints a node dot at the (first) intersection point.
\texttt{tzhXpoint} 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{tikzpicture}[scale=.7,font=\scriptsize]
\helplines(5,5)
\axes(5,5)
\def\Fx{((x-1)^2)}
\tznfn{Fx}{0:3} % name path = Fx (automatically)
\tzhXpointat{Fx}{2.5}(A)
\tzhXpoint{Fx}(1,2.8)(B)
% x=1 is ignored
\tztproj*[<->](A){$f^{-1}(2.5)$}{y=2.5}$
\tzdot(B){$B$}[0]
\end{tikzpicture}

You can see more details on \texttt{tzhXpoint} and its friends in Chapter 24 on page 170.

5.6 Tangent lines and secant lines

5.6.1 \texttt{tztangentat}

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

% syntax: simplified
\texttt{tztangentat\{path\}\{\texttt{x}\}\{domain\}\{text\}\{node opt\}}
% defaults
\[\{\texttt{m}\}\{\texttt{m}\}\{\texttt{m}\}\{}\]
% \texttt{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 tangent line by specifying the optional arguments \{\texttt{text}\} and \{\texttt{node opt}\}, after the domain.

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

5.6.2 \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.

% syntax: simplified
\texttt{tztangent\{path\}\{coor\}\{from:to\}\{text\}\{node opt\}}
% defaults
\[\{\texttt{m}\}\{\texttt{m}\}\{}\]

31
\begin{tikzpicture}[scale=.7]
  \zhelplines(5,5)
  \tzparabola[thick,blue]"curve"(0,1)(2,3)(4,2)
  \tzvXpoint+[blue]{curve}(1,0)(A){A}[135]
  \tzvXpoint+[blue]{curve}(3,0)(B){B}[45]
  \ztangent[red]{curve}(A)[0:3]
  \ztangent[red]{curve}(B)[1:5]{tangent at $B$}[b]
\end{tikzpicture}

See Section 25.2 on page 178 for more details on $\ttztangent$ and $\ttztangentat$.

5.6.3 $\tttzsecant$, $\tttzsecant$

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

\begin{tikzpicture}[scale=.7]
  \zhelplines(5,5)
  \tzaxes(5,5)
  \tzparabola"curve"(0,1)(2,3)(5,1)
  \tzsecantat{curve}{1}{2}
  \tzsecantat{curve}{1}{3}
  \tzsecantat{curve}{1}{4}
\end{tikzpicture}

$\tttzsecant$ works like $\tttzsecantat$, but it accepts two coordinates instead of two values of $x$.

$\tttzsecant\{\text{<path>}\}{\text{<coor1>}}{\text{<coor2>}}$ draws a secant line segment of $\text{<path>}$ from the $x$ coordinate of $\text{<coor1>}$ to the $x$ coordinate of $\text{<coor2>}$, ignoring $y$ values of the coordinates, on the behind layer by default.

\begin{tikzpicture}[scale=.7]
  \zhelplines(5,5)
  \tzaxes(5,5)
  \tzparabola"curve"(0,1)(2,3)(5,1)
  \tzsecant{curve}(1,0)(2,0)
  \tzsecant{curve}(1,0)(3,0)
  \tzsecant{blue,dashed}{curve}(1,0)(4,0)
\end{tikzpicture}

You can extend or shorten a secant line by specifying the domain $[\text{<from:to>}]$, which is an optional argument. If you specify the domain, $\tttzsecant$ draws a secant line over the domain.

You can also add some text next to the secant line by specifying the optional arguments $\{\text{text}\}$ and $\{\text{node opt}\}$.

\begin{tikzpicture}[scale=.7]
  \zhelplines[use as bounding box](5,5)
  \tzaxes(5,5)
  \tzparabola"curve"(0,1)(2,3)(5,1)
  \tzsecant{curve}(1,0)(2,0)
  \tzsecant[red]{curve}(1,0)(3,0)[0:5]{secant}[r]
  \tzsecantat{blue,dashed}{curve}{1}{4}[0:5]{blue}[r]
\end{tikzpicture}

See Section 25.1 on page 175 for more details on $\tttzsecant$ and $\tttzsecantat$. 

32
6 Examples: Economics

6.1 Markets

6.1.1 Market equilibrium: step by step

Step 1 Determine the size of the graph.

\begin{tikzpicture}[x=0.05cm,y=0.05cm,scale=.7]
% to avoid 'dimension too large' error: z/y=0.05cm
\tzhelplines\[step=.5cm\](110,110)
\tzaxes(110,110){$Q$}{$P$}
\tzto[bend right=15]"dem"(0,100)(100,0){$D$}[a]
\tzto[bend right=15]"supp"(0,10)(100,90){$S$}[ar]
\tzXpoint*(dem){supp}(eqm){$E$}
\tzproj(eqm){$Q^*$}{$P^*$}
\end{tikzpicture}

Step 2 Draw the demand and supply curves. Here, we are using tzto.

\begin{tikzpicture}[x=0.05cm,y=0.05cm,scale=.7]
% to avoid 'dimension too large' error: z/y=0.05cm
\tzhelplines[step=.5cm](110,110)
\tzaxes(110,110){$Q$}{$P$}
\tzto[bend right=15]"dem"(0,100)(100,0){$D$}[a]
\tzto[bend right=15]"supp"(0,10)(100,90){$S$}[ar]
\tzXpoint*(dem){supp}(eqm){$E$}
\tzproj(eqm){$Q^*$}{$P^*$}
\end{tikzpicture}

Step 3 Find an equilibrium point and name it. Use the starred version \tzXpoint* to print a dot and then label the point.

\begin{tikzpicture}[x=0.05cm,y=0.05cm,scale=.7]
% to avoid 'dimension too large' error: z/y=0.05cm
\tzhelplines[step=.5cm](110,110)
\tzaxes(110,110){$Q$}{$P$}
\tzto[bend right=15]"dem"(0,100)(100,0){$D$}[a]
\tzto[bend right=15]"supp"(0,10)(100,90){$S$}[ar]
\tzXpoint*(dem){supp}(eqm){$E$}
\tzproj(eqm){$Q^*$}{$P^*$}
\end{tikzpicture}
Step 4  If necessary, use \texttt{tzproj} to draw projection lines with text around the projection points.

\begin{tikzpicture}[x=0.05cm,y=0.05cm,scale=.7]
% to avoid 'dimension too large' error: x/y=0.05cm
%\zhelplines[step=.5cm](110,110)
\taxes(110,110)\{\$Q\}$\{\$P\}$
\ztoto[bend right=15]"dem"(0,100)(100,0)\{\$D\}\{a\}
\ztoto[bend right=15]"supp"(0,10)(100,90)\{\$S\}\{ar\}
\tzXpoint+(dem)(supp)(eqm)\{\$E\}$
\tzproj(eqm)\{\$Q^*\}$\{\$P^*\$
\end{tikzpicture}

6.1.2 Tax incidence: step by step

Step 1: Determine the size of the graph.

\begin{tikzpicture}[scale=.035,font=\scriptsize]
%\zhelplines[step=10cm](120,120)
\taxes(120,120)\{\$Q\}$\{\$P\}$
\end{tikzpicture}

\begin{tikzpicture}[scale=.035,font=\scriptsize]
%\zhelplines[step=10cm](120,120)
\taxes(120,120)\{\$Q\}$\{\$P\}$
% step 2
\def\demA{100-\x}
\def\suppA{10+\x}
\tzfn\demA[0:100]\{\$D\}\{ar\} % name path = \demA
\tzfn\suppA[0:90]\{\$S\}\{r\} % name path = \suppA
\tzXpoint+(demA)(suppA)(E)\{\$E\}$
% step 3
\tzfn[dashed]"suppB"\{suppA+20\}[0:80]\{\$S'\}$\{ar\}
\tzXpoint+(demA)(suppB)(newE)\{\$E'\}$
\tzproj(nevE)\{\$Q'\}$\{\$P_{d}\}$
% step 4
\tzvXpoint(suppA)(newE)(vX)
% step 5
\tzprojy(vX)\{\$P_s\}$
\tzprojy(E)\{\$P^c\}$
% step 6
\tzpath*[red](E)(newE)(vX);
% step 7
\tznode($E$)!10cm!(E-|0,0)\{pin={-70:SWL}\}
\end{tikzpicture}

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 both of 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.

6.2 Firms

6.2.1 Cost curves

\begin{tikzpicture}[scale=.4,font=\scriptsize]
\tzhelpines(10,10)
\tzaxes(10,10){\$q\$}{\$AC\$\$AVC\$\$MC\$}[align=center]
\tzparabola"MC"(.5,4)(2,2)(6,9){\$MC\$}[a]
\tzXpointat{MC}{5}(A)
\tzXpointat{MC}{3}(B) % (B-2) will be used!
\tzdots*(A)(B-2);
\tzparabola(2,7)(A)(8,9){\$AC\$}[r]
\tzparabola(1,4.5)(B-2)(8,8){\$AVC\$}[r]
\end{tikzpicture}

6.2.2 Equilibrium of a competitive firm

\begin{tikzpicture}[scale=.4,font=\scriptsize]
%\tzhelpines(10,10)
\tzaxes(10,10){\$q\$}{\$P\$\$AC\$\$MC\$}[align=center]
\tzparabola"MC"(2,2)(6,9){\$MC\$}[a]
\tzXpointat{MC}{4}(A)
\tzparabola"AC"(2,7)(A)(8,9){\$AC\$}[r] % (A): minAC
\tzfnat[blue]"price"{6}[0:9]{\$P\$}[1,at start]
\tzXpoint*{price}{MC}(E)
\tzprojx(E){\$q^*\$}
\tzvXpoint{AC}(E)(ACeqm) % point on AC in equilibrium
\tzprojy(ACeqm){\$AC(q^*)\$}
\tzpath*[red](E-|0,0)(E)(ACeqm-|0,0);
\tznode(2,5){\$\pi^*\$}
\end{tikzpicture}
6.2.3 Monopoly equilibrium

\begin{tikzpicture}[scale=.5,font=\scriptsize]
\tzaxes(10,10){Q}{$P$}
\def\DD{8-\x}
\def\MR{8-2*\x}
\def\MC{\x}
\tzfn\DD[0:8]{D}
\tzfn\MR[0:4.5]{MR}
\tzfn\MC[0:7]{MC}
\tzXpoint*(\MR)\(E)
\tzvXpoint*(\DD)(E)
\tzproj(EE){Q^M}{P^M}
\tzXpoint*(\DD)\(C)
\tzpath*(EE)(C)(E);
\tznode(C){SWL}
\tzpath[blue](0,8)(EE)(EE-|0,0);
\tznode(1,6){C.S.}
\tzpath[pattern=horizontal lines](0,0)(E)(E-|0,0);
\tznode(1,2){P.S.}
\end{tikzpicture}

6.3 Consumers: Budget lines and indifference curves

\begin{tikzpicture}[scale=.3,font=\scriptsize]
\tzaxes(15,12){x}{$y$}
\def\bgt{-3/4*\x+9} % 3x+4y=36
\tzfn\bgt[0:12]
\tzvXpoint*(\bgt)(6,0)(A){A}[45]
\tzplotcurve"ICC"(3,9)(A)(12,3); % trial and error
\tzproj(A){x^*}{y^*}
\tzticks{12}{9}
\end{tikzpicture}

\begin{tikzpicture}[scale=.15,font=\scriptsize]
\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}
\tzvXpoint*\IC[12]{A}[45]
\tztangent[blue,densely dashed]{\IC}(A){5:20}
\end{tikzpicture}
6.4 Production Possibility Curves

\begin{tikzpicture}[scale=.5,font=\scriptsize]
\tzhelplines(9,8)
\tzaxes(9,8){\$X\$}{\$Y\$}
\tzto[out=-10,in=105]{PPC}(0,7)(6,0){PPC}[ar]
\tzvXpointat*(PPC)(2){E}
\tzvXpointat*{tan}(4){F}
\tzplotcurve[densely dashed,red](2,7)(F)(8,4);
\end{tikzpicture}

6.5 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){\$x_2\$}[l]{\$y_2\$}[b]
\tshoworigin{\$O_1\$}
\tshoworigin(8,6){\$O_2\$}[ar]
\tto[bend right](2,5)(7,2){\$u_1\$}[r]
\tto[bend left](1,4)(6,1){\$u_2\$}[b]
\end{tikzpicture}

6.6 Growth

\begin{tikzpicture}[scale=.5,font=\scriptsize]
%\tzhelplines(10,8)
\tshoworigin
\taxes*(10,8){\$k\$}{\$y\$} % bounding box
\def\Fk{3*(\x)^{(1/3)}}
\def\sFk{2*(\x)^{(1/3)}}
\def\ndk{7/9*\x}
\tfn\Fk{0:9}{y=f(k)}[r] % name path=Fk
\tfn\sFk{0:9}{sf(k)}[r] % name path=sFk
\tfn\ndk{0:9}{(n+\delta)k}[r] % name path=ndk
\txpoint*(sFk){ndk}(E)[2] % name path=ndk
\tproj(YE){$y^*$}{$k^*$} % 2nd X point
\tprojy(B){sf(k_0)}{y^*}
\tprojy(C){(n+\delta)k_0}{y^*}
\end{tikzpicture}
6.7 Liquidity trap

LM curves are drawn with two paths. To shift the IS curve, \texttt{<shift coor>} is used. See Section 22.1 on page 160 for more details.

The money demand \((m_d)\) curve is drawn with one path. To do this, \texttt{<code.append>} is used. See Section 13.1 on page 86 for more details.
6.8 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}
\tzplot[mark=ball](1,3)(2,2)(3,1){A}[0];
\tzplot[mark=\ast](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}[r];
\tzlines(1,2)(2,1)(3,3){B}[r];
\tzlines(1,1)(2,3)(3,2){C}[r];
\end{tikzpicture}
Part III
Points, Lines, and Curves

7 Getting Ready

7.1 Styles: \texttt{tzdotted}, \texttt{tzdashed}, \texttt{tzhelplines}

The styles \texttt{tzdotted}, \texttt{tzdashed}, and \texttt{tzhelplines} are defined as follows:

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

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

7.2 \texttt{tzhelplines}, \texttt{tzhelplines*}

\texttt{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 \texttt{tzhelplines*} uses the grid as a bounding box.

\begin{verbatim}
% syntax: minimum
\tzhelplines(<coor>)
% syntax: full
\tzhelplines[<opt>](<coor1>)(<coor2>)
% defaults
  [help lines,tzdotted=10]<m>(<m>)()
% (<m>): mandatory argument
\end{verbatim}

Here, \(<m>\) stands for a \textit{mandatory} argument.

\begin{verbatim}
\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);
\end{verbatim}

By default, \texttt{tzhelplines} prints grid with 10 dots per 1cm. \texttt{tzhelplines} with the option value \texttt{[tzdotted=<n>]} prints \(<n>\) dots per 1cm. (That is, the default value is \texttt{tzdotted=10}.)

40
With the option value, [tzdotted=<n>/<d>], \tzhelplines prints <n> dots per <d>cm. Similarly for tzdashed.

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

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

8.1 \texttt{\textbackslash tzcdot(*)}: A small circle

A dot is usually expressed by a small circle. \texttt{\textbackslash tzcdot} prints a circle dot $\cdot$. The starred version \texttt{\textbackslash tzcdot*} prints a filled circle dot $\bullet$. The \textit{radius} of the circle is 1.2pt, by default.

\begin{verbatim}
% syntax: minimum
\texttt{\textbackslash tzcdot}(<coor>)
% syntax: medium
\texttt{\textbackslash tzcdot*}(<coor>)\{<label>\}[<angle>](<radius>)
% syntax: full
\texttt{\textbackslash tzcdot*}[<opt>]<shift coor>(<coor>)\{<label>\}[<label opt>][<angle>](<radius>)
% defaults
-\{ solid, thin, tzcdot=1.2pt \}<>(<m>)[]\{\}(1.2pt)
% tzcdot is a predefined key (in this package).
% <m>: mandatory
\end{verbatim}

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

\noindent\textbf{How to change the size} \textbf{There are THREE WAYS} to change the \textit{radius} of a circle dot drawn by \texttt{\textbackslash tzcdot}.

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

\begin{verbatim}
\texttt{\textbackslash tzcdot(0,0)} \% is an abbreviation of:
\texttt{\textbackslash draw (0,0) circle (1.2pt); \% default radius=1.2pt}
\end{verbatim}

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

2. You can use the key-value option \texttt{[tzcdot=<dim>], like \textbackslash 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{verbatim}
\texttt{\textbackslash tzcdot*(1,1)} \% works like:
\texttt{\textbackslash draw [fill] (1,1) circle [radius=1.2pt]; \% default radius=1.2pt}
\end{verbatim}

\begin{verbatim}
\texttt{\textbackslash tzcdot*[tzcdot=3pt]} \% works like:
\texttt{\textbackslash draw [fill] (0,0) circle [radius=3pt];}
\end{verbatim}

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

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

\textbf{How to label} \quad You can add a label to a specified coordinate by adding the optional argument \texttt{\{<label>\}} immediately after \texttt{(<coor>)}.
You can also change the \texttt{\{<label>\}} position by the option \texttt{\{<angle>\}}.

\begin{tikzpicture}
\tzhelplines(4,3)
\tzcdot(0,0){A}
\tzcdot*[red](1,0){\textbf{B}}[45]
\tzcdot*[red,fill=green](2,1){green}[blue,0](10pt)
\tzcdot*[fill=red,text=blue](4,3){D}[2*3pt]
\end{tikzpicture}

In \textit{TikZ}, the distance from the coordinate center to a label does not depend on the size of circle dots.

\textbf{How to change colors} \quad With the first optional argument \texttt{\{<opt>\}}, you can change the color of a dot. You can also change the color of a label, as shown in the following example.

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

\textbf{Shift} \quad Dots can be shifted by specifying the optional argument \texttt{\{<shift coor>\}} immediately before \texttt{(\<coor>\)}. The \textit{empty} shift option \texttt{<>} is \textit{not} allowed.
8.2 $\texttt{\textbackslash tzcdots(*)}$: Multiple circle dots

The macro $\texttt{\textbackslash tzcdots}$ 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.
- Without the semicolon ;, an error occurs with the the error message:
  
  Package tzplot Error: You may have forgotten a semicolon here or above!

The starred version $\texttt{\textbackslash tzcdots*}$ 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 \(\langle\text{coor}\rangle\{<label>\}[<angle>]\). In \TeX, \langle<angle>\rangle\) can have the label option like \([<\text{label opt}>\text{angle}]\).

\begin{tikzpicture}
  \tzhelplines(4,3)
  \tzcdots(0,0){A}
  (1,1)
  (2,1){C}[0]
  (3,2){D}[-90]
  (4,0){E}[[red,draw=45];
  \tzcdots*(0,3)(1,2){B}[-135](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 \texttt{tzcdot} key, like \texttt{\tzcdots[tzcdot=3pt]...} If both options are used the key-value option wins.

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

\begin{tikzpicture}
  \tzhelplines(4,3)
  \settzcdotradius{3pt}
  \tzcdots*(0,0)(1,0){3pt};
  \tzcdots*(1,1)(2,1){1pt};
  \tzcdots*(2,2){5pt};
  \tzcdots*(3,3)(4,3){3pt};
\end{tikzpicture}

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

\begin{tikzpicture}
  \tzhelplines(4,3)
  \settzcdotradius{3pt}
  \tzcdots*(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 \texttt{<shift coor>} option immediately before the first coordinate.
8.3 \texttt{\textbackslash tdot(*)}: A single node dot

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

The starred version \texttt{\textbackslash tdot*} prints a filled dot $\ast$.

\begin{tikzpicture}
\helplines(4,3)
\tzhelplines[red] (0,0){A}(1,1){C}(3,2){D}[0](4,0){E};
\tzhelplines<0,1>(0,0){A}(1,1){C}(3,2){D}[0](4,0){E};
\end{tikzpicture}

\textbf{8.3.1 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 \texttt{\textbackslash tdot(*)}.

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

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

2. The simplest way is to use the \textit{last parenthesis optional argument}, like \texttt{\textbackslash tdot(0,0)(5pt)}, which yields the same result as in \texttt{\textbackslash tdot[tzdot=5pt](0,0)}. If both options are used, the \texttt{\textbackslash tdot} (or \textit{minimum size}) option overwrites the last parenthesis option.

46
3. You can use the macro \settzdotsize to change the size (or diameter) of all node dots drawn by \tzdot*. It is effective within the \tikzpicture environment unless changed again.

\begin{tikzpicture}
\helplines(4,2)
\tzdot(0,0)
\tzdot*[tzdot=8pt](1,1)
\tzdot*(2,1)(4pt) \tzdot*(3,0)(6pt)
\end{tikzpicture}

\begin{tikzpicture}
\helplines(4,2)
\settzdotsize(8pt)
\tzdot(0,0)
\tzdot(1,1)
\tzdot*[2pt](2,1)(4pt) \tzdot*[3pt](3,0)
\end{tikzpicture}

8.3.2 How to label

You can add a label to a specified coordinate by specifying the optional argument \{<label>\} immediately after the coordinate \(<\text{coor}>\). You can also change the label position by the option \[<\text{angle}>\] or \[<\text{[label opt]angle}>\] following \{<label>\}.

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

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

String replacement (See also Section 2.2.5 on page 5.)

- In TikZ, to place labels you can use \texttt{<angle> or the positional words such as \{\texttt{above}, \texttt{below}, \texttt{center}\} (not \texttt{centered} for the main node option), \texttt{below right}, and so on.\n- Just to avoid frequent coding errors, from the version 2 of the \texttt{tzplot} package, you can use the abridged strings \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{br}\}, and so on.
- So \texttt{[blue,draw]-45}, \texttt{[blue,draw]below right}, and \texttt{[blue,draw]br} give the same result.

Unlike \texttt{\tzcdot}, the \texttt{\tzdot}'s label position depends on the size of a circle node. In TikZ jargon, \{\texttt{<label>}\} is in a \textit{label node} for a \textit{main node} that is a circle node with no text in it, so \texttt{<label>} moves accordingly as the main node dot gets bigger or smaller.
8.3.3 How to change colors and shapes

With the first optional argument [node opt], you can change 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].

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

8.3.4 How to move: shift

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

\begin{tikzpicture}
\draw [help lines] (0,0) grid (4,3);
\tzdot [regular polygon](0,0){A}(10pt)
\tzdot* [red]<0,2>(1,0){\textbf B-shifted}[45]
\tzdot* [red,green,rectangle](2,1){green}[0](2*10pt)
\tzdot* [tzdot=4*7pt]<1,-1>(3,2){big}[center]
\tzdot* [fill=red,star](4,3){D}(4*3pt)
\end{tikzpicture}
8.3.5 Comparison: \tzdot and \tzcdot

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

\begin{tikzpicture}[xscale=1.6,yscale=.8]
\zhelplines(3,3)
\tzcdot(0,0)(3pt) \hspace{1cm} % distorted
\tzdot(1,0)(6pt)
\tzcdots*(1,1)(2,1)(3,1);(2pt) \hspace{1cm} % distorted
\tzdots*(1,2)(2,2)(3,2);(4pt)
\end{tikzpicture}

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>\settztzdotsize [\tzdot=2.4pt]</td>
<td>% diameter</td>
</tr>
<tr>
<td>circle</td>
<td>\tzcdot</td>
<td>\tzcdots</td>
<td>\settztzcdotradius [\tzcdot=1.2pt]</td>
<td>% radius</td>
</tr>
</tbody>
</table>

Remark:
- In Ti\kern Z, a ‘node’ is ‘not’ affected by ‘scaling’ unless the Ti\kern Z option \texttt{transform shape} is used together. \tzdot is also useful for labelling a large dot.
  - In \tzdot, \texttt{<label>} is a label in a \texttt{label node} for a node dot (as a \texttt{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 \texttt{<label>} in \tzcdot does not depend on the size of dots.
- The package tzplot takes \tzdot as a \texttt{standard dot}, not \tzcdot. So, you can apply the \texttt{THREE WAYS} (on page 46) to change the size of any standard dots.

8.4 \tzdots(*): Multiple node dots

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

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

Remark:
- DO NOT FORGET to enter ‘;’ at the end of iteration.
- Without the semicolon ;, an error occurs with the error message:

\begin{verbatim}
! Package tzplot Error: You may have forgotten a semicolon here or above!
\end{verbatim}

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

\% syntax: minimum
\tzdots*\{<coor>\}{<coor>}..repeated..\{<coor>\} ;
\% syntax: full
\tzdots*[\texttt{node opt}]\{<coor>\}\{<coor>\}{<label>}[\{\texttt{label opt}\}angle] \hspace{1cm} ..repeated. (){}[] ; (<dot size>)
\% defaults
\*[\texttt{tzdot=2.4pt}]<> (){}[] ..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. (To avoid frequent coding errors, you can also use string replacement instead of angles. See also Section 2.2.5 on page 5

How to change the size of dots There are THREE WAYS of changing the diameter of node dots, as discussed in Section 8.3.1 on page 46.

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{\tdots[tzdot=3pt]}... If both options are used the tzdot option style wins.

3. You can also use the macro \texttt{\settztzdotsize}. 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>]}.
\% \tzdots: color
\begin{tikzpicture}
\textbf{Ben} \[\text{red}\] -90 \(2,1\)(3,0);
\tzdots*\[\text{blue}\] -90 \(3,2\)\{C\}\{4,2\}\{D\}; \(8pt\)
\tzdots*\[\text{green}\] \(0,0\)(1,1)\{A\}\{B\}\{C\}\{D\};
\end{tikzpicture}

Remark:
\begin{itemize}
\item \texttt{[<node opt>]} is the option of a \textit{main node} and \texttt{[<label opt>]} is the option of a \textit{label node}.
\item \texttt{[<label opt>]} is used in the form of \texttt{[\langle<label opt>\rangle angle]}], like \texttt{[\textcolor{red}{90}].}
\item You can control all labels together using \texttt{every label/.style} as in the following examples:
\end{itemize}

\% \ätzdots: every label/.style\begin{tikzpicture}
\textbf{Ben} \[\textcolor{red}{-90}\] \(2,1\)(3,0);
\tzdots*\[\text{blue}\] \(3,2\)\{C\}\{4,2\}\{D\}; \(8pt\)
\tzdots*\[\text{green}\] \(0,0\)(1,1)\{A\}\{B\}\{C\}\{D\};
\end{tikzpicture}

\% every label/.style\begin{tikzpicture}[every label/.style={draw,text=red}]
\textbf{Bob} \[\textcolor{blue}{135}\] \(3,2\)\{$C_1$\} \{$N_o$\};
\end{tikzpicture}

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

\% \ätzdots: shift\begin{tikzpicture}
\textbf{Ace} \[\textcolor{red}{-90}\] \(2,1\)\{Bob\}\{C\}\{135\};
\tzdots*\[\text{green}\] \(3,2\)\{$C_1$\} \{$N_o$\};
\end{tikzpicture}
9 Coordinates

9.1 \tcoor and \tzcoor*

9.1.1 \tcoor

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

\begin{verbatim}
\tcoor(0,0)(A) \% is an abbreviation of:
  \path (0,0) coordinate (A);
  \coordinate (A) at (0,0);
\end{verbatim}

Here, <m> stands for ‘mandatory.’ \tcoor 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).

\begin{verbatim}
\tcoor(0,0)(A){$A$}[0] \% works like:
  \path (0,0) coordinate [label={0:$A$}] (A);
\end{verbatim}

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

The starred version \tzcoor* works like \tzcoor with one exception. It prints a ‘node dot’ of the size 2.4pt, by default, at a specified coordinate.

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

% works like:
\path (0,0) coordinate (A);
EZdot*(0,0)
% works like:
\path (0,0) coordinate (A);
EZdot*(0,0){A}[right]

Changing the color and size of a dot You can change the color of a dot by specifying [dot opt], which is, in fact, TikZ’s node option. To change the size of dots, you can apply the THREE WAYS (see Subsection 8.3.1 on page 46).

% \tzcoor*
\begin{tikzpicture}
\zhelplines(4,3)
\tzcoor*(0,0)(A){A_1}[right] % TikZ default: 90 or above
\tzcoor*(30:3cm)(B){B_2}[draw,blue]
\draw (A) -- (B);
\end{tikzpicture}

Shift The optional argument <shift coor> works just like in \tzcoor. The empty shift option <> is not allowed.
9.2 `\tzcoors` and `\tzcoors*`: Semicolon versions

9.2.1 `\tzcoors`

The macro `\tzcoors` takes an *arbitrary number of pairs* of coordinates and their names as arguments. For example, `\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){O} (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 `{<label>}` and `[<angle>]` immediately after `{<name>}`.

\begin{tikzpicture}
\tzcoors(0,0)(A){Ace}[font=\LARGE\ttfamily]-90
(2,1)(B){\textbf{Bob}}[blue]135
(3,2)(C){$C_1$
$N_0$}[align=center]0;
\draw (A) -- (B) -- (C);
\end{tikzpicture}

By the option `<shift coor>`, all specified coordinates are shifted. The *empty* shift option `<>` is *not allowed*. 

% 	zcoors*: shift
\begin{tikzpicture}
\tzcoors*(0,0)(A){A_1} %
\tzcoors*(2,1)(B){B_2} %\end point\end{align*}0
\tzline(A)(B)
\end{tikzpicture}

% syntax: minimum
\tzcoors(<coor>(<name>)...repeated..<coor>(<name>) ;
% syntax: full
\tzcoors <shift coor>(<coor>(<name>),<label>){<label opt>[angle]}...epeated.. (){}[] ;
% defaults
<> (<m>)(<m>){}[] ..repeated.. (){}[] ;
9.2.2 \tzcoors*

The starred version \tzcoors* takes an arbitrary number of pairs of coordinates and names as mandatory arguments to print node dots at the coordinates.

The full repeating pattern is \texttt{((\textless \text{coor}\textgreater ) (\textless \text{name}\textgreater ) \{\textless \text{label}\textgreater \}\{\textless \text{angle}\textgreater \}}. It is required to type a colon ‘;’ to indicate when the iteration of coordinates ends.

You can label each dot by specifying the optional arguments \texttt{\{\textless \text{label}\textgreater \}} and \texttt{\{\textless \text{angle}\textgreater \}} after the pair \texttt{((\textless \text{coor}\textgreater ) (\textless \text{name}\textgreater ))}.

You can change the dot color by \texttt{\{\textless \text{dot opt}\textgreater \}} and the label color by \texttt{\{\textless \text{label opt}\textgreater \}. You can apply the THREE WAYS (on page 46) to change the dot size. The simplest way of changing the dot size is to specify the last (even after the semicolon) parenthesis option \texttt{\{\textless \text{dot size}\}}.

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

\subsection{\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 (by default) of the coordinates.

\begin{tikzpicture}
\helplines(4,3)
\tzcoorsquick[red](0,0)(A)(1,1)(B)(2,1)(C)(3,3)(D);
\setdotsize{6pt}
\tzcoorsquick[blue]<.5,0>(0,0)(A)(1,1)(B)(2,1)(C)(3,3)(D);
\end{tikzpicture}

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 \texttt{empty} option \texttt{<>} is not allowed.

\begin{tikzpicture}
\helplines(4,3)
\tzcoorsquick(0,0)(A)
(1,1)(Ben)
(2,1)(Cate){}
(3,2)(Daniel);
\tzcoorsquick[blue]<0,1>
(0,0)(A)
(1,1)(Ben)
(2,1)(Cate){}
(3,2)(Daniel);
\end{tikzpicture}

\subsection{\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.
A label can be suppressed by the empty braces {}. You can change the dot size using the **THREE WAYS** (on page 46). You can shift the coordinate by specifying `<shift coor>` immediately before the first coordinate. The *empty* shift option `<>` is not allowed.

**Remark:** The first optional argument `[<dot opt>]` 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.

9.4 `\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`. 

```
% syntax: minimum
\tzcoorsquick*{(coor)(<name>)(name)}..repeated..{(coor)(<name>)(name)} ;
% syntax: full
\tzcoorsquick*{<dot opt>}{<shift coor>}{<coor>}{<name>}{<label>}{<label opt>}{angle}{..repeated.. }();{}();{}();{}();{2.4pt}
% defaults
- [tzdot=1.2pt ]<> (m)(m)();{};()();{}; (2.4pt)

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

\begin{tikzpicture}
  \zhelplines(4,3)
  \tzcoorsquick*{(0,0)(A) (1,1)(Ben) (2,1)(Cate) (3,2)(Daniel)}\tzset{every label/.style={blue}}
  \tzcoorsquick*[blue](0,1)
  \tzcoorsquick*{(0,0)(A) (1,1)(Ben) (2,1)(Cate) (3,2)(Daniel)\[blue\];(6pt)}
\end{tikzpicture}

% \tzcoorsquick*: every label/.style
\begin{tikzpicture}
  \zhelplines(4,3)
  \tzset{every label/.style=\{draw,red\}}
  \tzcoorsquick*[fill=none,blue,very thick]
  (0,0)(A)(1,1)(B-1)(2,1)(C)(3,3)(D)\[blue\]-45;\[8pt\]
\end{tikzpicture}

% syntax: minimum
\tzgetxyval{(coor)}{(\macroXval)}{(\macroYval)}
% syntax: full
\tzgetxyval{(m)}{(m)}{(m)}
10 Plot Coordinates: \tzplot: Semicolon Versions

\tzplot takes an arbitrary number of coordinates as arguments. Internally, \tzplot uses the plot coordinates operation of \Tikz.

Each (\<coor\>) can be followed by the optional arguments \{\<label\>\} and [\<angle\>] to label the coordinate. This is a semicolon version and the whole repeating pattern is the triple (\<coor\>){\<label\>}[\<angle\>]. It is required to type a semicolon ';' to indicate when the coordinate iteration ends.

The macro \tzplot draws connected line segments that link specified coordinates.

The starred version \tzplot* prints dot marks at specified coordinates, without drawing line segments connecting the coordinates, by default. \tzplot* is equivalent to \tzplot[draw=none,mark=*]. The style tzmark is defined as follows:

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

\tzmarksize is the radius of a mark and the default is 2pt as in Ti\kZ. The value of \tzmarksize can be changed by the macro \settztzmarksize, like \settztzmarksize{3pt}. 58
10.2 \texttt{tzplot*}: Dots and marks

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

\begin{verbatim}
\texttt{tzplot*}(0,0)(1,2)(2,2)(3,3); \% works like:
  \texttt{draw} [draw=none,mark=*] plot coordinates { (0,0)(1,1)(2,2)(3,3) } ;
\end{verbatim}

\begin{verbatim}
\% \texttt{tzplot*}
\begin{tikzpicture}[scale=.8]
  \zhelplines (4,3)
  \texttt{tzplot*}(0,0)(1,1)(2,1)(3,3);
  \texttt{tzplot*[mark=\texttt{o}](0,3)(1,3)(2,2)(3,1)} ; \% semicolon
  \texttt{tzplot*[red](0,2)(1,2)(2,0)(3,2)} ;
\end{tikzpicture}
\end{verbatim}

**Labels, marks, and mark size** You can also add labels to specified coordinates with the optional arguments \{<label>\} and [\langle angle\rangle\} immediately after each (\langle coor\rangle\}.

\begin{verbatim}
\texttt{tzplot*}(0,0){\texttt{A}}[90\texttt{]}(1,2)(2,2)(3,3){\texttt{D}}[0]; \% works like:
  \texttt{draw} [draw=none,mark=*] plot coordinates { (0,0)(1,1)(2,2)(3,3) }
    (0,0) node [label={90:\texttt{A}\texttt{\}}] {}
    (3,3) node [label={0:\texttt{D}\texttt{\}}] {} ;
\end{verbatim}

There are **three ways** to change the mark size.
1. The simplest way is to use the parenthesis optional argument (\langle mark size\rangle\), \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.

\begin{verbatim}
\% \texttt{tzplot*: label, size}
\begin{tikzpicture}[scale=.8]
  \zhelplines (4,3)
  \texttt{tzplot*}(0,0){\texttt{A}}[-135\texttt{]}(1,1)(2,1)(3,3){\texttt{D}}[0](1mm);
  \texttt{tzplot*[mark=\texttt{o},tzmark=6pt](0,3)(1,3)(2,2)(3,1)} ;
  \texttt{settzzmarksize}{4pt}
  \texttt{tzplot*[red](0,2){\texttt{A}}(1,2){\texttt{B}}(2,0){\texttt{C}}(3,2){\texttt{D}};}
\end{tikzpicture}
\end{verbatim}

**Remark:** You can use strings such as \texttt{a}, \texttt{b}, \texttt{ar}, and so on, instead of angles, from the version 2 of the \texttt{tzplot} package. These strings are replaced by the corresponding positioning words such as \texttt{above}, \texttt{below}, \texttt{above right}, and so on. (See also Section 2.2.5 on page 5 for more details.)

\begin{verbatim}
\% \texttt{tzplot*: label: strings instead of angles}
\begin{tikzpicture}[scale=.8]
  \zhelplines (4,3)
  \texttt{tzplot*}(0,0){\texttt{A}}[\texttt{bl}](1,1)(2,1)(3,3){\texttt{D}}[1mm];
  \texttt{tzplot*[mark=\texttt{o},tzmark=6pt](0,3)(1,3)(2,2)(3,1)} ;
  \texttt{settzzmarksize}{4pt}
  \texttt{tzplot*[red](0,2){\texttt{A}}(1,2){\texttt{B}}(2,0){\texttt{C}}(3,2){\texttt{D}};}
\end{tikzpicture}
\end{verbatim}
With \tzplot*, you can draw line segments by giving the TikZ’s option \texttt{draw} in the first bracket optional argument, like \texttt{\tzplot*[draw]}.

\begin{tikzpicture}[scale=1]
\tzhelplines(4,3)
\settzmarks{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{Remark:} In TikZ, the mark shapes are affected by scale, xscale, and yscale.

\begin{tikzpicture}
\tzhelplines(4,3)
\settzmarks{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 <> is not allowed.

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

\textbf{Extending path} You can use \texttt{<code.append>} as the last optional argument after a semicolon.

\begin{tikzpicture}
\tzhelplines(4,3)
\tzplot*[draw](0,0)(1,1)(2,2)(C){C}(3,2){D};
< arc (-90:0:1cm) node [a] {ends here!}>
\end{tikzpicture}
\tzplotAtBegin and \tzplotAtEnd are also available. These work with \tzplot*[draw] and \tzplot to extend the paths at the beginning and at the end, respectively. Specifying <code.append> extends the path after \tzplotAtEnd.

```latex
\begin{tikzpicture}
\zhelplines(4,3)
\tzplotAtBegin{(0,2) to [bend right]} % ERROR!!%
\tzplotAtEnd{arc (90:0:1cm) node [b] {here?}}
\tzplot*[draw,blue](0,0)(1,1)(2,2){C}(3,2){D};
\tzplot*[draw,red]<.5,1>(0,0)(1,1)(2,2){C}(3,2){D};
\end{tikzpicture}
```

10.3 \tzplot: Lines

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

```latex
\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) } ;
```

\tzplot* is equivalent to \tzplot[draw=none,mark=*].

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

```latex
% \tzplot: line
\begin{tikzpicture}[scale=.8]
\zhelplines(4,3)
\tzplot(0,0)(1,1)(2,1)(3,3);
\tzplot[dashed,mark=0](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}
```

To close the path of \tzplot you can use the option smooth cycle in the first bracket optional argument [<opt>] or in the second bracket optional argument [<plot opt>].

```latex
% \tzplot: smooth cycle
\begin{tikzpicture}[scale=.8]
\zhelplines(4,3)
\tzplot(0,0)(1,1)(2,1)(3,3);
\tzplot[dashed,mark=0,smooth cycle](0,3)(1,3)(2,2)(3,1);
\tzplot[red,mark=*,mark options={blue}][smooth cycle]%% default tension=0(0,1)(1,2)(2,0)(3,2);
\end{tikzpicture}
```

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.
10.4 \texttt{tzplot}: Curves

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

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

To plot curves, the macro \texttt{tzplotcurve} is provided. Basically, \texttt{tzplotcurve} is the tension=1 version of \texttt{tzplot}. (See Section 10.6 on page 64.)

10.5 \texttt{tzplot}: Bars and combs

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

10.6 \texttt{tzplotcurve(*)}

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

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

\begin{verbatim}
% syntax: minimum
\tzplotcurve(<coor>)(<coor>)..repeated..(<coor>) ;
% syntax: full
\tzplotcurve[<opt>]{<tension>} [plot opt]<shift coor>"<path name>
  (coor){{<label>}[<label opt]angle]}
  ..repeated..
(){}[] ; (mark size) <code.append>
% defaults
*[tzmark=2pt]{1}[smooth] <>" (\texttt{draw})
\end{verbatim}

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

\texttt{tzplotcurve[blue, smooth cycle]{2}(0,0)(1,2)(2,2)(3,3); % works like:
\texttt{draw [blue, tension=2] plot [smooth cycle] coordinates { (0,0)(1,1)(2,2)(3,3) } ;

Since \texttt{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, \texttt{(A){here}[above]} represents \texttt{(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.

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

You can change the color of all labels together by adding `[text=<color>]` to the first bracket option list.
The simplest way to change the mark size is to specify `{<mark size>}` immediately after the semicolon `;`. To close the path of `\tzplotcurve`, you can use the TikZ option `smooth cycle` in the first bracket option or in the second bracket option.

**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.

**Shift** Use the optional argument `<shift coor>` before the first coordinate (to be precise, immediately before `^<path name>`, if it 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.

You can also use `\tzplotcurveAtBegin` and `\tzplotcurveAtEnd` to extend the path of `\tzplotcurve` at the beginning and at the end, respectively. Specifying `<code.append>` extends the path after `\tzplotcurveAtEnd`.

name path for intersection points  You can name the path of `\tzplotcurve` by specifying the option "<path name>" immediately before the first coordinate.
\begin{tikzpicture}
\zhelplines (4,3)
\tplotcurve[blue] "IC"(.5,3)(1.5,1.2)(4,.5){\$u\$}[0];
\tplot[draw=red] "bgt"(0,3)(4,0){budget}[-90];
\end{tikzpicture}

11 Nodes

11.1 \tznode and \tznode*

The macro \tznode allows you to put text at a specified coordinate. \tznode expects two mandatory arguments: \texttt{(\textlangle coor\textrangle)} and \texttt{\{\textlangle text\textrangle\}. You can also optionally name a node so that you can refer to the node coordinate later.

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

\begin{itemize}
  \item \tznode\texttt{(0,0)}\{\texttt{text}\} \texttt{\% works like:}
  \texttt{\path (0,0) node {text};}
  \texttt{\% or}
  \texttt{\node at (0,0) {text};}
  \end{itemize}

\begin{itemize}
  \item \tznode\texttt{[draw]} \texttt{(0,0)}\{\texttt{A}}\texttt{\{above right\} \% works like:}
  \texttt{\node [draw] (A) at (0,0) [above right] \{text};}
  \end{itemize}

\tznode* prints the perimeter of a node, which is a rectangle by default.

\begin{itemize}
  \item \tznode*\texttt{(0,0)}\{\texttt{A}}\texttt{\{above right\} \% works like:}
  \texttt{\node [draw] (A) at (0,0) [above right] \{text};}
  \end{itemize}
Putting text  You can use TikZ options in the first bracket optional argument \(<\text{opt}>\) or the second bracket option \([\text{node opt}]\) to put text with different colors, fonts, and so on.

Abbreviations  You can use abbreviations (or aliases) such as \texttt{a} for \texttt{above}, \texttt{l} for \texttt{left}, \texttt{ar} for \texttt{above right}, \texttt{bl} for \texttt{below left}, and so on to indicate where the text of a main node is placed. (See also Section 1.2 on page 2.)

Remark:

- A label node is placed by angles or the corresponding positioning words.
- A main node is placed by the placement words or their aliases.
- You cannot use angles to place main nodes.

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

Shift  You can move the coordinates by specifying the option \(<\text{shift coor}>\) immediately before the coordinate \(<\text{coor}>\). The empty shift option \(<\text{}>\) is not allowed.
Repetition: \texttt{foreach}  The last optional argument \texttt{<node.code>} can be used to iterate over to place multiple nodes.

\begin{tikzpicture}
\node*[fill=blue,text=yellow]<1,-1>(2,3)(B){B};
\node*[circle,text=red]<-2,-1>(4,1)(C){C};
\node*<(1,1),(A){A};
\node*[->,bend left,dashed](B)(C);
\end{tikzpicture}

11.2 \texttt{tznodes} and \texttt{tznodes*: Semicolon versions}  \texttt{tznodes} accepts any number of mandatory pairs of coordinates and their names to defined node coordinates at specified coordinates.

\begin{tikzpicture}
\node*[fill=blue,text=yellow]<1,-1>(2,3)(B){B};
\node*[circle,text=red]<-2,-1>(4,1)(C){C};
\node*<(1,1),(A){A};
\node*[->,bend left,dashed](B)(C);
\end{tikzpicture}

Each pair of a coordinate and a node coordinate name can be followed by the optional arguments \{\texttt{text}\} and \{\texttt{node opt}\} to print node text. Since this is a semicolon version, it is required to
type a semicolon ; to indicate when the repetition ends. \tznodes works similarly to \tzcoors, but the former uses main nodes and the latter uses label nodes.

The starred version \tznodes* is equivalent to \tznodes[draw]. That is, all node perimeters are drawn.

% \tznodes
\begin{tikzpicture}
\tzhelplines(4,3)
\tznodes(0,0)(A){A}
 (1,1)(B){B}
 (2,1)(C){C}[r]
 (3,2)(D){D}[b]
 (4,0)(E){E};
\% semicolon
\tzmnodes(A)(B)(C)(D)(E);
\end{tikzpicture}

% syntax: minimum
\tznodes(<coor>)(<node name>)..repeated..(<coor>)(<node name>);

% syntax: medium
\tznodes(<coor>)(<node name>){<text>}[<node opt]..repeated..()(){}[];

% syntax: full
\tznodes[<every node opt>]{<shift coor>}
 (coor>)(<node name>){<text>}[<node opt>..repeated..()(){}[];

% defaults
[] <> (<m>)(<m>){}[]..repeated..()(){}[];
*[draw]<> (<m>)(<m>){}[]..repeated..()(){}[];

% \tznodes*: retangle nodes (default)
\begin{tikzpicture}
\tzhelplines(4,3)
\tznodes*(0,0)(A){A}
 (1,1)(B){B}
 (2,1)(C){C}[r]
 (3,2)(D){D}[b]
 (4,0)(E){E};
\% semicolon
\tzmnodes(A)(B)(C)(D)(E);
\end{tikzpicture}

% \tznodes*: ellipse nodes
\begin{tikzpicture}
\tzhelplines*(5,4)
\tznodes*[ellipse,fill=yellow,text=blue]
 (0,0)(A){Ann}
 (1,1)(B){Bryn}
 (2,0)(C){C}[r]
 (3,2)(D){Donald}[b,font=\scshape]
 (4,0)(E){End!}[rectangle,fill=green];
\tzmnodes(A)(B)(C)(D)(E);
\end{tikzpicture}
11.3 \tznodedot(*)

\tznodedot names a node and prints a circle node dot (of the size 2.4pt, by default). \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.

You can apply the Three Ways (on page 46) to change the size of node dots. The simplest way is to use the last parenthesis option (\texttt{<dot size>}).
You can move the coordinates of dots by specifying the \texttt{<shift coor>} option immediately before the coordinate. The empty shift option \texttt{<>} is not allowed.

You can use the abridges strings instead of angles. (See also Section 2.2.5 on page 5.)

11.4 \texttt{\textbackslash tznodedots(*)}: Semicolon versions

\texttt{\textbackslash tznodedots} accepts any number of mandatory pairs of coordinates and their names to print multiple node circle dots at specified coordinates. It works just like \texttt{\textbackslash tzdots}, for one exception. It names multiple node coordinates. Everything else is the same as in \texttt{\textbackslash tzdots}.

The starred version \texttt{\textbackslash tznodedots*} prints filled node circle dots.
The simple first bracket option of `\tznodedots(*)` controls dots rather than labels. If you want to control all the labels, you can use TiKZ’s `every label` style as follows:

```latex
% every label/.style
\begin{tikzpicture}
\tzhelplines(4,3)
\begin{scope}[every label/.style={draw{text=red}}]
\tznodedots*(green)
(0,0)(Ace){Ace}[font=\LARGE\textfamily\textbf{]-90}]
(2,1)(Bob){\textcolor{blue}{Bob}}[135]
(3,2)(Cate){\textcolor{red}{\$C_1\$ \textcolor{black}{\$N_o\$}}}[align=\textcenter{0}]
(4,0)(Done!);(5pt)
\end{scope}
\end{tikzpicture}
```

Remark: Comparison of connecting nodes and coordinates:

```latex
% connecting \tznodedots
\begin{tikzpicture}
\tzhelplines(4,3)
\tznodedots(0,0)(A){A}
(1,1)(B)
(2,1)(C){C}[red]-90]
(3,2)(D)[draw,blue]a]
(4,0)(E);(6pt)
\tzlines[thick](A)(B)(C)(D)(E); % connects nodes
\end{tikzpicture}
```

While the previous example shows that `\tzlines` connects nodes, the following example shows that `\tzlines` connects coordinates.

```latex
% connecting \tzcoors
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoors*[fill=none](0,0)(A){A}
(1,1)(B)
(2,1)(C){C}[red]-90]
(3,2)(D)[draw,blue]a]
(4,0)(E);(6pt)
\tzlines[thick](A)(B)(C)(D)(E); % connects coordinates
\end{tikzpicture}
```
11.5 \texttt{\textbackslash tznodetable} and \texttt{\textbackslash tznodetable*}

\texttt{\textbackslash tznodetable} draws and \textit{optionally} names a rectangle node with (black, by default) text in it.

% syntax:
\texttt{\textbackslash tznodetable}[<opt>]<coor>(<node name>){<text>}[<node opt>]
% defaults
\begin{tikzpicture}
% \texttt{\textbackslash tznodetable(2,1)(A){Here}[a]} % works like
% node [draw,rectangle] (A) at (2,1) [above] {Here} ;
\end{tikzpicture}

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

The starred version \texttt{\textbackslash tznodetable*} fills the rectangle with color (black!50 by default) with fill opacity=.3 and text opacity=1, by default.

% syntax:
\texttt{\textbackslash tznodetable*}[<opt>]<shift coor>\texttt{\textbackslash text=black}
\begin{tikzpicture}
% \texttt{\textbackslash tznodetable*}\texttt{\textbackslash text=black!50,fill opacity=.3,text opacity=1} % works like
% \texttt{\textbackslash node}[draw,rectangle] (A) at (2,1) [above] {Here} ;
\end{tikzpicture}

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

74
\begin{tikzpicture}
\helplines[4,3]
\nodeframe(0,0)(A)
\nodeframe[draw,thick](A)(B)
\nodeframe*[green, scale=1.5](2,1){Node frame}
\[label=180:Left, pin=-45:pin\]
\end{tikzpicture}

\begin{tikzpicture}
\helplines[solid](4,3)
\nodebox[draw, circle](0,0)
\nodebox*[draw,fill=yellow](1,2)(A){A}{.1}
\nodebox*[draw, red](3,2)(C){Cat}
\nodecircle[draw, fill=green](4,2)(D){Dog}
\[->,bend left=45\](A.north)(C.90)
\[->,bend right=45, dashed\](B.90)(D.west)
\end{tikzpicture}

\begin{tikzpicture}
\helplines[4,3]
\nodeframe(0,0)(A)
\nodeframe*[fill=yellow](1,2)(B){B}
\nodeframe*[red](3,2)(C){Cat}
\nodebox[fill=green]<-1,-2>(4,2)(D){Dog}
\[->, bend left=45\](A.north)(C.90)
\[->, bend right=45,dashed\](B.90)(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 \isetztzfillopacity.
- The default fill color is black!50. You can use the macro \isetztzfillcolor to change the default fill color.
- With \nodeframe, you can change the color of the perimeter and text with the second bracket option \{<node opt>\}.

11.6 \tznodercircle and \tznodercircle*

\tznodercircle works just like \nodeframe but with a circle node.

% syntax
\tznodercircle[<opt>]<shift coor>(<coor>)(<node name>){<text>}[<node opt>]
% defaults
 [draw, circle]<>(<m>){}{text=black}

\tznodercircle* works just like \nodeframe* but with a circle node.

% syntax
\tznodercircle*[<opt>]<shift coor>
  (<coor>)(<node name>){<text>}[<node opt>]{<fill opacity>}
% defaults
 *{circle, fill=black!50, fill opacity=.3, text opacity=1} <>
 (<m>){} [draw=black, text=black].3}
You can change the fill opacity of `\tznodecircle*` by specifying the last curly brace option `{<fill opacity>}` or using `\settzhillopacity`. You can also move the node by specifying the `<shift coor>` option immediately before the coordinate. The empty shift coordinate option `<>` is not allowed.

11.7 `\tznodeellipse` and `\tznodeellipse*`

`\tznodeellipse` works just like `\tznodeframe` but with an ellipse node.

The starred version `\tznodeellipse*` works just like `\tznodeframe*` but with an ellipse node.
\texttt{\textbackslash \textchar{tzn}ode\textchar{e}llipse} and \texttt{\textbackslash \textchar{tzn}ode\textchar{oval}^\ast} are aliases of \texttt{\textbackslash \textchar{tzn}ode\textchar{ellipse}} and \texttt{\textbackslash \textchar{tzn}ode\textchar{ellipse}^\ast}, respectively.

You can change the fill opacity of \texttt{\textbackslash \textchar{tzn}ode\textchar{ellipse}^\ast} by specifying the last curly brace option \{\texttt{<fill opacity>}\}. You can also move the node by specifying the \texttt{<shift coor>} option immediately before the coordinate. The empty shift option \texttt{<>} is not allowed.

12 Lines

12.1 \texttt{\textbackslash \textchar{tzn}oline}: Connecting two points

\texttt{\textbackslash \textchar{tzn}oline} connects two points with a straight line.

\begin{tikzpicture}
  \node[draw, ellipse, fill=blue!20] (A) at (2,2) {A} ;
  \node[draw, ellipse, fill=red!20] (B) at (4,2) {B} ;
  \node[draw, ellipse, fill=green!20] (C) at (6,2) {C} ;
  \node[draw, ellipse, fill=yellow!20] (D) at (8,2) {D} ;
  \draw[->, bend left=45] (A.north) to (C.north) ;
  \draw[->, bend right=45, dashed] (B.south) to (D.south) ;
\end{tikzpicture}

12.1.1 Line styles

You can use the first optional argument \{\texttt{<opt>}\} to control the line styles.

\begin{tikzpicture}
  \node[draw, ellipse, fill=blue!20] (A) at (2,2) {A} ;
  \node[draw, ellipse, fill=red!20] (B) at (4,2) {B} ;
  \node[draw, ellipse, fill=green!20] (C) at (6,2) {C} ;
  \node[draw, ellipse, fill=yellow!20] (D) at (8,2) {D} ;
  \draw[->, bend left=45] (A.north) to (C.north) ;
  \draw[->, bend right=45, dashed] (B.south) to (D.south) ;
\end{tikzpicture}

\texttt{\textbackslash \textchar{tzn}oline}\{(0,1),(2,1)\} % works like:
\texttt{\textchar{draw} \{(0,1) -- (2,1)\};}
12.1.2 Adding text

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

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

You can use the abbreviations (or aliases) 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 [\langle node opt2\>] immediately after the second coordinate.
12.1.3 Moving lines: shift

You can move the line generated by \texttt{tzone} by specifying the option \texttt{<shift coor>} before the first coordinate (to be precise, immediately before the option \texttt{<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)(3,3){A}[r]
\tzline[blue]<1,0>(0,0)(3,3){A}[r]
\tzline[dashed](1,2)(4,0){C}[r]
\tzline[dashed]<-.5,-.5>(1,2)(4,0){C}[r]
\tdots*(1,2)(4,0);
\end{tikzpicture}

12.1.4 Extending paths

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

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

You can also use \texttt{tzoneAtBegin} and \texttt{tzoneAtEnd} to extend a path of \texttt{tzone} at the beginning and at end, respectively. Specifying \texttt{<code.append>} extends the path after \texttt{tzoneAtEnd}.

\begin{tikzpicture}
\tzhelplines(4,3)
\tzline[blue,->](0,0)(3,3)<arc(0:-60:1cm)>
\tzlineAtBegin{(2,3)} \}
\tzlineAtEnd{ arc (-90:90:.5) node [a] {here?} }
\tzline[dashed,->](1,2)(4,0)
\t<to[bend left] ++(-4,1) node [a] {ends here!}>
\end{tikzpicture}

12.1.5 Naming paths: 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.
12.2 \texttt{\textbackslash{}tzline+:} Relative coordinates

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

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

\begin{verbatim}
\texttt{\textbackslash{}tzline+}(0,1)(2,1) \% works like:
\texttt{\textbackslash{}draw} (0,1) -- ++ (2,1);
\end{verbatim}

\begin{verbatim}
\texttt{\textbackslash{}tzline+}[dashed]"AA"(0,1){A}[red](2,1){B}[right,blue] \% works like:
\texttt{\textbackslash{}draw} [dashed,name path=AA]
(0,1) -- node [red] {A} ++ (2,1) node [right,blue] {B};
\end{verbatim}

\begin{verbatim}
\texttt{\textbackslash{}tzline+}
\begin{tikzpicture}
\tzhelplines(4,3)
\texttt{\textbackslash{}tzline+}[blue,->](0,1)(3,2){A}[r]
\texttt{\textbackslash{}tzline+}[<->](0,2)(3,-2){B}[r]
\texttt{\textbackslash{}tzline+}[dashed]<1,0>(0,2){b}(3,-2){B}[r]
\end{tikzpicture}
\end{verbatim}

\begin{verbatim}
\texttt{\textbackslash{}tzline+: name path}
\begin{tikzpicture}
\tzhelplines(4,3)
\texttt{\textbackslash{}tzline+}[blue,->] "dem" (0,0)(3,3){demand}[r]
\texttt{\textbackslash{}atBegin}(0,1) to [bend left1]
\texttt{\textbackslash{}tzline+}[dashed,-\rightarrow]"supp"(1,2)(3,-2){supply}[ar]
\texttt{\textbackslash{}intersection point}
\texttt{\textbackslash{}tzxpoint+}{dem}{supp}
\end{tikzpicture}
\end{verbatim}

80
12.3 Styles: `tzshorten` and `tzextend`

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

```latex
% tzshorten
\tikzset{%
  tzshorten/.style 2 args ={shorten <=#1, shorten >=#2},
  tzshorten/.default={(2pt)(2pt)}
}

% tzextend (negative tzshorten)
\tikzset{%
  tzextend/.style 2 args ={shorten <=-#1, shorten >=-#2},
  tzextend/.default={(2pt)(2pt)}
}
```

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}`.

12.4 `\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.

```latex
% syntax: minimum
\tzlines(\<coor>\)(\<coor>\)...repeated..(\<coor>\); 
% syntax: medium
\tzlines(\<coor>\){\<text>\}[\<node opt>\]...repeated..(\<coor>\){\<text>\}[\<node opt>\] ; 
% syntax: full
\tzlines[\<opt>\]<shift coor>"<path name>"
  (\<coor>\){\<text>\}[\<node opt>]
  ...repeated.. ()[] ; <code.append>
% defaults
[]<>"" (<m>){}[] ..repeated.. ()[] ; <>
```

`\tzlines(1,1)(2,2)(3,1)(4,3); % works like:
\draw (1,1) -- (2,2) -- (3,1) -- (4,3);`

The whole repeating pattern in `\tzlines` is the triple `(\<coor>\){\<text>\}[\<node opt>].`
\tzlines(1,1)(2,2)(3,1)(C)(4,3){D}[r]; % works like:
\draw (1,1) -- (2,2) -- (3,1) -- node {C} (4,3) node [right] {D};

**Line styles**  Use the first optional argument [\texttt{<opt>}] to control the style of the connected line drawn by \tzlines.

% \tzlines: line style
\begin{tikzpicture}
\tzhelplines(4,3)
\tzlines[blue,<->](0,1)(1,2)(3,1)(4,3); % semicolon
\tzlines[dashed](0,0)(1,1)(2,1)(3,0);
\end{tikzpicture}

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

% \tzlines: adding text
\begin{tikzpicture}
\tzhelplines(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 \texttt{<shift coor>} before the first coordinate or immediately before the option "\texttt{<path name>}" if it exists. (The empty shift option \texttt{<>} is not allowed.)

% \tzlines: shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzlines[red,thick,<->,text=blue]
<0,-1>(0,1)
(1,2)
(3,1)
(4,3){shifted\line A}[align=left,ar] ;
\tzlines[dashed]<1,0>(0,0)(1,1)(2,1)(3,0){line B}[r] ;
\end{tikzpicture}

**Naming paths**  You can name the path of \tzlines by specifying the option "\texttt{<path name>}" immediately before the first coordinate.
Extending paths  You can extend the path of \texttt{tzlines} by writing TikZ code in the last (even after the semicolon) optional argument \texttt{<code.append>}.  

You can close the path with a straight line by \texttt{<--cycle>}.  

You can also use \texttt{tzlinesAtBegin} and \texttt{tzlinesAtEnd} to extend a path of \texttt{tzlines} at the beginning and at the end, respectively. Specifying \texttt{<code.append> extends the path after \texttt{tzlinesAtEnd, if it exists.}
12.5 \texttt{\tzlines+: Relative coordinates: Semicolon version}

The \textit{plus version} \texttt{\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{\tzlines}.} It is also required to enter a \textit{semicolon} to indicate when the coordinate iteration ends.

```latex
\begin{tikzpicture}
\helplines(4,3)
\tzlinesAtBegin{(1,3) to [bend right]}
\tzlinesAtEnd{arc(0:90:1cm)}
\tzlines[red,thick,->,text=blue]
(0,1){Up}
(1,2){Down}[a,sloped]
(3,1){Up}
(4,3) ;
\tzlinesAtEnd{to[bend right] ++(2,2) node{Here!}}
\tzlines[dashed]
(0,0)(1,1)(2,1)(3,0){not here}[b];
< -| ++(-1,-1) node [draw,b] {Ends here!!} >
\end{tikzpicture}
```

\texttt{\tzlines+}: Semicolon version

- \texttt{\tzlines+} connects two or more points with connected line segments, but each coordinate (except the first one) is \textit{relative} to the previous coordinate.
- It is also required to enter a \textit{semicolon} to indicate when the coordinate iteration ends.

```
\tzlines+(0,1)(1,1)(2,-1)(1,2); \% works like:
\draw (0,1) -- ++(1,1) -- ++(2,-1) -- ++(1,2);
```

```
\tzlines+[dashed]"AA"(0,1){A}(1,1){B}(2,-1){C}(1,2){D}[right]; \% works like:
\draw [dashed,name path=AA]
(0,1) -- node (A) ++(1,1)
-- node (B) ++(2,-1)
-- node (C) ++(1,2) node [right] {D} ;
```
\begin{tikzpicture}
\helplines(4,3)
\lines+[red,thick,->,text=blue]
 (0,1){Up}
 (1,1){Down}[a,sloped]
 (2,-1){Up}
 (1,2){line A} ;
\lines+[dashed,auto,text=red]
 (0,0){A}
 (1,1){B}
 (1,0){C}
 (1,-1){line B}[blue,draw,r] ;
\end{tikzpicture}

\begin{tikzpicture}
\helplines*(4,3)
\lines+[red,thick,->,text=blue]
 <1,1> (0,1){Up}
 (1,1){Down}[a,sloped]
 (2,-1){Up}
 (1,2){line A} ;
\coors(0,0)(A)(1,1)(B)(1,0)(C)(1,-1)(D);
\lines+[dashed,auto,text=red]
 <-1,.5> (A){A}
 (B){B}
 (C){C}
 (D){line B}[blue,draw,r] ;
\end{tikzpicture}
13 Connecting Points

13.1 \texttt{\textbackslash tzto: Two points}

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

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

\begin{verbatim}
\texttt{\textbackslash tzto}[\texttt{dashed}](1,1)(3,2) % works like:
\texttt{\textbackslash draw}[\texttt{dashed}](1,1)\texttt{ to (3,2)};
\end{verbatim}

\textbf{Line styles} You can control line styles with the first optional argument \texttt{[<opt>]}.

\begin{verbatim}
\begin{tikzpicture}
\texttt{\textbackslash tzhelplines}(4,3)
\texttt{\textbackslash tzto}[\texttt{blue,thick}](0,1)(3,3)
\texttt{\textbackslash tzto}[\texttt{out=0}](0,0)(4,2)
\texttt{\textbackslash tzto}[\texttt{bend right,dashed}](2,0)(4,1)
\texttt{\textbackslash settzmidarrow}{\texttt{very thick,red}}
\texttt{\textbackslash tzarc}(2,2)(180:45:1cm)
\end{tikzpicture}
\end{verbatim}

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

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

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

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

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

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

You can also use \tztoAtBegin and \tztoAtEnd to extend a path of \tzto at the beginning and at the end, respectively. Specifying <code.append> extends the path after \tztoAtEnd.
13.2 \texttt{\tzto+: Relative coordinates}

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

\begin{verbatim}
\tzto+(1,1)(3,2) % works like:
\draw (1,1) to ++(3,2);
\end{verbatim}

13.3 \texttt{\tztos: Multiple points: Semicolon version}

\texttt{\tztos} takes an arbitrary number of coordinates as arguments to connect them by the TikZ's \texttt{to} operation. \texttt{\tztos} is much more flexible than \texttt{\tzlines}. Since this is a \textit{semicolon version}, you need to enter a \textit{semicolon} to indicate when the coordinate iteration ends.
The quadruple \((<\text{coor}>),[<\text{to opt}>],{<\text{text}>},[<\text{node opt}>]\) is the whole repeating pattern. Here, \([<\text{to opt}>]\) is for the options of Ti\(\text{kZ}\)'s \texttt{to} operation such as [bend right], [bend left], [bend left=<angle>], [out=<angle>,in=<angle>] and so on.

\begin{tikzpicture}
\draw (0,0) to (1,2) to (3,1);
\end{tikzpicture}

\begin{tikzpicture}[blue,->]
\draw (0,0) to (1,2) to (3,1);
\end{tikzpicture}

\textbf{How to connect coordinates} You can use the options of Ti\(\text{kZ}\)'s \texttt{to} operation to connect the coordinates with different types of curves.

\begin{tikzpicture}
\begin{scope}[shift={(1,0)}]
\draw[blue,->] (0,0) to (1,2) to (3,1);
\end{scope}
\end{tikzpicture}

\textbf{Adding text} You can add text next to lines or curves by specifying the options \{<\text{text}>\} and \[<\text{node opt}>\] in-between coordinates or after the option \([<\text{to opt}>]\), if it exists. You can also add text at or around the last coordinate by the last options \{<\text{text}>\} and \[<\text{node opt}>\].
Shift You can move the line or curve of \tztos using the option \texttt{shift coor} before the first coordinate or immediately before the option "\texttt{path name}", if any. (The empty shift option \texttt{<>} is not allowed.)

\begin{tikzpicture}
\helplines(4,3)
\ztos(0,1)[bend right]{A}
  (1,2) {Ben}[red,sloped,a]
  (3,1)[bend right]{Cate}[sloped,near end]
  (4,3) {Name}[draw,blue,a] ;
\end{tikzpicture}

Naming paths: intersections You can name the path of \tztos by specifying "\texttt{path name}" immediately before the first mandatory coordinate.

\begin{tikzpicture}
\helplines(4,3)
\ztos(0,0)
\ztos[blue,<-] (0,1)(1,2)(3,1)(4,3);
\ztos[red,thick]
  (0,0)[bend right]
  (1,1)
  (2,1)[out=-135,in=45]
  (3,0);
\end{tikzpicture}

Extending paths You can extend the path of \tztos from the last coordinate, by writing Ti\texttt{kZ} code in the last (after the semicolon) optional argument \texttt{<code.append>}.
You can also use \tztosAtBegin and \tztosAtEnd to extend a path of \tztos at the beginning and at the end, respectively. Specifying the option \texttt{append} extends the path after \tztosAtEnd.

13.4 \tztos+: Relative coordinates: Semicolon version

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

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

13.5 \tzlink: Two points

\tzlink is a \textit{generalized version} of \tzline. You can decide the style that links two coordinates.

The default link style is to, which can be changed with the second optional argument \texttt{[<code1>]} \texttt{[<code2>]} \texttt{[<code3>]} between two coordinates like, for example, \texttt{[<coor1>][-|]<coor2>}. Another way is to use the first curly brace option \texttt{[<link style>]}. You can also change it by \texttt{settztlinkstyle} like, for example, \texttt{settztlinkstyle[-|]}, which is effective until the end of the \tikzpicture environment.
% syntax: minimum
\tzlink(\langle\text{coor1}\rangle)(\langle\text{coor2}\rangle) % default: to

% syntax: medium
\tzlink[\langle\text{opt}\rangle](\langle\text{coor1}\rangle)[\langle\text{code1}\rangle](\langle\text{coor2}\rangle) % change link style by [\langle\text{code1}\rangle]

% syntax: full
\tzlink{\langle\text{link style}\rangle}[\langle\text{opt}\rangle]\langle\text{shift coor}\rangle"\langle\text{path name}\rangle"
(\langle\text{coor1}\rangle)[\langle\text{code1}\rangle]\langle\text{text1}\rangle[\langle\text{node opt1}\rangle]\langle\text{code2}\rangle
(\langle\text{coor2}\rangle)[\langle\text{text2}\rangle][\langle\text{node opt2}\rangle]\langle\text{code.append}\rangle

% remark:
- \{\langle\text{link style}\rangle\}: % default: to
- [\langle\text{code1}\rangle]: --, -|, to, edge, etc. (optionally more appropriate tikz code)
- \{\langle\text{code2}\rangle\}: + or ++ (possibly with other tikz code)
- \{\langle\text{text1}\rangle\}: node text next to line
- \{\langle\text{text2}\rangle\}: node text at the last coordinate

% defaults
\begin{tikzpicture}
\tzhelplines(4,3)
\tzlink[->](0,0)(1,0)
\tzlink[->,bend right](0,2)(3,0)
\tzlink[->](1,2)[--](4,1)
\end{tikzpicture}

You can also use \settzlinkstyle to change the link style. Its effect is valid until the end of the \tikzpicture environment, unless changed again.

\begin{tikzpicture}
\tzhelplines(4,3)
\settzlinkstyle[to][bend left]
\tzlink[->](0,0)(1,0)
\tzlink[->](0,2)[--](3,0)
\tzlink[->](1,2)[](4,1)
\end{tikzpicture}

The plus version \tzlink+ uses the second coordinate relative (with ++, by default) to the first one. Everything else is the same as in \tzlink.

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

\tzlink(1,0)[-1]++(3,1) % works like:
\draw (1,0) -| ++ (3,1);

\textbf{Link styles} You can change the link style (to by default) using the optional argument [\langle\text{code1}\rangle] immediately after the first coordinate.
\tzlink (1,0) [to[bend right]]{A}{a} \leftrightarrow (3,1) \% or
\tzlink+(1,0) [to[bend right]]{A}{a} (3,1) \% works like:
\draw (1,0) to[bend right] node [above] {A} ++ (3,1);

% \tzlink(+): various link styles
\begin{tikzpicture}
\tzhelplines(4,3)
\tzlink[->,blue,out=90,in=180](0,2)(3,0)
\tzlink[->,dashed,blue](1,2)[to[bend left]](3,-1)
\tzlink[->](0,0)[..controls (1,1) and (3,0)..](4,2)
\tzlink+[red,thick](1,0)[edge[bend left]](2,3)
\end{tikzpicture}

Adding text You can add text next to the line or curve of \tzlink by specifying the option {<text1>} followed by the optional argument [<node opt1>] between two coordinates (or after the option [<code1>] if it exists).

You can name the line or curve of \tzlink by specifying the optional arguments {<text2>} and [<node opt2>] after the second coordinate. The option [<node opt2>] is often for the position of node text at or around the second coordinate.

\begin{tikzpicture}
\tzhelplines(4,3)
\tzlink[->](1,3)\{line A\}(4,3)
\tzlink[blue](0,3)\{line B\}[sloped](3,0)\{E\}[r]
\tzlink[->,red](1,0)\{Start\}[at start,l](4,2)\{Line C\}[r]
\end{tikzpicture}

% \tzlink(+)
\begin{tikzpicture}
\tzhelplines(4,3)
\tzlink[->](0,0)\{A\}(1,0)
\tzlink+[bend right](0,2)\{curve\}[sloped](3,-2)\{End\}[r]
\tzlink[->,auto](1,2)\{-\}\\(B)\{4,1\}\{Goal\!\}\\[b,draw,blue]
\end{tikzpicture}

Shift You can move the line or curve by specifying the option <shift coor> before the first coordinate (or ever before the option "<path name>" if it exists).

% \tzlink(+): shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzlink+[bend right]0,2,3,-2\{A\}[r]
\tzlink+[bend right]<.5,5>(0,2)\{3,-2\}[A-shifted][r]
\tzlink[->](1,2)\{-\}\\(4,1)
\tzlink[->,dashed]<-.2,3>(1,2)\{-\}\\(4,1)
\end{tikzpicture}
**Naming paths: Intersections** You can name the path of `\tzlink` by specifying the quote option "<path name>" immediately before the first coordinate. You can use the path names of two paths to find an intersection point.

\begin{tikzpicture}
\helpelines(4,3)
\tzlink[blue]"BB"(0,3){line B}[sloped](3,0)
\tzlink[-,red,bend right]"CC"(1,0)(3,2){Line C}[r]
\txXpoint{BB}{CC}(E) \% intersection
\tdot*(E){E}[a]
\end{tikzpicture}

**Extending paths** You can extend the path of `\tzlink` by specifying TiKZ code in the last optional argument `<code.append>`. You can also use the macros `\tzlinkAtBegin` and `\tzlinkAtEnd` to extend the path at the beginning and at the end, respectively. Specifying the option `<code.append>` extends the path after `\tzlinkAtEnd`.

\begin{tikzpicture}
\helpelines(4,3)
\tzlinkAtBegin{(1,1) to[bend left]}
\tzlinkAtEnd{arc (180:90:1cm) node [r,draw] {BB}}
\tzlink[blue]"BB"(0,3){line B}[sloped](3,0)
\tzlink[-,red,bend right]"CC"(1,0)(3,2){Line C}[r]
\txXpoint{BB}{CC}(E) \% intersection
\tdot*(E){E}[a]
\end{tikzpicture}

### 13.6 `\tzlinks`: All in one: Semicolon versions

`\tzlinks` is a *semicolon version* of `\tzlink`. `\tzlinks` accepts any number of coordinates to connect them with connected line segments or curves. You can change how two adjacent points are connected. `\tzlinks` is quite flexible that you can think of it as a *generalized version* of `\tzlines`.

\begin{tikzpicture}
\helplines(4,3)
\tzlink{(coor)}{(coor)} ..repeated.. (coor) ;
\end{tikzpicture}

- `<code1>` must be link style, such as --, to, -|, etc. (possibly followed by other code)
- `<code2>` is especially for + or ++ (possibly with other code)
- repetition MUST be ended by ; (semicolon)
- full repeating pattern is (){}{}<>..repeated.. (){}{}<> {<fill opacity>} <code.append>

### Defaults: `\tzlinks`, `\tzlinks*`, `\tzlinks+`, `\tzlinks*+`
How to change link styles  There are three ways of changing link styles:

- The fist curly brace option \{<link style>\} controls the link style of connecting any two coordinates.
- The second bracket option [<code1>] (locally) changes the link style of two adjacent coordinates. [<code1>] (locally) overrides \{<link style>\}.
- The effect of the macro \settzlinkstyle remains until the end of \tikzpicture environment, unless changed again.

% standard version
\tzlinks(1,1)(2,2)(3,1)(4,3); % works like:
\draw (1,1) to (2,2) to (3,1) to (4,3);

The default path style is to, which can be by \settzlinkstyle. The effect remains valid until the end of \tikzpicture environment unless changed again.

% \tzlinks: \settzlinkstyle
\begin{tikzpicture}
\tzhelpelines(4,3)
\settzlinkstyle{to[bend right]} \%
\tzlinks[dashed](0,1)(1,2)[-|](3,1)(4,3);
\tzlinks[blue]
 (0,0)
 (1,1)
 (2,1)[to[out=-135,in=45]]
 (3,0);
\end{tikzpicture}

% \tzlinks: various link styles
\begin{tikzpicture}
\tzhelpelines(4,3)
\tzlinks[->](1,2)[-|]  
 (0,0)[..controls (1,2) and (2,0)..]  
 (3,1)[to[bend left]]<+++  
 (1,2);  
\end{tikzpicture}
Remark: In TikZ, edge is not part of the main path. So you need to be careful when you move the path or find intersection points.

13.6.2 \tzlinks+, \tzlinks*, \tzlinks++: Variants

The plus version \tzlinks+ treats a coordinate as relative (with ++ by default) to the previous coordinate, except the first coordinate.

\begin{tikzpicture}
\tzhelplines(4,3)
\tzlinks+(0,1)(1,2)[edge[dashed]]<+(3,1)>(3,1)(4,3);
\end{tikzpicture}
\begin{tikzpicture}
\tzhelplines(4,3)
\tzlinks+(0,1)(1,1)[to[bend right]](2,-1)(1,2);
\end{tikzpicture}

The starred version \tzlinks* is to fill the closed area formed by \tzlinks with color or pattern. The related default value is \[\text{fill=black!50, fill opacity=.3, text opacity=1}\]. In fact, \tzlinks*[draw=none] is (almost) the same as \tzpath* (See Section 14.2 on page 101 for more details).

\begin{tikzpicture}
\tzhelplines(4,3)
\tzlinks*(1,1)(2,2)(3,1)(4,3); \% works like:
\draw [fill=black!50,fill opacity=.3,text opacity=1]
(1,1) to (2,2) to (3,1) to (4,3);
\end{tikzpicture}
\begin{tikzpicture}
\tzhelplines(4,3)
\tzlinks++(1,1)(2,2){--}(3,1)(4,3);
\end{tikzpicture}
\begin{tikzpicture}
\tzhelplines(4,3)
\tzlinks+++(1,1)<++>
(1,1)[[rounded corners=10pt]--]
(0,3)
(0,0)[[sharp corners]--]
(4,0)<++>
(-1,2);
\end{tikzpicture}

\tzlinks*+ is the plus version of \tzlinks*.
You can also change the fill opacity by specifying the last curly brace optional argument \{<fill opacity>\}, after the semicolon (or before the option <code.append> it it exists).

You can also use the macros \settzfillcolor and \settzfillopacity to change the defaults. The effect remains valid until the end of the \tikzpicture environment, unless changed again.

### 13.6.3 Putting text, shift, intersections, and extending paths

**Adding text** You can add text next to connected line segment or around the last coordinate by specifying options \{<text>\}[<node option>].

\begin{tikzpicture}
\settzfillcolor{green}
\tzline[blue] (0,0) (3,1) (2,3) (0,2) ;
\tzlinks+ [dashed] "AA" (1,1){A} (2,2) [--] (B) [a]+ (3,1){C} [b] ; \% works like\draw [dashed, name path=AA]
    (1,1) to node {A}
    ++(2,2) -- node [above] {B}
    +(3,1) to node [below] {C} ;
\end{tikzpicture}
Remark: Note that edge is not part of the main path.

Shift You can move the path of \tzlinks with the option <shift coor>. An empty option <> is not allowed.

Naming paths: intersections You can name the path of \tzlinks immediately before the first coordinate and use it to find intersection points.

Extending paths You can extend the path of \tzlinks by writing TikZ code in the last (after the semicolon) optional argument <code.append>. So <--cycle> closes the path with a straight line.
You can also use \tztlinksAtBegin and \tztlinksAtEnd to extend the path of \tztlinks at the beginning and at the end, respectively. Specifying the option \code.append extends the path after \tztlinksAtEnd.

14 Filling Area

14.1 \tzpath: Semicolon version

The macro \tzpath is the same as \tztlinks[draw=none]. In other words, \tzpath creates a path connecting an arbitrary number of coordinates, but it does not stroke the path. Since \tzpath is a semicolon version macro, you need to enter a semicolon ‘;’ to indicate where the coordinate iteration ends.

You can visualize the path with \tzpath[draw], which is the same as \tztlinks.
Remark:

- The main purpose of \texttt{tzpath} is to fill an enclosed area with colors or patterns. You can use \texttt{tzpath[fill]} or \texttt{tzpath[pattern=<...>]} to do it.
- Use \texttt{settzpathlayer}, like \texttt{settzpathlayre{behind}}, to change the layer of \texttt{tzpath} (default: main).

\textbf{Path construction operation}  (If you are not an experienced user of TikZ just skip this part.)  \texttt{tzpath} allows you to choose how to construct a path using the \texttt{[<path style>]} option \texttt{in-between} coordinates. \textit{Path extension operation} can be selected from ‘--’, ‘to’, ‘|-’, ‘-|’, etc. You can use the \textit{first brace option} \texttt{[<path style>]} to change all the \texttt{[path style]} in-between coordinates.

\begin{tikzpicture}
\draw[help lines](4,3);
tzpath[draw](0,1)(1,2)[-|](3,1)(4,3);
tzpath[draw]
(0,0)[to[bend right]]
(1,1)
(2,1)[to[out=-135,in=45]]
(3,0);
\end{tikzpicture}

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

\begin{tikzpicture}
\settzpathstyle{to[bend right]}
tzpath[draw](0,1)(1,2)[-|](3,1)(4,3);
tzpath[draw]
(0,0)[to[bend right]]
(1,1)
(2,1)[to[out=-135,in=45]]
(3,0);
\end{tikzpicture}

\begin{tikzpicture}
\settzpathstyle{to[bend right]}
tzlinks(0,1)(1,2)[-|](3,1)(4,3);
tzlinks
(0,0)[to[bend right]]
(1,1)
(2,1)[to[out=-135,in=45]]
(3,0);
\end{tikzpicture}

You can extend the path of \texttt{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.
14.2 \texttt{\textbackslash path\textasciitilde}: Semicolon version

The starred version \texttt{\textbackslash path\textasciitilde} is the (almost) same as \texttt{\textbackslash links\textasciitilde[draw=none]}. It fills the interior of \texttt{\textbackslash path} with \texttt{fill=black\!50} with \texttt{fill opacity=.3} and \texttt{text opacity=1}, by default. The only difference between \texttt{\textbackslash path\textasciitilde} and \texttt{\textbackslash links\textasciitilde[draw=none]} is that the layer of \texttt{\textbackslash path\textasciitilde} can be changed by \texttt{\settzpathlayer}, like \texttt{\settzpathlayer(behind)}.

\texttt{\textbackslash path\textasciitilde} works like \texttt{\textbackslash path[fill=black\!50,fill opacity=.3,text opacity=1]}. You can change the defaults by \texttt{\settzfillcolor} and \texttt{\settzfillopacity}.

\begin{verbatim}
% syntax: minimal
\textbackslash path\textasciitilde\{<coor>\}<path> \{\texttt{(\textbackslash to)}\}\{\texttt{\textbackslash fill}} {\textbackslash fill\textbackslash op\textbackslash t\textbackslash ac\textbackslash i\textbackslash t\textbackslash y}}{\textbackslash t\textbackslash e\textbackslash x\textbackslash \textbackslash \textbackslash \textbackslash c\textbackslash o\textbackslash p\textbackslash a\textbackslash t\textbackslash i\textbackslash \textbackslash \textbackslash y}}\{\texttt{\textbackslash t\textbackslash e\textbackslash x\textbackslash \textbackslash \textbackslash \textbackslash c\textbackslash o\textbackslash p\textbackslash a\textbackslash t\textbackslash i\textbackslash \textbackslash \textbackslash y}}\}\texttt{\textbackslash ;}
\end{verbatim}

\begin{verbatim}
% syntax: full
\textbackslash path\textasciitilde\{<path style>\}[<opt>]<shift coor>="<path name>" <\textbackslash to> \{<code1>\}<\texttt{\textbackslash fill}} {\textbackslash fill\textbackslash op\textbackslash t\textbackslash ac\textbackslash i\textbackslash t\textbackslash y}}{\textbackslash t\textbackslash e\textbackslash x\textbackslash \textbackslash \textbackslash c\textbackslash o\textbackslash p\textbackslash a\textbackslash t\textbackslash i\textbackslash \textbackslash \textbackslash y}}\{\texttt{\textbackslash t\textbackslash e\textbackslash x\textbackslash \textbackslash \textbackslash c\textbackslash o\textbackslash p\textbackslash a\textbackslash t\textbackslash i\textbackslash \textbackslash \textbackslash y}}\}<\texttt{\textbackslash ;}
\end{verbatim}

\begin{verbatim}
% remark:
- \texttt{\textbackslash to} must be link style, such as \texttt{\textbackslash --}, \texttt{\textbackslash to}, \texttt{\textbackslash -|}, etc.
- \texttt{\textbackslash code2} is especially for \texttt{\textbackslash +} and \texttt{\textbackslash ++}
- repetition MUST be ended by ; (semicolon)
\end{verbatim}

\begin{verbatim}
% defaults: \texttt{\textbackslash path\textasciitilde}
*\textbackslash to}=\{\texttt{\textbackslash fill=black\!50,fill opacity=.3,text opacity=1}\}
\texttt{\textbackslash repeated}.. (\texttt{\textbackslash to})\{\texttt{\textbackslash fill}} {\textbackslash fill\textbackslash op\textbackslash t\textbackslash ac\textbackslash i\textbackslash t\textbackslash y}}{\textbackslash t\textbackslash e\textbackslash x\textbackslash \textbackslash \textbackslash c\textbackslash o\textbackslash p\textbackslash a\textbackslash t\textbackslash i\textbackslash \textbackslash \textbackslash y}}\{\texttt{\textbackslash t\textbackslash e\textbackslash x\textbackslash \textbackslash \textbackslash c\textbackslash o\textbackslash p\textbackslash a\textbackslash t\textbackslash i\textbackslash \textbackslash \textbackslash y}}\}
\end{verbatim}

\noindent\textbf{Filling the interior} You can optionally change the opacity of fill color using the \texttt{TikZ} option \texttt{fill opacity}.

\begin{verbatim}
% \texttt{\textbackslash path: fill, fill opacity}
\begin{verbatim}
\textbackslash begin\{tikzpicture\}
\textbackslash helplines\{4,3\}
\textbackslash line\{red,very thick\}\{0,3\}(4,0)
\textbackslash path\{\textbackslash fill\}\{0,0\}(3,1)(2,3)(0,2);
\texttt{\textbackslash repeat}..(\texttt{\textbackslash to})\{\texttt{\textbackslash fill}} {\textbackslash fill\textbackslash op\textbackslash t\textbackslash ac\textbackslash i\textbackslash t\textbackslash y}}{\textbackslash t\textbackslash e\textbackslash x\textbackslash \textbackslash \textbackslash c\textbackslash o\textbackslash p\textbackslash a\textbackslash t\textbackslash i\textbackslash \textbackslash \textbackslash y}}\{\texttt{\textbackslash t\textbackslash e\textbackslash x\textbackslash \textbackslash \textbackslash c\textbackslash o\textbackslash p\textbackslash a\textbackslash t\textbackslash i\textbackslash \textbackslash \textbackslash y}}\}
\end{verbatim}
\end{verbatim}

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

\begin{tikzpicture}
\helplines(4,3)
\path[preaction={pattern=bricks,preaction={fill=brown}}]
(0,0)(3,1)(-1,2)(-2,-1);
\path[pattern=north east lines,opacity=.5]<1,-1>(0,0)(3,1)(-1,2)(-2,-1);
\end{tikzpicture}

\settzfillcolor
\begin{tikzpicture}
\helplines(4,3)
\fill[red](0,0)(3,1)(2,3)(0,2);
\fill[green]<1,-1>(0,0)(3,1)(2,3)(0,2) {.5} 
\end{tikzpicture}

14.3 \texttt{\textbackslash path+} and \texttt{\textbackslash path++}: Relative coordinates: Semicolon versions

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

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

\texttt{\textbackslash path++} is simply the plus version of \texttt{\textbackslash path*}.

\settzfillcolor
\begin{tikzpicture}
\helplines(4,3)
\fill[red,very thick](0,3)(4,0)
\path[preaction={pattern=bricks,preaction={fill=brown}}]
(0,0)(3,1)(-1,2)(-2,-1);
\path[pattern=north east lines,opacity=.5]<1,-1>(0,0)(3,1)(-1,2)(-2,-1);
\end{tikzpicture}

You can change the layer (main by default) using \settzpathlayer.

\settzpathlayer
\begin{tikzpicture}
\helplines(4,3)
\fill[red,very thick](0,3)(4,0)
\path[preaction={pattern=bricks,preaction={fill=brown}}]
(0,0)(3,1)(-1,2)(-2,-1);
\path[pattern=north east lines,opacity=.5]<1,-1>(0,0)(3,1)(-1,2)(-2,-1);
\end{tikzpicture}
15 Curves

There are many ways to draw curves.

15.1 Bézier curves

15.1.1 \tzbezier

\tzbezier accepts three or four coordinates to draw a Bézier curve from the first coordinate to the last coordinate, with one or two control points.

Control points You can specify one or two control points, \(<\text{cntl-coor}>\).

% 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);

% \tzbezier
\begin{tikzpicture}
\tzhelplines(4,4)
\tzbezier[red](0,3)(1,2)(2,4)(4,3)
\tzbezier[blue](0,2)(1,.5)(4,0)
\end{tikzpicture}
The style `tzshowcontrols` displays the control points by drawing dotted lines, by default. You can also change the dotted line style, like `tzshowcontrols={dashed,green}`.

```latex
\begin{tikzpicture}
  \tzhelplines(4,4)
  \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 `{<text>}` and `[<node opt>]` immediately after the last coordinate.

```latex
\begin{tikzpicture}
  \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{tikzpicture}
```

**Shift** You can move the curve of `tzbezier` 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}
  \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{tikzpicture}
```

**Naming paths: intersections** You can name the path of `tzbezier` by specifying the option "<path name>" immediately before the first coordinate.
Extending paths  You can extend the path of \tzbezier from the last coordinate, by writing Ti\k Z code in the last optional argument <code.append>.

You can also use \tzbezierAtBegin and \tzbezierAtEnd to extend the path of \tzbezier at the beginning and at the end, respectively. Specifying the option <code.append> extends the path after \tzbezierAtEnd.

15.1.2 \tzbezier+: Relative coordinates

For the plus version \tzbezier+, the last coordinate is relative to the first coordinate. And the first control point is relative to the first coordinate.

In Ti\k Z, the second control point is relative to the last coordinate.

Therefore, for \tzbezier+(A)(B)(C)(D), (B) and (D) are relative to (A), and the second control point (C) is relative to the last coordinate (D).
15.2 Parabolas

\texttt{\texttt{\textbackslash tzparabola}} accepts \textit{two or three} coordinates to draw a parabola from the first coordinate to the last coordinate. In the case of three coordinates, the parabola bends at the second coordinate.
Parabolas \tzplot draws the graph of a quadratic function \( f(x) = ax^2 + bx + c \) for appropriate values of \( a, b, \) and \( c \).

```
% \tzparabola: two coordinates
\begin{tikzpicture}
\tzhelplines(4,3)
\tzparabola[thick](0,0)(2,2)
\tzparabola[bend at end,dashed](0,0)(2,2)
\tzparabola[red,thick,bend pos=.5](2,0)(4,3)
\tzparabola[blue,parabola height=-2cm](2,3)(4,2)
\end{tikzpicture}
```

```
% \tzparabola: three coordinates
\begin{tikzpicture}
\tzhelplines(4,3)
\tzparabola[thick](0,2)(1,0)(2,2)
\tzparabola[blue](0,3)(2,1)(4,2)
\tzparabola[red,thick](2,0)(3,2)(4,3)
\end{tikzpicture}
```

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.

```
% \tzparabola: adding text
\begin{tikzpicture}
\tzhelplines(4,3)
\tzparabola[thick](0,0)(2,2){parabola}[blue,draw,a]
\tzparabola[blue](0,3)(2,1)(4,2){AC}[r]
\tzparabola[red,thick,bend pos=.5](2,0)(4,3){C}[ar]
\end{tikzpicture}
```

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

```
% \tzparabola: shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzparabola[blue](0,3)(2,1)(4,2){AC}[r]
\tzparabola[dashed]<0,-.5>(0,3)(2,1)(4,2){AC}[r]
\tzparabola[dotted]<0,-1>(0,3)(2,1)(4,2){AC}[r]
\end{tikzpicture}
```

Naming paths: intersections 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 \tzparabola from the last coordinate by writing TikZ code in the last optional argument <code.append>.

You can also use the macros \tzparabolaAtBegin and \tzparabolaAtEnd to extend the path of \tzparabola at the beginning and at the end, respectively. Specifying the option <code.append> extends the path after \tzparabolaAtEnd.

15.2.2 \tzparabola+: Relative coordinates

The plus version \tzparabola+ uses the second and the third coordinates relative to the first coordinate.

Everything else is the same as in \tzparabola.

% two coordinates
\tzparabola+(0,1)(1,2) % works like:
\draw (0,1) parabola \((0,1)+(1,2)\);

% three coordinates
\tzparabola+(0,1)(1,1)(3,-1) % works like:
\draw (0,1) parabola bend +(1,1) \((0,1)+(3,-1)\);
15.3 Edges

15.3.1 \tzedge(+)

The macro \tzedge connects two coordinates with a straight or curved line, using the Ti\textit{K}Z edge operation.

\begin{verbatim}
\begin{tikzpicture}
  \tzhelplines(4,3)
  \tzparabola+[thick](0,0)(2,2){parabola}[blue,draw,a]
  \tzparabola+[blue](0,3)(2,-2)(4,-1){AC}[r]
  \tzparabola+[red,thick,bend pos=.5](2,0)(2,3){C}[r]
\end{tikzpicture}
\end{verbatim}

Remark: In Ti\textit{K}Z, the edge operation works like the to operation, but it is independently drawn after a main path is drawn and does not form a main path. So the option "<path name>" (for finding intersections) is not provided in \tzedge.

\begin{verbatim}
\tzedge(1,1)(3,2)  \% works like:
\draw (1,1) edge (3,2);
\end{verbatim}

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

The plus version \tzedge+ use the second coordinate relative to the first coordinate.

\begin{verbatim}
\tzedge+(1,1)(3,2)  \% works like:
\draw (1,1) edge +(3,2);
\end{verbatim}

\begin{verbatim}
\tzedge[->,bend right](1,1){A}[near start](3,2){B}[right]  \% works like:
\draw (1,1) edge [->,bend right] node [near start] {A} ++(3,2)
\end{verbatim}
The macro \texttt{\tzedges} accepts any number of coordinates to draw edges from the first coordinate to each of the next using the TikZ \texttt{edge} operation. That is, the first coordinate is the unique start coordinate, and all others are target coordinates. \texttt{\tzedges} is a semicolon version, so you need to type \texttt{;} to indicate when the coordinate iteration ends.

Remark: In TikZ, the \texttt{edge} operation works like \texttt{to} operation, but it is added after main path is formed, like \texttt{node} does.

- Each \texttt{edge} is drawn independently from a main path as well as any other \texttt{edge}'s. (See TikZ manual for more details.)
- The \texttt{edge} operation of TikZ does not change change anything about a main path, so the current point is not changed by \texttt{\tzedges}. This means that the last node (with \texttt{\{<text>\}} and \texttt{\{<node opt>\}}) works with the first (namely, start) coordinate.
The plus version \texttt{\texttt{tzedges+}} uses the second and next coordinates relative to the first coordinate.

\texttt{\texttt{\tzedges}(0,1)(2,1)(3,0)(4,2);} % works like
\begin{verbatim}
\draw (0,1) edge ++ (2,1) edge ++ (3,0) edge ++ (4,2);
\end{verbatim}

\textbf{Remark:} The option \texttt{code.append} works with the first coordinate because \texttt{edge} and \texttt{node} do not change the current point.
15.4 More curves

15.4.1 `\tzplotcurve`, `\tzplot`

You can draw curves with `\tzplotcurve` and `\tzplot`.

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

See Section 10.6 on page 64, for more details on `\tzplotcurve`. See Section 10.1 on page 58, for more details on `\tzplot`.

15.4.2 `\tzto`, `\tztos`

You can draw curves with `\tzto` and `\tztos`. `\tztos` are quite useful to draw various curves.

\begin{tikzpicture}
\coors*(0,0)(A)(1,2)(B){B}[b](3,3)(C)(4,2)(D);
\tos(A)[out=80,in=180]
(B)[out=0,in=180]
(C)[out=0,in=180]
(D){End!}[r];
\end{tikzpicture}

See Section 13.1 on page 86, for more details on `\tzto`. See Section 13.3 on page 88, for more details on `\tztos`.

15.4.3 `\tzlink`, `\tzlinks`

You can also draw curves with `\tzlink` and `\tzlinks`.
15.4.4 \tzfn

With \tzfn, you can plot functions such as $f(x) = \frac{1}{3}(x - 1)^3 + 1$, $g(x) = \sin x$, $h(x) = \sqrt{x - 1}$, and so on. See Section 21.1 on page 152, for more details on \tzfn.
16 Polygons and Circles

16.1 Polygons: \tzpolygon: Semicolon versions

16.1.1 \tzpolygon

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

% syntax: minimum
\tzpolygon(<coor>)(<coor>)..repeated..(<coor>) ;
% syntax: medium
\tzpolygon (coor)(<text>)[<node opt>]..repeated..(<coor>{<text>}[<node opt>] ;
% syntax: full
\tzpolygon<opt><shift coor>"<path name>" (coor){<text>}[<node opt>] ..repeated.. (){}[] ; <code.append>
% defaults
{}<>"{}[] ..repeated.. (){}[] ; <>

\tzpolygon(1,1)(2,2)(3,1)(4,3); % works like:
\draw (1,1) -- (2,2) -- (3,1) -- (4,3) -- cycle;

You can add text next to lines by specifying the options \{<text>\} and \[<node opt>\] in-between coordinates.

% \tzpolygon
\begin{tikzpicture}
\tzhelplines(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 \texttt{shift coor} before the first coordinate. The \texttt{empty} shift option <> is \texttt{not allowed}.

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

The starred version \texttt{\tzpolygon*} paints the interior of the polygon with the default options \texttt{fill=black!50} with \texttt{fill opacity=.3} and \texttt{text opacity=1}.

\begin{verbatim}
% syntax: minimum
\tzpolygon*{(coor)}{(coor)}..repeated..{(coor)};
% syntax: medium
\tzpolygon*{(coor)}{(text)}{(node opt)}..repeated..{(coor)}{(text)}{(node opt)};
% syntax: full
\tzpolygon*[<opt>]<shift coor>"<path name>"
   {(coor)}{(text)}{(node opt)}..repeated..(){[]} {<fill opacity>} <code.append>
% defaults
*[{fill=black!50,fill opacity=.3,text opacity=1}]<""{(m)}{}[] .. repeated.. (){}[] ; {.3} <>
\end{verbatim}

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

\begin{verbatim}
% \texttt{\tzpolygon: shift}
\begin{tikzpicture}
\tzhelplines(4,3)
\tzpolygon*[draw=blue,auto]
   (0,1){Side A}[sloped,red]
   (1,3){B}
   (2,3)
   (3,2){Side D}[swap,sloped]
   (1,0);
\tzpolygon*[green,auto,dashed,text=black]<1,-.5>
   (0,1)(2,2)\{C\}(3,2)(1,0); {.7}
\end{tikzpicture}
\end{verbatim}

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

16.1.3 \texttt{\tzpolygon+}, \texttt{\tzpolygon++}: Relative coordinates: Semicolon versions

The plus version \texttt{\tzpolygon+} uses each coordinate (except the first one) relative (with ++) to the previous coordinate. \textit{Everything else is the same as in \texttt{\tzpolygon}.} \texttt{\tzpolygon++} is just a plus version of \texttt{\tzpolygon*}.

\begin{verbatim}
\tzpolygon+(1,1)(2,2)(3,1)(4,3); % works like:
\draw (1,1) -- ++(2,2) -- ++(3,1) -- ++(4,3) -- cycle;
\end{verbatim}

\begin{verbatim}
\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;
\end{verbatim}
16.2 Rectangles

16.2.1 \tzframe and its variants

\tzframe accepts two coordinates draws a rectangle.

\% syntax: minimum
\tzframe\(<\text{coor}>\)(<\text{coor}>)
\% syntax: full
\tzframe[<opt>]<\text{shift coor}>"<\text{path name}>"(<\text{coor1}>)(<\text{coor2}>)<\text{code.append}>
\% defaults
[]<>""(<\text{m}>)(<\text{m}>)<

\tzrectangle and \tzbox are aliases of \tzframe.

\tzframe\((0,1)\)(3,2) \% works like:
\begin{tikzpicture}
\draw (0,1) rectangle (3,2);
\end{tikzpicture}

The plus version \tzframe+ uses the second coordinate as the coordinate relative (with ++) to the first. \tzrectangle+ and \tzbox+ are aliases of \tzframe+.

\tzframe+(0,1)(3,2) \% works like:
\begin{tikzpicture}
\draw (0,1) rectangle ++(3,2);
\end{tikzpicture}

The starred version \tzframe* fills the interior with black!50 with fill opacity=.3 and text opacity=1, by default. (\tzrectangle* and \tzbox* are aliases of \tzframe*.) \tzframe+ has also its starred version \tzframe++. (\tzrectangle++ and \tzbox++ are aliases of \tzframe++.)

\% syntax
\tzframe*[<opt>]<\text{shift coor}>"<\text{path name}>"
\(<\text{coor1}>\)(<\text{coor2}>){<\text{fill opacity}>}<\text{code.append}>

116
With the starred versions, you can change the fill opacity using the last option {<fill opacity>}. You can move \texttt{\texttt{tzframe}} and its variants by specifying the option <\texttt{shift coor}> immediately before the first mandatory coordinate. The empty shift option <> is not allowed.

You can use the last option <\texttt{code.append}> to add more Ti\text{KZ} code.

16.2.2 \texttt{\texttt{tzrectangling}}(*)

\texttt{\texttt{tzrectangling}}* draws two rectangles and draws a rectangle ring by filling the interior with the default options even odd rule, fill=black!50, fill opacity=.3, and text opacity=1.
\trectangling*(A1)(A2)(B1)(B2) \% works like:
\begin{tikzpicture}
\zhelplines(4,2)
\trectangling*[blue](0,0)(3,2)(.5,.5)(2.5,1.5)
\end{tikzpicture}
\trectangling draws two rectangles with the default option \emph{even odd rule}.

% syntax: minimal
\trectangling (coorA1)(coorA2)(coorB1)(coorB2)
% syntax: full
\trectangling [opt]<shift coor>(coorA1)(coorA2)
\hspace{1cm}(coorB1)(coorB2)<code.append>
% defaults
[even odd rule]<>(<m>)(<m>)(<m>)(<m>)
\trectangling(0,0)(2,2)(.5,.5)(1.5,1.5) \% works like:
\begin{tikzpicture}
\draw (0,0) rectangle (2,2)
(0,0) rectangle (2,2);
\end{tikzpicture}
\begin{tikzpicture}
\zhelplines(4,3)
\trectangling[blue](0,0)(3,2)(.5,.5)(2.5,1.5)
\draw[red,fill=green]<.9,.9>(0,0)(3,2)(.5,.5)(2.5,1.5)
\end{tikzpicture}
% syntax: shift
\begin{tikzpicture}
\zhelplines(4,3)
\trectangling[blue](0,0)(3,2)(.5,.5)(2.5,1.5)
\draw[red,fill=green]<.9,.9>(0,0)(3,2)(.5,.5)(2.5,1.5)
\end{tikzpicture}
\begin{tikzpicture}
\zhelplines(4,3)
\trectangling*[blue](0,0)(3,2)(1.5,1.5)(4,3)
\end{tikzpicture}
With the last option \texttt{<code.append>} you can add some more \LaTeX{} code.

\begin{tikzpicture}
\helplines
\cntaglering*[red,draw=none,nonzero rule]
\begin{scope}
\path[fill=white]
(0,0)(3,2)(1.5,1.5)(4,3);
\end{scope}
\end{tikzpicture}

\cntaglering and \cntaglering* are aliases of \cntaglering.*

\cntaglering* and \cntaglering* are aliases of \cntaglering.*

16.3 Circles and rings

16.3.1 \cntaglering(*)

\cntaglering draws a circle around a specified coordinate with a specified radius. The coordinate and the radius are mandatory.

\begin{tikzpicture}
\helplines
\cntaglering(1,1)(1cm)
\cntaglering[blue,dashed]"AA"(2,2)(1cm)
\dots*(A-1){$A_1$}[r](A-2){$A_2$}[r];
\end{tikzpicture}

A

The starred version \cntaglering* fills the interior with \texttt{fill=black!50} with \texttt{fill opacity=.3} and \texttt{text opacity=1}, by default. You can change the fill opacity using the curly brace option \texttt{\{<fill opacity>\}} right after the option \texttt{(<radius>)}.
% syntax
\tzcircle*[<opt>]<shift coor>"<path name>"
  (coor)(radius){fill opacity}<code.append>
% defaults
*[fill=black!50,fill opacity=.3,text opacity=1]<>"<(m)>(m){.3}<>}

You can move the circles by specifying the option <shift coor> before the center coordinate or immediately before the option "<path name>" if it exists. The empty shift option <> is not allowed.

With the last option <code.append>, you can add some Ti\kZ code.

16.3.2 \tzring(*)
\tzring* draws two circles and draws a circle ring by filling the interior with the default options even odd rule, fill=black!50, fill opacity=.3, and text opacity=1.
\tzzring* fills opacity
\begin{tikzpicture}
\tzhelplines(4,4)
\tzzring*[green](2,2)(1)(2,2)(1.5)
\tzzring*[fill=blue](3,1)(1)(3,1)(.7){1}
\end{tikzpicture}

% \tzzring*: nonzero rule
\begin{tikzpicture}
\tzhelplines(4,2)
\tzzring*[fill=red, nonzero rule](1,1)(1)(2,1)(1)
\end{tikzpicture}

% \tzzring*: nonzero rule
\begin{tikzpicture}
\tzhelplines(4,2)
\tzzring*[fill=red, nonzero rule](1,1)(1)(2,1)(1){1}
\tzzring*[fill=white](1,1)(1)(2,1)(1){1}
\end{tikzpicture}

% \tzzring* draws two circles with the default option even odd rule.
\begin{tikzpicture}
\tzframe(0,0)(4,3)
\tzzring*[blue](1.5,1.5)(1)(2.5,1.5)(1)
\end{tikzpicture}

% syntax: minimal
\tzzring*\{<coor>\}(<radius>\}(<coor>\}(<radius>\}
% syntax: full
\tzzring*\{<opt>\}<shift coor>
\{<coor>\}(<radius>\}(<coor>\}(<radius>\)<code.append>
% defaults:
\{even odd rule\}\{<m>\}(<m>\){<m>\})}

% \tzzring: shift
\begin{tikzpicture}
\tzhelplines(4,4)
\tzcoors(2,2)(A)(3,1)(B);
\tzzring*[blue](2,2)(1)(2,2)(1.5)
\tzzring*[fill=green]<-1,0>(3,1)(1)(3,1)(.7)\% shift
\end{tikzpicture}

You can add some TikZ code with the last option <code.append>.
16.4 Ellipses

16.4.1 \tzellipse(*)

\tzellipse draws an ellipse around a specified coordinate with the specified x-radius and y-radius. The starred version \tzellipse* fills the interior with fill=black!50 with fill opacity=.3 and text opacity=1, by default. \tzellipse(*) is basically the same as \tzcircle(*).

\begin{tikzpicture}
\zhelplines(4,3)
\tzellipse(0,0)(1 and .5) \% works like:
\draw (0,0) ellipse (1 and .5);
\end{tikzpicture}

\% \tzellipse(*)
\begin{tikzpicture}
\zhelplines(4,3)
\tzellipse(2,2)(1.5 and 1)
\tzellipse*[blue](2,1)(1 and 1.5){.5} \% fill opacity
\tzellipse*[fill=green](3,1){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. Using the last option \{<fill opacity>\}, you can change the fill opacity.

\begin{tikzpicture}
\zhelplines(4,3)
\tzellipse*[blue](2,2)(1.5 and 1)
\tzellipse*[blue](2,1)(1 and 1.5){.5} \% fill opacity
\end{tikzpicture}

You can add some TikZ code with the option \texttt{code.append}. 122
\tzoval is an alias of \tzellipse and \tzoval* is an alias of \tzellipse*.

16.4.2 \tzellipsering(*)

\tzellipsering* draws two ellipses and draws an ellipse ring by filling the interior with the default options \texttt{even odd rule}, \texttt{fill=black!50}, \texttt{fill opacity=.3}, and \texttt{text opacity=1}.
\tzellipsering draws two ellipses with the default option \texttt{even odd rule}.
\tzellipsering(*) is basically the same as \tzring(*).
\begin{tikzpicture}
  \helplines(4,4)
  \coors(A)(B);
  \ellipsering*[blue](1 and 1.5)
  (1.5,2.5) circle (3mm) >
  \ellipsering[pattern=bricks](1 and 1.5)
  (2.5,.5) rectangle ++(1,1) >
\end{tikzpicture}

\begin{tikzpicture}
  \helplines(4,4)
  \coors(A)(B);
  \ovalring*[fill=red,draw=none,nonzero rule]
  (1 and 1.5)(1 and 1.5)
  \ovalring[fill=white]
  (1 and 1.5)(1 and 1.5)
\end{tikzpicture}

\ovalring is an alias of \ellipsering and \ovalring* is an alias \ellipsering*.

17 Arcs, Wedges, and Angle Marks

17.1 \arc(): Centered arcs

17.1.1 Arcs

\arc draws an arc around a specified center coordinate.

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

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

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

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

You can move arcs by specifying the option \texttt{<shift coor>} before the center coordinate or immediately before the option "\texttt{<path name>}" if it exists. The empty shift option \texttt{<>} is not allowed.

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

17.1.2 Elliptical arcs

\texttt{\tzarc} draws an elliptical arc if you specify \texttt{x-radius} and \texttt{y-radius}.
17.2 `{\tzarcfrom}`: Arcs as in Ti\kZ

`{\tzarcfrom}` draws an arc starting from a specified point, like Ti\kZ does.

\begin{tikzpicture}
\helplines[4,4]
\tzarc[->,red](1,1)(30:270:1.5 and 0.5)
\tzarc'[-,dotted](1,1)(30:270:1.5 and 0.5)
\tdots*(1,1)(2,2);
\tzarc[-,blue,dashed](2,2)(0:-270:1 and 2)
\tzarc'(2,2)(0:-270:1 and 2)
\end{tikzpicture}

\begin{tikzpicture}
\helplines[4,4]
\tzarc[->,red](1,1)(30:270:1.5 and 0.5)
\tdots*(1,1)(2,2);
\tzarc[blue,fill=green,fill opacity=.3](2,2)(0:-270:1 and 2)<--(2,2)--cycle  \% code.append
\tdots*(2,2)(0:-270:1 and 2) \% shift
\end{tikzpicture}

\begin{tikzpicture}
\helplines[4,4]
\tzarcfrom[blue,->] (1,1)(-45:180:1){A}[r]
\tzarcfrom'[dashed,->](1,1)(-45:180:1)
\tzarcfrom'[->](3,2)(-45:180:1){arc}[midway,sloped]
\tzarcfrom'[dashed,->](3,2)(-45:180:1)
\end{tikzpicture}

Everything else is the same as in `{\tzarc}`.  
`{\tzarcfrom}' is the swap version of `{\tzarcfrom}`.
17.3 \texttt{\textbackslash{tzarcsfrom}: Connected arcs: Semicolon version}

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

% syntax: minimum
\texttt{\textbackslash{tzarcsfrom}}(<start coor>)
\hspace{1em}(<angA:angB:radius>).repeated..(<angA:angB:radius>) ;

% syntax: full
\texttt{\textbackslash{tzarcsfrom}}[<opt>]<shift coor>"<path name>"
\hspace{1em}(<start coor>)(<angA:angB:radius>{<text>}[<node opt]}
\hspace{1em}..repeated..();<code.append>

% defaults
[]<>""(<m>) (<m>){}..repeated..(){} ; <code.append>

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

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

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

17.4.1 \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. \tzwedge does not have the option \texttt{\textit{code.append}}.

% syntax
\tzwedge\[<\text{opt}>>\text{shift coor}><\text{path name}>
<\text{coor}>({\text{ang}\text{A}:\text{ang}\text{B}:\text{radius}})<\text{text}>[\text{node opt}]\]
% defaults
[]\"{}\[\text{midway}]\]

\tzwedge(1,1)(30:120:1) \% works like:
\draw (1,1) -- ++(30:1) arc (30:120:1) -- cycle;

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

\tzwedge'(1,1)(30:120:1) \% works like:
\draw (1,1) -- ++(30:1) arc (30:120-360:1) -- cycle;
17.4.2 \tzwedge\texttt{'}

The starred version \tzwedge\texttt{*} fills the wedges with \texttt{fill=black!50} with \texttt{fill opacity=.3} and \texttt{text opacity=1}, by default. With \texttt{\settzfillcolor} and \texttt{\settzfillopacity}, you can change the default values. You can also change the fill opacity by specifying the last optional argument \{\texttt{fill opacity}\}.

\begin{tikzpicture}[->,>=stealth]
\tzhelplines
\tzwedge[very thick,blue](1,1)(30:120:1.5)
\tzwedge*[->,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{tikzpicture}
17.5 Angle marks

17.5.1 \tzpointangle: Angles between points

\tzpointangle(<coor1>)(<coor2>)(<\mymacro>) computes the angle between two points and allows you to use, where (coor1) serves as the coordinate of the center.
17.5.2 \texttt{tzanglemark}(':') Angle marks

\texttt{tzanglemark} accepts three mandatory coordinates to display an angle mark by an arc (of radius 10pt, by default) for the second coordinate, on the \texttt{behind} layer by default. You can change the angle arc radius by \texttt{\settzAAradius}. You can change the layer by \texttt{\settzanglelayer}. Its alias is \texttt{\settzanglemarklayer}.

The default line width of angle marks is very thin. You can change the default line width with \texttt{\settzAAlinestyle}.

You can add angle text by the options \{<text>\} and \[<node opt>\].

\begin{tikzpicture}
\tzcoors*(4,2)(A){A}(1,1)(B){B}[180](2,3)(C){C};
\tzlines(A)(B)(C);
\tzlines[thick,blue](A){(2.5,0)}(C);
\tzanglemark(A){(B)}{C}{\textbackslash theta}{\% angle mark}
\end{tikzpicture}

\begin{tikzpicture}
\tzcoors(4,2)(A){A}(1,1)(B){B}[180](2,3)(C){C};
\tzlines(A){(B)}{C};
\tzlines[\settzAAradius=20pt](A){(2.5,0)}{\%}
\tzanglemark(C){(2.5,0)}{A}{\textbackslash alpha}{\%}
\end{tikzpicture}

\textbf{How it works}  Every \texttt{tzanglemark} calculates angles (from 0° to 360°) and stores the values under the names \texttt{\tzangleONE} and \texttt{\tzangleTWO}. The difference of the two numbers is stored as an absolute value under the name \texttt{\tzangleresult}. Of course, you can use these values \textit{only after} running \texttt{tzanglemark}.

- \texttt{tzanglemark} draws an angle mark between two angles, counterclockwise, from small to large.
- \texttt{tzanglemark'} draws an angle mark between two angles, clockwise, from small to large.
Remark: Simple to use:

- \texttt{\textbackslash \tzmarkangle(A)(B)(C)} draws an angle mark by an arc from (A) to (C) about (B).
- \texttt{\textbackslash \tzmarkangle(C)(B)(A)} draws an angle mark by an arc from (C) to (A) about (B).
- Ignoring the direction, \texttt{\tzanglemark(A)(B)(C)} and \texttt{\tzanglemark(C)(B)(A)} give the same result.

Swap version The \textit{swap} version \texttt{\tzanglemark'} draws an angle mark for an angle in $360^\circ - \theta$. In other words, \texttt{\tzanglemark'} \textit{switches the direction} of drawing an angle arc from counterclockwise to clockwise, and vice versa.

Angle mark text position The midpoint of an angle arc is stored under the coordinate name \texttt{tzAAmid}. The angle mark text is put on the line that goes through the middle point and (tzAAmid). The default (\texttt{<arc radius>}) is 10pt and the default position of angle text is \texttt{pos=1.5} in \texttt{[<node opt>]}.
\begin{tikzpicture}
\helplines(4,3)
\coors*(4,2)(A){A}(1,1)(B){B}[180](2,3)(C){C} ;
\lines(A)(B)(C);
\anglemark(A)(B)(C){$\theta$}[pos=.65](20pt) \%
\dot*(tzAomid)
\line[red,dashed,tzextend={1cm}{2cm}](B)(tzAomid)
\end{tikzpicture}

\begin{tikzpicture}
\helplines(4,3)
\coors*(4,2)(A){A}(1,1)(B){B}[180](2,3)(C){C} ;
\lines(A)(B)(C);
\anglemark*[red](C)(B)(A){$\theta$}
\end{tikzpicture}

\begin{tikzpicture}
\helplines(4,3)
\coors*(4,3)(A){A}(1,2)(B)(3,0)(C){C} [0]
\lines(A)(B)(C);
\anglemark*[red](A)(B)(C){$\theta$}
\anglemark*[blue](A)(B)(C){$\theta'$}(15pt)
\end{tikzpicture}

\begin{tikzpicture}
\helplines(4,3)
\coors*(4,3)(A){A}(1,2)(B)(3,0)(C){C} [0]
\lines(A)(B)(C);
\anglemark*[red](A)(B)(C){$\theta$}
\anglemark*[blue](A)(B)(C){$\theta'$}[15pt] % swap
\end{tikzpicture}

Remark: Instead of using the options \{<text>\} and \[<node opt>\], you can also use the coordinate (tzAomid) to place the angle text wherever you want, without using \anglemark\('. Of course, you can use the correct (tzAomid) only after running \anglemark\.

17.5.3 \anglemark\*\(': Fill angle marks
\anglemark\* fills (in the behind layer, by default) the angle mark area with fill=black!50 and with the options fill opacity=.3 and text opacity=1 by default. It does not draw any lines: [draw=none] by default.

Using the macros such as \settzfillcolor, \settzfillopacity, and \settzanglelayer, you can change the default values.

\begin{tikzpicture}
\helplines(4,3)
\coors*(4,2)(A){A}(1,1)(B){B}[180](2,3)(C){C} ;
\lines(A)(B)(C);
\anglemark*{red}[draw=black!50,fill opacity=.3])\anglemark*[red,draw=none]{\theta}
\end{tikzpicture}

\begin{tikzpicture}
\helplines(4,3)
\coors*(4,3)(A){A}(1,2)(B)(3,0)(C){C} [0]
\lines(A)(B)(C);
\anglemark*[red](A)(B)(C){$\theta$}
\anglemark*[blue](A)(B)(C){$\theta'$}
\end{tikzpicture}

\begin{tikzpicture}
\helplines(4,3)
\coors*(4,3)(A){A}(1,2)(B)(3,0)(C){C} [0]
\lines(A)(B)(C);
\anglemark*[red](A)(B)(C){$\theta$}
\anglemark*[blue](A)(B)(C){$\theta'$}[15pt] % swap
\end{tikzpicture}

\begin{tikzpicture}
\helplines(4,3)
\coors*(4,3)(A){A}(1,2)(B)(3,0)(C){C} [0]
\lines(A)(B)(C);
\anglemark*[red](A)(B)(C){$\theta$}
\anglemark*[blue](A)(B)(C){$\theta'$}[15pt] % swap
\end{tikzpicture}
17.5.4 \texttt{\textbackslash tzrightanglemark}: Right angle marks

\texttt{\textbackslash tzrightanglemark} takes three coordinates as mandatory arguments to display a right angle mark for the second coordinate. The mark is drawn on the behind layer by default, which can be changed by \texttt{\textbackslash settzanglelayer}.

The default line width is very thin, which can be changed by the option \texttt{[<opt>]}. You can also change the line width using \texttt{\textbackslash settzRAlinestyle}, which is valid until the end of \texttt{tikzpicture} environment. \texttt{\textbackslash settzRAlinestyle} is an alias of \texttt{\textbackslash settzAAlinestyle}. The length of the side is 5pt by default, and it can be changed by the last option \texttt{(<size>)}. You can also change the size with \texttt{\textbackslash settzRAsize}, which is valid until the end of the \texttt{tikzpicture} environment.

Remark:

- \texttt{\textbackslash tzrightanglemark(A)(B)(C)} and \texttt{\textbackslash tzrightanglemark(C)(B)(A)} give the same result.
- \texttt{\textbackslash tzrightanglemark'} is redundant, but it is provided to avoid frequent coding errors.

Each \texttt{\textbackslash tzrightanglemark} defines (tzRAvertex) as the coordinate of the right angle mark vertex. The angle text is placed on the line going through the second coordinate and (tzRAvertex). The default position is \texttt{pos=2} in \texttt{[<node opt>]}.
Remark: You can also use the coordinate (tzRAvertex) to place angle text wherever you want, after \t祚rightanglemark.

17.5.5 \t祚rightanglemark*: Fill right angle marks

The starred version \t祚rightanglemark* fills the interior of right angle marks with fill=black!50, with fill opacity=.3 and text opacity=1. It does not draw any line: [draw=none] by default. The filled mark is drawn on the behind layer by default, which can be changed by \settzanglelayer. Its alias is \settzanglemarklayer.

With \settzfillcolor and \settzfillopacity, you can also change the default fill color and fill opacity.
17.6 \tzdistance: Distances and changes

\tzdistance(<coor1>)(<coor2>){<\mylength>} calculates the Cartesian distance from (<coor1>) to (<coor2>) and stores the (absolute) value to the user-specified macro \mylength in centimeters. And \tzdistance also stores the changes in x and y to {<\Delta x>} and {<\Delta y>}, respectively. Note that the calculated distance is approximate.
\begin{tikzpicture}
\helplines(4,3)
\coors*(1,2)(A){A}[180](3,1)(B){B}[0];
\distance(A)(B){\lenA}{\xdist}{\ydist} % A to B
\pointangle(A)(B){\angA}
\line[red](A)(B)
\arc[->,blue](A)(\angA-180:90:\lenA)
\line+[\rightarrow]<0,-.5>(A|-B)
\node at (\xdist,0) {$\Delta x = \xdist$};
\line+[\rightarrow]<-1,0>(A)(0,\ydist)
\end{tikzpicture}

% \Delta x: (+), \Delta y: (-)
\begin{tikzpicture}
\helplines(4,3)
\coors*(1,2)(A){A}[180](3,1)(B){B}[0];
\distance(B)(A){\lenA}{\xdist}{\ydist} % B to A
\pointangle(B)(A){\angA}
\line[red](B)(A)
\arc[-,blue](A)(\angA:90:\lenA)
\line+[\rightarrow]<0,-.5>(A|-B)
\node at (\xdist,0) {$\Delta x = \xdist$};
\line+[\rightarrow]<-1,0>(B-|A)(0,\ydist)
\end{tikzpicture}
Part IV
Plotting Graphs

18 Axes

18.1 Draw axes

18.1.1 \tzaxes

Basically, \tzaxes(<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.

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

\begin{tikzpicture}
\helplines(4,3)
\tzaxes(4,3){$x$}{$y$}
\end{tikzpicture}

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

\begin{tikzpicture}[scale=.7]
\helplines(4,3)
\tzaxes(4,3){$x$}{$y$}
\end{tikzpicture}

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

\begin{tikzpicture}[scale=.7]
\helplines(4,3)
\tzaxes<1,5>(4,3){$x$}{$y$}
\end{tikzpicture}
name path  With the option "<path name>" , you can name the path of \texttt{\textbackslash tzaxes} (by default, \texttt{axes}). This makes it easy to find \textit{x-intercepts} and \textit{y-intercepts} of a graph.

The x-intercepts are found first and then the y-intercepts are found, as in the following example.

\begin{tikzpicture}[scale=.7]
\begin{scope}
\begin{scope}
\begin{scope}
\end{scope}
\end{scope}
\end{scope}
\end{tikzpicture}
\begin{tikzpicture}[scale=.5]
\end{tikzpicture}

18.1.2 \texttt{\textbackslash tzaxes}\texttt{*}

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

\tzaxisx draws only the $x$ axis.

\begin{verbatim}
% syntax
\tzaxisx[<opt>]<y-shift>"<path name>"{<from>}{<to>}
{<x-axis label>}[<node opt>]
% defaults
[-r]<0>"axisx"{<m>}{<m>}{r}
% arguments:
[1]: line style, arrow type (for $x$-axis)
[2]: $y$-shift of $x$-axis
[4]: $x$-axis starts from $m$
[5]: $x$-axis runs to $m$
[6]: $x$-axis label
[7]: $x$-axis label option
\end{verbatim}

With the option "<path name>", you can name each path of \tzaxisx and \tzaxisy (by default, axisx and axisy, respectively).

\tzaxisy draws only the $y$ axis.

\begin{verbatim}
% syntax
\tzaxisy[<opt>]<x-shift>"<path name>"{<from>}{<to>}
{<y-axis label>}[<node opt>]
% defaults
[-r]<0>"axisy"{<m>}{<m>}{r}
% arguments:
[1]: line style, arrow type (for $y$-axis)
[2]: $x$-shift of $y$ axis
[4]: $y$-axis starts from $m$
[5]: $y$-axis runs to $m$
[6]: $y$-axis label
[7]: $y$-axis label option
\end{verbatim}

With the option "<path name>", you can name each path of \tzaxisx and \tzaxisy (by default, axisx and axisy, respectively).
18.3 Display the origin

18.3.1 \tzshoworigin

\tzshoworigin prints ‘0’ (approximately) at the bottom left of the origin (0,0), by default.

% syntax
\tzshoworigin<shift coor><origin>{<text>}{<node opt>}
% default
<>(0,0){0}[below left,text height=1.25ex,text depth=.25ex]

All arguments of \tzshoworigin are optional.

% \tzshoworigin
\begin{tikzpicture}[scale=.45]
\zhelplines(8,5)
\tzshoworigin
\tzaxes(8,5)
\end{tikzpicture}

You can change the text by specifying the curly brace option {<text>}, like, for example, \tzshoworigin{$O$}. You can also change the coordinate of origin by the option (<origin>). Specifying the option <shift coor> also moves the origin.
18.3.2 \texttt{\textbackslash tzshoworigin*}

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

% syntax
\texttt{\textbackslash tzshoworigin*}[<dot opt>]<shift coor><(origin)>{<text>}[<node opt>]<(dot size)>
% default
*()[(0,0){[below left,text height=1.25ex,text depth=.25ex]}(2.4pt)]

You can add text with the option \texttt{<text>}. The default size of the dot is 2.4pt, and it can be changed with the last option \texttt{<dot size>}. You can change the dot style using the first optional argument \texttt{<dot opt>}. You can also move the dot by specifying the option \texttt{<shift coor>}.

Remark: For \texttt{\textbackslash tzshoworigin*}, 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 \texttt{label} for the node dot. This also means that the origin text cannot be positioned by an \texttt{<angle>}.)

% \texttt{\textbackslash tzshoworigin* (with tick labels)}
\begin{tikzpicture}[scale=.5]
\tzhelplines(-1,-1)(7,7)
\tzshoworigin*[blue]{$O_1$}[red](5pt)
\tzaxes(-1,-1)(7,7){$x$}{$y$}
\tzshoworigin<0,4>($O_1$){$O'$}[blue,ar](3pt)
\end{tikzpicture}
18.4 \texttt{\textbackslash tzaxesL\textbackslash '}: L-type axes

\texttt{\textbackslash tzaxesL} is similar to \texttt{\textbackslash tzaxes}, but it draws only the ‘L’ type axes with \((x_1,y_1)\) as the origin and \((x_2,y_2)\) as the opposite corner of the rectangle. Those two coordinates are mandatory.

% syntax
\texttt{\textbackslash tzaxesL\{<opt>\}<shift coor>"<path name>"<(x_1,y_1)>(<x_2,y_2)\}
\{<x-axis label>\}[<node opt>]{<y-axis label>}[<node opt>]

% defaults
[]<>"axesL"<(m)>(<m>)\{[right]\}{[above]}% defaults
% arguments:
[#1]: line style, arrow type
<#2>: shift coordinate
"#3": name path=#3 % default: axesL
(#4): (x_1,y_1) % mandatory
(#5): (x_2,y_2) % mandatory
{#6}: x-axis label
[#7]: x-axis label option % node option
{#8}: y-axis label
[#9]: y-axis label option % node option

The \texttt{\textbackslash tzaxesL\textbackslash '} swaps \((x_1,y_1)\) and \((x_2,y_2)\). That is, \texttt{\textbackslash tzaxesL\textbackslash '}(A)(B) is equivalent to \texttt{\textbackslash tzaxesL}(B)(A).

\begin{tikzpicture}[scale=.5]
\node at (0,4) {x};
\node at (8,0) {y};
\draw (8,0) -- (0,0) -- (0,8);
\end{tikzpicture}

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

\begin{tikzpicture}[scale=.5]
\node at (0,4) {x};
\node at (8,0) {y};
\draw (8,0) -- (0,0) -- (0,8);
\end{tikzpicture}

\textit{name path} With the option "\texttt{\textbackslash path name}" , you can name the path of \texttt{\textbackslash tzaxesL} (by default, \texttt{axesL}). This makes it easy to find the intercepts.
19 Ticks

19.1 \tzticks: Tick labels

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

\begin{tikzpicture}[scale=.4,font=\scriptsize]
\helplines(10,10)
\axes(-1,-1)(10,10)
\tzticks[blue]{1,...,8}{2,...,7}
\end{tikzpicture}

You can change the numbered labels to a different format with slashes and other text, as follows: \texttt{<number>/<other text>}.

\begin{tikzpicture}[scale=.4,font=\scriptsize]
\helplines(10,10)
\axes(-1,-1)(10,10)
\tzticks[blue]{1,...,8}{2,...,7}
\end{tikzpicture}
Tick marks  By specifying the options \texttt{(<x-from:x-to>)} for $x$ ticks and/or \texttt{(y-from:y-to)} for $y$ ticks, you can print tick marks. (The default is \texttt{(0pt:0pt)} for both options.)

The position of tick labels does not depend on the length of the tick marks. You can change the position of tick labels using \texttt{[<node opt>]}.

Shift  You can move (or shift) the tick marks and labels together by specifying the optional argument \texttt{(<x-shift,y-shift>)}, where \texttt{<x-shift>} is for $y$-ticks and \texttt{<y-shift>} is for $x$-ticks.
19.2 \zticks*: Tick marks

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

\texttt{\begin{tikzpicture}[scale=.4,font=\scriptsize]
\zhelplines
\tzshoworigin
\tzaxes(-1,-1)(10,10)
\zticks*[draw=red,thick]
\{1,...,7,8/\alpha\} % labels ignored
\{0,0.2,...,7/\beta\} % labels ignored
\end{tikzpicture}}

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

19.3 \zticksx(*) and \zticksy(*)

You can handle x ticks and y ticks independently.

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

\texttt{\begin{tikzpicture}[scale=.4,font=\scriptsize]
\zhelplines
\tzshoworigin
\tzaxes(-1,-1)(10,10)
\zticksx*[\texttt{draw=red,thick}]
\{1,...,7,8/\alpha\} % labels ignored
\texttt{\node at (7,0) \small \texttt{\texttt{\alpha}};}
\{0,0.2,...,7/\beta\} % labels ignored
\end{tikzpicture}}

\texttt{\begin{tikzpicture}[scale=.5]
\zhelplines
\tzaxes(-1,-1)(10,8)
\zticksx*[\texttt{draw=red,thick}]
\{0,0.2,...,8\} % default (Opt:3pt)
\{0,0.2,...,7\} % default (Opt:3pt)
\end{tikzpicture}}
\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 (\ltxt{<x-from>:<x-to>}).
\zticksy* only prints y-ticks from 0pt to 3pt by default, suppressing tick labels.

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

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

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

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

20.1 \texttt{\textbackslash tproj}\texttt{(*)}: Projections on the axes

\texttt{\textbackslash tproj} accepts a mandatory coordinate and draws perpendicular lines onto each axis from the coordinate. The lines are dotted, by default.

\begin{verbatim}
\% syntax: minimum
\tproj{<coor>}
\% syntax: medium
\tproj*{<coor>}{<x-text>}{<y-text>}{<node opt>}
\% syntax: full
\tproj*[<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)
\end{verbatim}

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

\textbf{Dot size} You can only control the size of dots by the last optional argument \texttt{(<dot size>)} or by the \texttt{THREE WAYS} on page \pageref{three-ways}. If you want to control \texttt{fill} or \texttt{color} of dots, use \texttt{\textbackslash tzdot*} separately.

\begin{verbatim}
\%\texttt{\textbackslash tproj(*)}
\begin{tikzpicture}[scale=.5]
\tzhelplines(8,6)
\tzaxes(8,6)
\tproj[dashed,blue](2,3)\{\textit{x}\}\{\textit{y}\}
\tzcoors(30:7)(A)(50:6)(B);
\tproj*[A]
\tproj*[dashed,text=blue](B)(5pt)
\end{tikzpicture}
\end{verbatim}

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

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

\textbf{Projection shift} Specifying the option \texttt{<x-shift,y-shift>} moves the projection point and text on each axis.
20.2 \tzprojsx and \tzprojsy

\tzprojsx draws a dotted line, which is perpendicular to the x axis. \tzprojsx* additionally prints a ‘black node dot’ of the size 2.4pt, by default.

\tzprosy draws a dotted line, which is perpendicular to the y axis. \tzprosy* additionally prints a ‘black node dot’ of the size 2.4pt, by default.

You can only control the size of dots by the last option \langle dot size\rangle. If you want to control fill or color of dots, use \tzdot* separately. You can also add text around the projection point on each axis by specifying the option \langle x-text\rangle or \langle y-text\rangle followed by the option \langle node option\rangle.

Specifying the option <x-shift,y-shift> with \tzprojsx and \tzprojsy moves the projection point and text accordingly.
20.3 \texttt{tzprojs(*): Semicolon versions}

\texttt{tzprojs} accepts any number of coordinates and draws perpendicular lines onto each axis from the coordinates. The lines are dotted, by default. \texttt{tzprojs} is a semicolon version of \texttt{tzproj}, so a semicolon is needed to indicate when the coordinate iteration ends. Its repeating pattern is \( (<\text{coor}>){<\text{x-text}>}[<\text{node opt}>]{<\text{y-text}>}[<\text{node opt}>] \).

\texttt{tzprojs*} additionally prints \texttt{tzdots*} of the 2.4pt (by default) on the coordinates. The first option \([<\text{opt}>]\) does not control the node dots.

You can move the projection points and text accordingly, using the option \(<\text{x-shift},y\text{-shift}>\) before the first coordinate. The empty shift option \(<\text{<>}>\text{ not allowed. You can also change the dot size with the last option \(<\text{dot size}>\) after the semicolon, as in \texttt{tzproj(*)}.

\[x = 3\]
20.4 \texttt{\textbackslash tzprojsx(*)} and \texttt{\textbackslash tzprojsy(*)}: Semicolon versions

\texttt{\textbackslash tzprojsx} is a semicolon versions of \texttt{\textbackslash tzprojx}. It draws dotted lines, which are perpendicular to the x axis from the specified coordinates, by default.

\texttt{\textbackslash tzprojsx*} additionally prints \texttt{\textbackslash tzdots*} of the size 2.4pt, by default.

\texttt{\textbackslash tzprojsy} and \texttt{\textbackslash tzprojsy*} work similarily as \texttt{\textbackslash tzprojsx} and \texttt{\textbackslash tzprojsx*} do but to the y axis.

---

Specifying the option \texttt{<x-shift,y-shift>} moves the projection points and text accordingly.
21 Plot Functions

21.1 \tzfn and \tzfn': Plot functions and inverse functions

21.1.1 \tzfn
\tzfn plots a function of \x.

\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].

\tzfn{.5*(\x)^2-1}[1:5] \% works like:
\draw [samples=201,domain=1:5] plot (\x,{.5*(\x)^2-1});

21.1.2 Inverse functions: \tzfn'

The swap version \tzfn' draws the inverse function of \tzfn.

\tzfn'{.5*(\x)^2-1}[1:5] \% works like:
\draw [samples=201,domain=1:5] plot ({.5*(\x)^2-1},\x);
21.1.3 Define and name functions

To use \texttt{\textbackslash{t}f\textbackslash{n}} you need to express a function as a function of \texttt{x}. 

You can also use the predefined functions of Ti\textsc{k}Z such as \texttt{\textbackslash{sin}}, \texttt{\textbackslash{cos}}, \texttt{\textbackslash{ln}}, \texttt{\textbackslash{log10}}, \texttt{\textbackslash{log2}}, \texttt{\textbackslash{exp}}, \texttt{\textbackslash{sqrt}}, and so on. (See Ti\textsc{k}Z manual.)
21.1.4 Name paths: name path

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

\begin{verbatim}
\texttt{tzn}"mypath"\texttt{Fx}[1:5] \% works like:
\texttt{draw} [samples=201,,domain=1:5,name path=mypath] plot (\texttt{x},\texttt{\{Fx\}});
\end{verbatim}

Remark: Advantage of defining functions:

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

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

\begin{verbatim}
\texttt{tzn}\texttt{\textbackslash Fx}[1:5] \% works like:
\texttt{draw} [samples=201,\domain=1:5,\name path=Fx] plot (\texttt{x},\texttt{\{Fx\}});
\end{verbatim}

\begin{verbatim}
\% \texttt{tzn}: name path: intersection point
\begin{tikzpicture}[scale=.5]
\tzhelplines(8,8)
\tzzaxes(8,8)
\def\Dx{7-\x}
\def\Sx{1+\x}
\texttt{tzn}\texttt{Dx}[0:7]{$\$D\$}[ar] \% name path = Dx
\texttt{tzn} \texttt{\Sx}[0:7]{$\$S\$}[blue,r] \% name path = Sz
\texttt{tzzXpoint(Dx}{Sx)(E){E} \% intersection
\end{tikzpicture}
\end{verbatim}

21.1.5 Move graphs: shift

You can move the graph of \texttt{tzn} by specifying the option \texttt{<shift coor>} before the mandatory argument \{\texttt{<fn of \(x\)>}\} or immediately before the option "\texttt{<path name>}", if it exists. The empty shift option <> is not allowed.

\begin{verbatim}
\% \texttt{tzn}: shift
\begin{tikzpicture}[scale=.6]
\tzhelplines(8,8)
\def\Dx{7-\x}
\def\Sx{1+\x}
\texttt{tzn} \texttt{\Dx}[0:7]{$\$D\$}[right] \% name path = Dx
\texttt{tzn}\texttt{\textbackslash supply}[0:7]{$\$S\$}[right] \% name path = supply
\texttt{tzzXpoint(Dx}{Sx}(E){E}\% intersection
\end{tikzpicture}
\end{verbatim}
21.1.6 Extend paths: `<code.append>`, `{\tzfnAtBegin}`, `{\tzfnAtEnd}

`<code.append>` You can extend the path of `{\tzfn}`, by writing \text{TikZ} code in the \textit{last optional argument} `<code.append>`. Internally it adds the \text{TikZ} code to the path \textit{after} the options `{<text>}` and `[<node opt>].

\begin{tikzpicture}[scale=.5]
\tzhelplines(-2,-2)(8,6)
\tzaxes*(-2,-2)(8,6)
\def\Fx{\sin(\x)+3}
\tzfn[-]{\Fx[-2:2*pi]\{\sin,\x+3\}}[blue,r]
\quad< to [bend right] ++(2,-2) node [b] {End!} >
\end{tikzpicture}

\begin{tikzpicture}[scale=.5]
\tzhelplines(-2,-2)(8,6)
\tzaxes*(-2,-2)(8,6)
\def\Fx{\sin(\x)+3}
\tzfnAtEnd(to [bend right] ++(2,-2) node [b] {End!})
\tzfn[-]{\Fx[-2:2*pi]\{\sin,\x+3\}}[blue,r]
\end{tikzpicture}

Specifying the option `<code.append>` extends the path after `{\tzfnEnd}` if it exists.

\begin{tikzpicture}[scale=.5]
\tzhelplines(-2,-2)(8,6)
\tzaxes*(-2,-2)(8,6)
\def\Fx{\sin(\x)+3}
\tzfnAtBegin{(1,1) -|}
\tzfn[-]{\Fx[-2:2*pi]\{\sin,\x+3\}}[blue,r]
\quad< arc (0:90:4cm) node [draw,red,left] {Here!!} >
\end{tikzpicture}

\begin{tikzpicture}[scale=.5]
\tzhelplines(-2,-2)(8,6)
\tzaxes*(-2,-2)(8,6)
\def\Fx{\sin(\x)+3}
\tzfnAtBegin{(1,1) - |}
\tzfn[-]{\Fx[-2:2*pi]\{\sin,\x+3\}}[blue,r]
\end{tikzpicture}

 `{\tzfnAtBegin}` You can use `{\tzfnAtBegin}` \textit{(immediately) before each `{\tzfn}` to insert \text{TikZ} code at the beginning of the path of `{\tzfn}`.

\begin{tikzpicture}[scale=.5]
\tzhelplines(-2,-2)(8,6)
\tzaxes*(-2,-2)(8,6)
\def\Fx{\sin(\x)+3}
\tzfnAtBegin{(1,1) - |}
\tzfn[-]{\Fx[-2:2*pi]\{\sin,\x+3\}}[blue,r]
\end{tikzpicture}
Remark:

- \tzfn is based on the plot operation of TikZ.
- Appending TikZ code at the beginning of \tzfn may cause a problem when you use some operations (such as to or \relax -1 or \relax -) that expect a coordinate to link.
- In the version 2 of the tzplot package, this issue is internally taken care of by using (current subpath start), which is a special coordinate pre-defined in TikZ. (See TikZ manual for more details.)

21.2 \tzfnofy and \tzfnofy': Functions of variable \( y \)

\tzfnofy plots a function of \( y \). Define a function with the (predefined) variable \( \y \).

\tzfnofy works just like \tzfn but as a function of \( \y \).

```latex
% syntax: minimum
\tzfnofy{<fn of \y>}[<domain>]
% syntax: medium
\tzfnofy{<fn of \y>}[<domain>]{<text>}[<pos>]
% syntax: full
\tzfnofy[<opt>]<shift coor>"<path name>"
{<fn of \y>}[<domain>]{<text>}{<node opt>}{code.append}
% [<domain>] should be of the form [<from num:to num>]
% defaults
[samples=201]\""{\<m>}{\<m>}{\}<>
```

\tzfnofy(2*\y+1)[1:5] \% works like:
\draw [samples=201,domain=1:5,variable=\y] plot (\{2*\y+1\},\y);

The swap version \tzfnofy' plots the inverse function of \tzfnofy.

\tzfnofy'(2*\y+1)[1:5] \% works like:
\draw [samples=201,domain=1:5,variable=\y] plot (\y,\{2*\y+1\});

% \tzfnofy(): variable = \y
\begin{tikzpicture}[scale=.5]
\tzhelplines(8,8)
\tzaxes(8,8){\x}{$x$}{\y}{$y$}
\tzfnofy(5-5/7*\y)[0:7]{\f(y)}[r=5mm]
\tzfnofy'[blue](5-5/7*\y)[0:7]
{\f^{-1}(\y)}[a=5mm,align=center]
\ttzticks(5,7){5,7}
\end{tikzpicture}
21.3 Horizontal lines

21.3.1 \tzhfnt

\tzhfnt draws a horizontal line at a specified value of $y$.

\begin{tikzpicture}[scale=.5]
\def\Fy{7/y}
\tzfnoyAtBegin{(6,3) node [a] {Start} to[bend left]}\Fy(1:7)
\tzfnoyAtEnd{-| ++(3,-1)} \Fy(1:7)
\end{tikzpicture}

\tzhfnt accepts only one mandatory argument \{\textit{y-val}\}. The domain is optional and should be of the form \{\textit{from num:to num}\}. The default domain is from left to right of the current bounding box.

Remark: Internally, the default domain of \tzhfnt depends on the current bounding box.

- Each \tzhfnt may draw a line with a (slightly) different length.
- If an appropriate current bounding box is not formed before \tzhfnt is executed, you will probably get an unexpected result.
- In that case, you can fix a bounding box in the beginning of the \tikzpicture environment using macros such as \tzbbox, \tzhelplines*, \tzaxes*, or Ti\kZ's \useasboundingbox.
You can name the path of \texttt{tzhfnat} by the option "<path name>". You can move the line by the option <shift coor>. You can also extend the path from the end of the line by writing TikZ code in the last optional argument <code.append>.

You can also use \texttt{tzhfnAtBegin} and \texttt{tzhfnAtEnd} to extend the path of \texttt{tzhfnat} at the beginning and at the end. Specifying the option <code.append> extends the path after \texttt{tzhfnAtEnd} if it exists. (See other examples of using \texttt{tz<..>AtBegin} and \texttt{tz<...>AtEnd}.)

21.3.2 \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(<x>,3)}, ignoring \texttt{x}, is equivalent to \texttt{tzhfnat{3}}.

Everything else is the same as in \texttt{tzhfnat}.
You can also use `\tzfnAtBegin` and `\tzfnAtEnd` to extend the path of `\tzfn` at the beginning and at the end. Specifying the option `<code.append>` extends the path after `\tzfnAtEnd` if it exists. (See other examples of using `\tz<..>AtBegin` and `\tz<...>AtEnd`.)

### 21.4 Vertical lines

#### 21.4.1 \tzvfnat

`\tzvfnat` draws a vertical line at a specified value of $x$.

\begin{tikzpicture}
\helplines (4,4)
\tzfnat[blue,thick]{0}
\tzfnat[red,thick]{1}[1:3]{line A}[a]
\tzfnat[blue]{2}{line B}[a]
\tzfnat[red]{3}[0:3]{line C}{draw=blue,red,a}
\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 extend the path from the end of the line by writing TikZ code in the last optional argument `<code.append>`.

\begin{tikzpicture}
\helplines* (4,4)
\tzfn "XX"{\x}{0:4}
\tzfnat[blue,thick]{0}
\tzfnat[red,thick]<.5,-.5>"AA"{1}[1:3]{line A'}[a]
\tzXpoint*{AA}{XX}
\tzvfnat[blue,->]{2}{line B}[a]
< arc (90:-30:1.5) node [b,draw] {End!} >
\end{tikzpicture}

In the previous example, `\helplines*` is used to fix a bounding box. (See Section 7.2 on page 40, for more details.)

You can also use `\tzvfnatAtBegin` and `\tzvfnatAtEnd` to extend the path of `\tzvfnat` at the beginning and at the end. Specifying the option `<code.append>` extends the path after `\tzvfnatAtEnd` if it exists. (See other examples of using `\tz<..>AtBegin` and `\tz<...>AtEnd`.)
21.4.2 \texttt{tzvfn}

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

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

\begin{verbatim}
\% syntax: minimal
\texttt{tzvfn}(\texttt{coor})(\texttt{domain})
\% syntax: full
\texttt{tzvfn}[\texttt{opt}]\texttt{<shift coor>\"path name\"}
\texttt{(\texttt{coor})(\texttt{domain})\{\texttt{text}\}\texttt{<node opt>\texttt{code.append}}
\% defaults
\[\texttt{<>}\{(<m>)\[\texttt{south:north (of current bounding box)}\]\}{}\]\
\end{verbatim}

You can also use \texttt{tzvfnAtBegin} and \texttt{tzvfnAtEnd} to extend the path of \texttt{tzvfn} at the beginning and at the end. Specifying the option \texttt{<code.append>} extends the path after \texttt{tzvfnAtEnd} if it exists. (See other examples of using \texttt{tz<...>AtBegin} and \texttt{tz<...>AtEnd}.)

22 Plot Linear Functions

22.1 \texttt{tzLFn}: Plot linear functions

22.1.1 \texttt{tzLFn} and \texttt{tzLFn’}

Knowing two coordinates or one coordinate with a slope, you can draw a linear function with the macro \texttt{tzLFn}, without writing the explicit definition of a linear function.

- \texttt{tzLFn\{<coor1>\}\{<coor2>\}} is prepared for when you know two coordinates on a line.
  - If you provide two points \((x_1,y_1)\) and \((x_2,y_2)\), \texttt{tzLFnofy(x1,x2)(y1,y2)} draws the graph of \( f(x) = \frac{y_2-y_1}{x_2-x_1}(x-x_1) + y_1 \).
- \texttt{tzLFn\{<coor1>\}\{<slope>\}}\texttt{\{<coor2>\}}\texttt{\{<domain>\}} is used when you know one coordinate and the slope of a line.
- If you specify all the three arguments \texttt{\{<coor1>\}\{<coor2>\}\{<slope>\}\{<domain>\}}, then the slope is ignored.

For example, \texttt{tzLFn\{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{tzLFn\{1,1\}\{.5\}\{0:4\}} draws a line passing through a point \((1,1)\) with a slope .5, over \(0 \leq x \leq 4\).

\texttt{tzLFn} accepts two mandatory arguments: \texttt{\{<coor1>\}} and \texttt{\{<domain>\}}.
- The \texttt{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.

\begin{verbatim}
\% syntax: minimum
\texttt{tzLFn\{<coor1>\}\{<coor2>\}\{<domain>\}} % two coordinates
\texttt{tzLFn\{<coor1>\}\{<slope>\}\{<domain>\}} % one coordinate and slope
\end{verbatim}
You can add text at the end of the line of \texttt{tzLFn} by the options \{<text>\} and [\texttt{<node opt>\}]. You can also name the path of \texttt{tzLFn} by specifying the option "\texttt{<path name>}" immediately before the mandatory coordinate.

\begin{tikzpicture}
\helplines(4,3)
\coors*(1,1)(A)(1,2)(B)(3,1)(C);
\tzcoors*{blue}{Gx}(B)(C)[0:4]{$g(x)$}[r]
\end{tikzpicture}

% \texttt{tzLFn}: two coordinates
\begin{tikzpicture}
\helplines(4,3)
\coors*(1,1)(A)(1,2)(B)(3,1)(C);
\tzcoors*{red}{Fx}(A){.5}[0:4]{$f(x)$}[a]
\end{tikzpicture}

**Remark:** If you inadvertently try an infinite slope, you will get an error message.

**Inverse function** The \texttt{swap version \texttt{tzLFn}} draws the inverse function of \texttt{tzLFn}.

\begin{tikzpicture}
\helplines(4,4)
\coors*(1,1)(A)(1,2)(B)(3,1)(C);
\tzcoors*{red}{Fx}(A){.5}[0:4]{$f(x)$}[a]
\tzcoors*{blue}{Gx}(B)(C)[0:4]{$g(x)$}[r]
\tzcoors*{fill=none}{Fx}(Gx)(E){(E)}(3pt)
\end{tikzpicture}

You can move the line of \texttt{tzLFn} by specifying the option \texttt{<shift coor>} immediately before the option "\texttt{<path name>}". (The empty shift option \texttt{<>} is not allowed.) You can also extend the path of \texttt{tzLFn} by writing TikZ code in the last optional argument \texttt{<code.append>}.  

\begin{tikzpicture}
\helplines(4,4)
\coors*(1,1)(A)(1,2)(B)(3,1)(C);
\tzcoors*{red}{Fx}(A){.5}[0:4]{$f(x)$}[a]
\tzcoors*{blue}{Gx}(B)(C)[0:4]{$g(x)$}[r]
\end{tikzpicture}

\texttt{tzLFnAtBegin} and \texttt{tzLFnAtEnd} are available to extend a path of \texttt{tzLFn} at the beginning and the end, respectively. Specifying the option \texttt{<code.append>} extends the path after \texttt{tzLFnAtEnd}, if it exist.
\begin{tikzpicture}[scale=.8]
  \helplines(4,4)
  \coors*(1,1)(A)(1,2)(B)(3,1)(C);
  \LFn{Gx}(B)(C){0:4}{\$g(x)\$}[r]
  \LFnAtBegin{(1,0) to [bend left] }
  \LFnAtEnd{ arc (0:140:2) }
  \LFn[dashed,red,->]<1,1>"Gx"(B)(C){0:4}{\$g(x)\$}[r]
\end{tikzpicture}

\begin{tikzpicture}[scale=.8]
  \helplines(4,4)
  \coors*(1,1)(A)(1,2)(B)(3,1)(C);
  \LFnofy{Fx}(A){.5}{\$f(y)\$}[a]
  \LFnofy{Gx}(B)(C){.5:2.5}{\$g(y)\$}[a]
  \Xpoint*[fill=none]{Fx}{Gx}(E){\$E\$}(3pt)
\end{tikzpicture}

\begin{tikzpicture}[scale=.8]
  \helplines(4,4)
  \coors*(1,1)(A)(1,2)(B)(3,1)(C);
  \LFnofy{Fx}(A){.5}{\$f(y)\$}[a]
  \LFnofy{Gx}(B)(C){.5:2.5}{\$g(y)\$}[a]
  \Xpoint*[fill=none]{Fx}{Gx}(E){\$E\$}(3pt)
\end{tikzpicture}

\begin{tikzpicture}[scale=.8]
  \helplines(4,4)
  \coors*(1,1)(A)(1,2)(B)(3,1)(C);
  \LFnofy'{Fx}(A){.5}{\$f(y)\$}[a]
  \LFnofy'{Gx}(B)(C){.5:2.5}{\$g(y)\$}[r]
  \Xpoint*[fill=none]{Fx}{Gx}(E){\$E\$}(3pt)
\end{tikzpicture}

\begin{tikzpicture}[scale=.8]
  \helplines(4,4)
  \coors*(1,1)(A)(1,2)(B)(3,1)(C);
  \LFnofyAtBegin{E}
  \LFnofyAtEnd{E}
\end{tikzpicture}

% \\text{Remark:} If you inadvertently try an infinite slope, you will get an error message.

\textbf{Inverse function} The swap version \texttt{\LFnofy'} draws the inverse function of \texttt{\LFnofy}.

You can use \texttt{\LFnofyAtBegin} and \texttt{\LFnofyAtEnd} to extend the path of \texttt{\LFnofy} at the beginning and at the end. Specifying the optional argument \texttt{<code.append>} extends the path after \texttt{\LFnofyAtEnd} if it exists. (See other examples of using \texttt{\t<...>AtBegin} and \texttt{\t<...>AtEnd}.)
22.2 \texttt{\tzdefLFn}

\texttt{\tzdefLFn} simply defines a linear function $ax + b$ and saves it to a macro. You can use \texttt{\tzdefLFn} together with \texttt{\tzfn} to graph a linear function, \textit{without writing an explicit definition} of a linear function. (Of course, you can directly use \texttt{\tzLFn}.)

\begin{verbatim}
% syntax
\tzdefLFn{<fn csname>}{<coor1>}{<coor2>}{<slope>}
% defaults
{<m>}{<m>}{<m>}{1}
\end{verbatim}

If \texttt{<coor2>} is specified, \texttt{<slope>} is ignored. If \texttt{<coor2>} is missing \texttt{<slope>} is considered as the slope of the line (by default \texttt{1}).

For example, \texttt{\tzdefLFn{\Gx}(1,1){.5}} defines as \texttt{\Gx} a linear function passing through the point \texttt{(1, 2)} with a slope \texttt{.5}. That is, it defines a function as $f(x) = .5(x - 1) + 1$. Knowing two coordinates, linear function \texttt{\Gx} passing through the two points can be defined, for example, by \texttt{\tzdefLFn{\Gx}(1,2)(3,1)}. That is, it defines a function as $g(x) = \frac{1-2}{3-1}(x - 1) + 2$.

\begin{verbatim}
\tzdefLFn{\Fx}(1,2)(3,1) \% works like \def{\Fx}{-1/2*(x-1)+2}
\end{verbatim}

Remark: If you inadvertently try an infinite slope, you will get an error message.

\begin{verbatim}
% \tzdefLFn and \tzfn
\begin{tikzpicture}
\helplines(4,4)
\axes(4,4){$x$}{$y$}
\ticks{1,2,3,4}{1,2,3,4}
\coors*(1,1)(A){A}[90](1,2)(B){B}(3,1)(C){C}[45];
\tzdefLFn{\Fx}(A){.5}
\tzdefLFn{\Gx}(B)(C)
\tzfn[red]{\Fx}[0:4]{$f(x)$}[r]
\tzfn[blue]{\Gx}[0:4]{$g(x)$}[r]
\end{tikzpicture}
\end{verbatim}

Remark: The swap version \texttt{\tzfn'} simply draws the graph of the inverse function of \texttt{\tzfn}. So \texttt{\tzfn'(A)(B)} and \texttt{\tzfn'(A){.5}} do not guarantee passing through the coordinate \texttt{(A)} or \texttt{(B)}.

\begin{verbatim}
\begin{tikzpicture}
\helplines(4,4)
\axes(4,4){$x$}{$y$}
\ticks{1,2,3,4}{1,2,3,4}
\coors*(1,1)(A){A}[90](1,2)(B){B}(3,1)(C){C}[45];
\tzdefLFn{\Fx}(A){.5}
\tzdefLFn{\Gx}(B)(C)
\tzfn'[red]{\Fx}[0:4]{$f(x)$}[r]
\tzfn'[blue]{\Gx}[0:4]{$g(x)$}[r]
\end{tikzpicture}
\end{verbatim}

22.3 \texttt{\tzdefLFnofy}

\texttt{\tzdefLFnofy} defines a function of \texttt{\y}. 

163
% syntax
\tzdefLFnofy\{fn\ (csname)\}(<coor1>)(<coor2>){<slope>}
% defaults
\{m\}\{m\}\{1\}

If (<coor2>) is specified, {<slope>} is ignored. If (<coor2>) is missing <slope> is considered as the slope of the line (by default 1).

\tzdefLFnofy\Fy (1,1){.5} % works like \
def\Fy(1.5*(\y-1)+1)

\tzdefLFnofy\Gy (1,2)(3,1) % works like \
def\Gy(-2*(\y-2)+1)

Remark: If you inadvertently try an infinite slope, you will get an error message.
You can use \tzdefLFnofy together with \tzFnofy to graph a linear function of y. (Of course, you can directly use \tzLFnofy.)

% \tzdefLFnofy and \tzLFnofy
\begin{tikzpicture}
\tzhelplines(4,4)
\tzaxes*(4,4)\{\$x\}\{\$y\}
\tzticks(1,2,3,4){1,2,3,4}
\tzcoors*(1,1)(A){\$A\}[-90](1,2)(B){\$B}(3,1)(C){\$C}[45];
\tzdefLFnofy\Fy(A){.5}
\tzdefLFnofy\Gy(B)(C)
\tzfnofy[red]\Fy[0:4]\{\$f(y)\}[r]
\tzfnofy[blue]\Gy[0:4]\{\$g(y)\}[a]
\end{tikzpicture}

Remark: The swap version \tzfnofy’ simply draws the graph of the inverse function of \tzfnofy. So \tzfnofy’(A)(B) and \tzfnofy’(A){.5} do not guarantee passing through the coordinate (A) or (B).

% \tzdefLFnofy and \tzLFnofy’
\begin{tikzpicture}
\tzhelplines(4,4)
\tzaxes*(4,4)\{\$x\}\{\$y\}
\tzticks(1,2,3,4){1,2,3,4}
\tzcoors*(1,1)(A){\$A\}[-90](1,2)(B){\$B}(3,1)(C){\$C}[45];
\tzdefLFnofy\Fy(A){.5}
\tzdefLFnofy\Gy(B)(C)
\tzfnofy’[red]\Fy[0:4]\{\$f(y)\}[r]
\tzfnofy’[blue]\Gy[0:4]\{\$g(y)\}[r]
\end{tikzpicture}
23 Some More Functions

23.1 $\texttt{tzt\textbackslash pdf}\texttt{N}$ and $\texttt{tzt\textbackslash pdf}\texttt{Z}$: Normal distributions

$\texttt{tzt\textbackslash pdf}\texttt{N}$, $\texttt{tzt\textbackslash pdf}\texttt{N}$*, and $\texttt{tzt\textbackslash pdf}\texttt{Z}$ are predefined functions to plot the probability density function (pdf) of a normal distribution $N(\mu, \sigma^2)$.

Normal distributions $\texttt{tzt\textbackslash pdf}\texttt{N}$ accepts two mandatory arguments, \{<mean>\} $\mu$ and \{<variance>\} $\sigma^2$, to define the pdf function:

$$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{x-\mu}{\sigma} \right)^2}$$

$\texttt{tzt\textbackslash pdf}\texttt{N}$* uses a standard deviation $\sigma$ instead of a variance.

With these predefined functions together with $\texttt{tzt\textbackslash fn}$ you can plot the pdf of normal distributions.

\begin{tikzpicture}
\begin{scope}[y=0.5cm, x=5cm, font=\scriptsize]
\tzhelplines[ystep=0.1](-4,0)(6,.5)
\tzaxes(-4,0)(6,.5)
\tzticks{1,...,5}{0.1,0.2,0.3,0.4}
\tzfn[blue]{$\texttt{tzt\textbackslash pdf}\texttt{N}\{0\,1.5\}$}[-4:4]
\tzfn[red]{$\texttt{tzt\textbackslash pdf}\texttt{N}\{*\{2\,\sqrt{1.5}\}$][-2:6]
\end{scope}
\end{tikzpicture}

Remark: If \texttt{tikzpicture} is scaled when plotting the inverse function using $\texttt{tzt\textbackslash fn}'$, you may want to change the scale accordingly.

\begin{tikzpicture}
\begin{scope}[xscale={1/0.5*5}, yscale={1/5*0.5}]
\tzfn[blue]{$\texttt{tzt\textbackslash pdf}\texttt{N}\{0\,1.5\}$}[-4:4]
\tzfn[red]{$\texttt{tzt\textback\textbackslash pdf}\texttt{N}*$\{2\,\sqrt{1.5}\}[-2:6]
\end{scope}
\end{tikzpicture}

Standard normal distributions $\texttt{tzt\textbackslash pdf\textbackslash Z}$ (with no arguments) represents the pdf of the standard normal distribution.
23.2 \texttt{tzfnarea(*)}: Fill under the graph

\texttt{tzfnarea} fills the area between the graph of a function and the x axis, with a color or pattern, on the \texttt{behind} layer by default. With \texttt{settzfnarealayer}, you can change the layer of \texttt{tzfnarea}.

\texttt{tzfnarea} accepts two mandatory arguments: \{\texttt{fn of }x\}\ and \[\texttt{domain}\], like \texttt{tzfn}.

\begin{verbatim}
\begin{tikzpicture}
\helplines[step=0.5cm](-4,0)(6,.5)
\ticks{1,...,5}{0.1,0.2,0.3,0.4}
\function[blue]{tzpdfZ}[-3:3] \%	exttt{tzpdfZ}
\function[red]{2,0}{tzpdfZ}[-3:3] \%	exttt{shift}
\function[dashed]{3,1}{tzpdfZ}[-3:3] \%	exttt{shift}
\end{tikzpicture}
\end{verbatim}

The starred version \texttt{tzfnarea*} fills the area with \texttt{fill=black!50} and \texttt{fill opacity=.3}, and \texttt{text opacity=1}, by default. The default values can be changed by \texttt{settzfillcolor}, \texttt{settzfillopacity}, and the option \{\texttt{fill opacity}\}.  

\begin{tikzpicture}
\helplines[step=.5cm](-4,-.1)(6,.5)
\axes(-4,-.1)(6,.5)
\fn{\pdfZ}{-3:3}
\fnarea*[\pattern=north east lines]{\pdfZ}{-3:1}
\fn{\text{"AA"} \pdfN_{21}}{-1:5}
\fnarea*[\pattern=north east lines]{\pdfN_{21}}{1:3}
\end{tikzpicture}

\begin{tikzpicture}
\helplines(0,-1)(4,2)
\axes(0,-1)(4,2)
\def\Fx{.5*(x-1)^2-1}
\fn{\Fx}{0:3.5}
\fnarea*[red]{\Fx}{1:3}{.1}
\node[black] at (2,-1) {Area}
\end{tikzpicture}

\begin{tikzpicture}
\helplines(0,-2)(4,3)
\axes(0,-2)(4,3)
\def\Fx{.5*(x-1)^2-1}
\fn{\Fx}{0:3.5}[a]
\fnarea*[red]{\Fx}{1:3}
\vXpointat{\Fx}{1}(A)
\vXpointat{\Fx}{3}(B)
\line[blue,thick](A)(A|-0,0)
\line[very thick](B)(B|-0,0)
\end{tikzpicture}

\subsection{\texttt{tzfnarealine}}
\texttt{tzfnarealine} draws one or two boundary lines of \texttt{tzfnarea} using \texttt{tzto}, on the \texttt{behind} layer by default. The layer can be changed by \texttt{settzfnarealayer}. It takes two mandatory arguments: \{<path>\} and \{<x1>\}.

- \texttt{tzfnarealine}<path>{<x1>} draws a vertical line to the x axis at \texttt{<x1>}.
- \texttt{tzfnarealine}<path>{<x1>}{<x2>} draws two vertical lines at \texttt{<x1>} and \texttt{<x2>}.

The first option [\texttt{opt}] controls both lines and is overwritten by each of [\texttt{opt1}] (for the line at \texttt{<x1>}) and [\texttt{opt2}] (for the line at \texttt{<x2>}). The line width is very thin by default, which can be changed by \texttt{settzfnarealinestyle}.

You can also change the length of lines by specifying \{<coor1>\} and \{<coor2>\}.

% syntax: minimum
\texttt{tzfnarealine}<path>{<x1>}

% syntax: minimal
\texttt{tzfnarealine}<path>{<x1>}{<x2>}

% syntax: medium
\texttt{tzfnarealine}<path>{<x1>}[<opt1>]{<x2>}[<opt2>]
The `\tzfnarealine` command draws one or two horizontal lines from the point \((x_i, f(x_i))\) to the \(y\) axis. Everything else is the same as in `\tzfnarealealine`. 

\begin{tikzpicture}
\helplines[step=.5cm](-4,-.1)(6,.5)
\axes(-4,-.1)(6,.5)
\psy\{AA\}[<opt>]{<path>}{<x1>}{<opt1>}{<coor1>}
\psy\{AA\}{1}{3}
\end{tikzpicture}
23.3 Envelope curves

23.3.1 \texttt{tzfnmax}: Upper envelope curves

\texttt{tzfnmax} plots the maximum of a list of functions, that is, an *upper envelope curve*. The first mandatory argument should be a comma separated list of functions. The second mandatory argument \texttt{[<domain>]} should be colon separated.

The default line width of \texttt{tzfnmax} is \texttt{thick}.

\begin{verbatim}
% syntax: minimum
\tzfnmax{<fn list>}{<domain>}
% syntax: full
\tzfnmax[<opt>]<shift coor>"<path name>"
\{<fn list>}{<domain>}{<text>}{<node opt>}{code.append}
% remark:
- \{<fn list>\} should be comma separated
- \[<domain>\] should be of the form \[<from num:to num>\]
% defaults
[thick,samples=201]<>"{}{"{}{}\}{}{"{
\end{verbatim}

Remark: If you want sharp corners at kinked points, you may need to select an appropriate number of sample points, which is samples=201, by default. You can try an odd number.

If you want, you can shift and extend path using the option \texttt{<shift coor> and <code.append>}, respectively. You can also use \texttt{tzfnmaxAtBegin} and \texttt{tzfnmaxAtEnd} to extend the path of \texttt{tzfnmax} at the beginning and at the end, respectively.

\begin{verbatim}
% shift, <code.append>
\begin{tikzpicture}
\tzhelplines*(4,4)
\def\Fx{\x}
\def\Gx{.5*\x+.5}
\def\Hx{-.25*\x+2}
\tzfn(\Fx)[0:4] \tzfn(\Gx)[0:4] \tzfn(\Hx)[0:4]
\tzfnmax[blue]{\Fx,\Gx,\Hx}[0:4]{upper envelope}[1]
\end{tikzpicture}
\end{verbatim}

23.3.2 \texttt{tzfnmin}: Lower envelope curves

\texttt{tzfnmin} plots the minimum of a list of functions. \texttt{tzfnmin} works just like \texttt{tzfnmax}, but it draws a *lower envelope curve*.
Remark: If you want sharp corners at kinked points, you may need to select appropriate number of sample points. Try odd number.

To extend the path of \texttt{tzfnmin}, you can use the option \texttt{<code.append>} or the macros \texttt{tzfnminAtBegin} and \texttt{tzfnminAtEnd}.

24 Intersections

24.1 \texttt{tzXpoint(\ast)}: Intersection points

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

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

You can determine which intersection point is named directly by specifying the option \texttt{<nth>}. If you select the second intersection point to be named \texttt{(A)} out of multiple intersections, they are named as follows: \texttt{(A-1)}, \texttt{(A-1)}=\texttt{(A-2)}, \texttt{(A-3)}, \texttt{(A-4)}, etc. You can label intersection points by specifying the option \texttt{<label>} and \texttt{<angle>}.
\begin{tikzpicture}[scale=.5]
\tzhelplines(8,8)
\tzaxes(8,8)
\tzto[red,bend right]"AA"(1,8)(8,1)
\tzto[blue,bend right]"BB"(0,2)(8,6)
\tzXpoint[AA]{}{BB}{A}
\tzdot*(A)
\end{tikzpicture}

Warning: When using \tzXpoint, the intersection of two paths must actually exist. Otherwise, an error will occur when using coordinates.

\tzXpoint* The starred version \tzXpoint* simply adds a node dot to \tzXpoint. The default dot size is 2.4pt and it can be changed by the last option (\texttt{<dot size>}) or the THREE WAYS (on page 46).

% syntax: minimum
\tzXpoint*{<path>}{<path>}
% syntax: medium
\tzXpoint*{<path>}{<path>}{<coor name>}{<label>}{<angle>}
% syntax:
\tzXpoint*{<opt>}{<path>}{<path>}
% (\texttt{<coor name>\{\texttt{<nth>}{\texttt{<label>}{\texttt{<label opt>}{\texttt{angle}>}}}}(\texttt{<dot size>})
% defaults
[tzdot=2.4pt]{<m>}{<m>}(intersection){1}{}{2.4pt}

% syntax: minimum
\tzvXpointat(*)
% \tzvXpointat determines vertical intersection points of a path at a specified value of \texttt{x}. So it takes \{\texttt{<path>}\} and \{\texttt{<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

24.2 Vertical intersection points

24.2.1 \tzvXpointat(*)

\tzvXpointat determines vertical intersection points of a path at a specified value of \texttt{x}. So it takes \{\texttt{<path>}\} and \{\texttt{<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 \texttt{tzbbox} or \texttt{tzaxes*} or TikZ's \texttt{useasboundingbox}.

```latex
% syntax: minimal
\tzvXpointat{<path>}{<x-val>}(<coor name>)
% syntax: medium
\tzvXpointat{<path>}{<x-val>}(<coor name>{<label>}[<angle>]
% syntax: full
\tzvXpointat[<opt>]{<path>}{<x-val>}(<coor name>)[<n>]
    {<label>}{[<label opt]angle}]
% defaults
\{<m>}{<m>}(intersection){1}{1}
```

The starred version \texttt{\tzvXpointat*} additionally prints a node dot of the size 2.4pt, by default, at the (first) intersection point.

```latex
% syntax: minimal
\tzvXpointat*{<path>}{<x-val>}
% syntax: medium
\tzvXpointat*{<path>}{<x-val>}(<coor name){<label>}[<angle>]
% syntax: full
\tzvXpointat*[<opt>]{<path>}{<x-val>}(<coor name>)[<n>]
    {<label>}{[<label opt]angle}(<dot size>)
% defaults
\tzdot{2.4pt}{<m>}{<m>}(intersection){1}{1}(2.4pt)
```

\subsection{tzvXpoint(*)}

\texttt{\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{\tzvXpoint{mypath}{3}{<y>}}, ignoring \texttt{<y>}, is equivalent to \texttt{\tzvXpointat{mypath}{3}}.

\emph{Everything else is the same as in \tzvXpointat}.

```latex
\begin{tikzpicture}
\tzhelplines(4,3)
\def\Fx{.5*(x-2)^2}
\tzfn\Fx[0:4] % name path=Fx
\tzvfnat[dashed]{1}
\tzvXpointat{Fx}{1}(A)
\tzdot(A){A}[45]
\tzvXpointat*{Fx}{3.2}{B}[[red,draw]br]
\end{tikzpicture}
```

The starred version \texttt{\tzvXpoint*} just adds a node dot to \texttt{\tzvXpoint}.
\texttt{\textbackslash tzvXpoint*} prints, at the first intersection point, a node dot of the size 2.4pt by default.

\begin{tikzpicture}
\begin{helpLines}(4,3)
\def\Fx{.5*(x-2)^2}
\tzfn\Fx[0:4] % name path=Fx
\tzcoors(1,0)(A)(3.2,0)(B);
\tzvfn[dashed](1,0)
\tzvXpoint(Fx)(A)(Ax)
\tzdot(Ax)(Ax)[45]
\tzvXpoint*[Fx](B){B}[red,draw][br] % abb
\end{tikzpicture}

\section{24.3 Horizontal intersection points}

\subsection{\texttt{\textbackslash tzhXpointat(*)}}

\texttt{\textbackslash tzhXpointat} determines horizontal intersection points of a path at a specified value of \textit{y}. So it takes \{\texttt{<path>}\} and \{\texttt{<y-val>}\} as mandatory arguments.

\begin{tikzpicture}
\begin{helpLines}(4,3)
\def\Fx{.5*(x-2)^2}
\tzfn\Fx[0:4] % name path=Fx
\tzcoors(1,0)(A)(3.2,0)(B);
\tzvfn[dashed](1,0)
\tzvXpointat(Fx)(A)(Ax)
\tzdot(Ax)(Ax)[45]
\tzvXpointat*[Fx](B){B}[red,draw][br] % abb
\end{tikzpicture}

The starred version \texttt{\textbackslash tzhXpointat*} additionally prints a node dot of the size 2.4pt, by default, at the intersection point.

\begin{tikzpicture}
\begin{helpLines}(4,3)
\def\Fx{.5*(x-2)^2}
\tzfn\Fx[0:4] % name path=Fx
\tzcoors(1,0)(A)(3.2,0)(B);
\tzvfn(dashed)(1,0)
\tzvXpointat(Fx)(A)(Ax)
\tzdot(Ax)(Ax)[45]
\tzvXpointat*[Fx](B){B}[red,draw][br] % abb
\end{tikzpicture}
24.3.2 \texttt{tzhXpoint(*)}

\texttt{tzhXpoint} 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{tzhXpoint(mypath)\{(<>),3\}}, ignoring \(<x>\), is equivalent to \texttt{tzhXpointat{mypath}\{3\}}.

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

\texttt{tzhXpoint*} just adds a node dot to \texttt{tzhXpoint}.

\texttt{tzhXpoint*} prints, at the (first) intersection point, a node dot of the size 2.4pt, by default.

24.4 \texttt{tzhLFnXpoint(*)}: Intersection point of linear functions

Sometimes you may want to quickly find intersection point of two linear functions \textit{without specifying path names}.

\texttt{tzhLFnXpoint} finds the solution of two linear functions without printing anything by default. You can name it and use it.
The starred version \texttt{\textbackslash tzLFnXpoint*} additionally print a filled node dot at the solution of two linear functions of \texttt{x}.

\begin{tikzpicture}[scale=.8]
\zhelplines(4,3)
\tzLFnXpoint*{\x}{3-\x}(A){A}[\[red,draw\]90] % invisible
\tzline+[blue](A)(2,-1)
\end{tikzpicture}

\section{Secant and Tangent Lines}

\subsection{Secant lines}

\texttt{\textbackslash tzsecant} draws a line segment or a \textit{secant} line of a curve, on the \texttt{behind} layer by default. \texttt{\textbackslash tzsecant} accepts three mandatory arguments. Three mandatory arguments are a path name and two values of \texttt{x}: \{\texttt{path}\}, \{\texttt{from-x}\}, and \{\texttt{to-x}\}. With \texttt{\settzsecantlayer}, you can change the layer, like \texttt{\settzsecantlayer{main}}.
Domain  The domain of the form \texttt{[from num:to num]} is optional. Without specifying the optional domain, \texttt{\textbackslash tzsecantat} draws a line segment connecting two points on the (curved) path.

Specifying the option \texttt{[domain]}, you can extend (or shorten) the line of \texttt{\textbackslash tzsecantat}.

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}, and use it to find intersection points.
\begin{tikzpicture}[scale=.5]
\helplines (8,6)
\axes (-1,-1)(8,6){$x$}{$y$}
\bezier["curve"/.expanded](.5,1)(1,6)(-1,-4)(7,5){curve}[ar]
\secantat[thick,red]{curve}{1}{4}
\secantat[blue]{curve}{1}{3}[0:5]{secant line}[a]
\secantat[blue]<1,-1>"shift"{curve}{1}{3}[0:5]
\vPointat*{shift}{3}{X}[-45]
\vTicks{x}{(-1mm:2mm)}{3/$x=3$}
\end{tikzpicture}

\include{code}

\begin{tikzpicture}[scale=.5]
\helplines (8,6)
\axes (-1,-1)(8,6){$x$}{$y$}
\bezier["curve"/.expanded](.5,1)(1,6)(-1,-4)(7,5){curve}[ar]
\secantat[blue]{curve}{1}{3}[0:5]{secant line}[a]
\secantat[blue,->]<1,-1>"shift"{curve}{1}{3}[0:5]
\vPointat*{shift}{3}{X}[-45]
\end{tikzpicture}

\section{\texttt{tzsecant}}
\texttt{tzsecant} uses two \emph{coordinates} \emph{instead of two values of $x$} to draw a line segment or a \emph{secant} line of a curve, on the \emph{behind} layer by default. You need to specify a path name and two coordinates, then \texttt{tzsecant} uses the $x$ values of the two coordinates.

\emph{Everything else is the same as in \texttt{tzsecant}.}

You can change the layer with \texttt{settzsecantlayer}.

\begin{quote}
% syntax: minimum
\texttt{\tzsecant\{<path>\}(<coor>)(<coor>)}
% syntax: medium
\texttt{\tzsecant\{<path>\}(<coor>)(<coor>)[<domain>][<text>][<node opt>]}\texttt{\code.append}
% syntax: full
\texttt{\tzsecant\{<opt>\}<shift>|<coor>|"path name"
{<path>}{(<coor>)(<coor>)[<domain>][<text>][<node opt>]}\texttt{\code.append}
% defaults
{}\texttt{<\{}<\{}<\{}<\{}<\{}<\{}\texttt{>\{}<\{}<\{}<\{}<\{}<\{}<\{<\}
\end{quote}

The domain should be of the form \texttt{[<from num>\texttt{to num}>]. Without specifying the optional domain, \texttt{tzsecant} draws a line segment connecting two points on the (curved) path.
25.2 Tangent lines

25.2.1 \tzttangentat
\tzttangentat draws a tangent line to a curve at a specified value of \textit{x}. Three mandatory arguments are a curve name, a value of \textit{x}, and a domain: \{<path>, <x-val>, and [domain]. By default, the tangent line is drawn on the behind layer, which can be changed by \settztangentlayer, like \settztangentlayer{main}.

\textbf{Remark:} To calculate the slope at \textit{x}, \textit{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 approximate.

\begin{verbatim}
% syntax: minimum \tzttangentat{<path>}{<x-val>}{<domain>}
% syntax: medium \tzttangentat{<path>}{<x-val>}{<domain>}{<text>}{<node opt>}
% syntax: full \tzttangentat{<opt>}{<shift coor>"<path name>"}{<path>}{<x-val>}{<epsilon1>}{<epsilon2>}{<domain>}{<text>}{<node opt>}{<code.append>}
% The domain should be of the form [<from num:to num>]
% defaults
{}<>{"<m>}{<m>}(0.01,0.01){<m>}{<m>}{<m>}{<m>}{<m>}
\end{verbatim}

\textbf{Domain} The mandatory argument [domain] should be of the form [from num:to num].
% \	exttt{tztangentat}\\begin{tikzpicture}[scale=.5]\\zhelplines(9,8)\\tzaxes(9,8)\\tztplotcurve"AA"(1,7)(3,3)(8,1);\\tztangentat{AA}{2}[.5:4]\\tztangentat[blue]{AA}{4}[1:7]{tangent}[red,b]\\tzvXpointat\*\{AA\}{4}\\tzticksx\{1,7\}\\end{tikzpicture}

\textbf{Shift} \hspace{1em} You can move the tangent line by specifying the option \texttt{<shift coor>}. \\

% \texttt{tztangentat: shift}\\begin{tikzpicture}[scale=.5]\\zhelplines(9,8)\\tzaxes(9,8)\\tztplotcurve"AA"(1,7)(3,3)(8,1);\\tztangentat{AA}{2}[.5:4]\\tztangentat[blue]{AA}{4}[1:7]{tangent}[red,b]\\tztangentat[blue]<2,1>\{AA\}{4}[1:7]{tangent'}[r]\\tzvXpointat\*\{AA\}{4}\\tzticksx\{1,7\}\\end{tikzpicture}

\textbf{Naming paths} \hspace{1em} By specifying the option "\texttt{<path name>}", you can name the path of \texttt{tztangentat} and use it to find intersection points. \\

% \texttt{tztangentat: name path (intersection)}\\begin{tikzpicture}[scale=.5,font=\footnotesize]\\zhelplines(9,8)\\tzaxes(9,8)\\tztplotcurve"AA"(1,7)(3,3)(8,1);\\tztangentat{AA}{2}[.5:4]\\tztangentat[blue]{AA}{4}[1:7]{tangent}[red,b]\\tzvXpointat\*\{AA\}{4}\\tztangentat[blue]<2,1>"BB"\{AA\}{4}[1:7]{tangent'}[x]\\tzvXpointat\*\{blue\}\{BB\}\{X\}\{\$x_1\}\{\$x_2\}\{\$x_3\}\{\$x_4\}\{\$x_5\}\{\$x_6\}\{\$x_7\}\{\$x_8\}\{\$x_9\}\{\$x_{10}\}\{\$x_{11}\}\{\$x_{12}\}\{\$x_{13}\}\{\$x_{14}\}\{\$x_{15}\}\{\$x_{16}\}\{\$x_{17}\}\{\$x_{18}\}\{\$x_{19}\}\{\$x_{20}\}\{\$x_{21}\}\{\$x_{22}\}\{\$x_{23}\}\{\$x_{24}\}\{\$x_{25}\}\{\$x_{26}\}\{\$x_{27}\}\{\$x_{28}\}\{\$x_{29}\}\{\$x_{30}\}\{\$x_{31}\}\{\$x_{32}\}\{\$x_{33}\}\{\$x_{34}\}\{\$x_{35}\}\{\$x_{36}\}\{\$x_{37}\}\{\$x_{38}\}\{\$x_{39}\}\{\$x_{40}\}\{\$x_{41}\}\{\$x_{42}\}\{\$x_{43}\}\{\$x_{44}\}\{\$x_{45}\}\{\$x_{46}\}\{\$x_{47}\}\{\$x_{48}\}\{\$x_{49}\}\{\$x_{50}\}\{\$x_{51}\}\{\$x_{52}\}\{\$x_{53}\}\{\$x_{54}\}\{\$x_{55}\}\{\$x_{56}\}\{\$x_{57}\}\{\$x_{58}\}\{\$x_{59}\}\{\$x_{60}\}\{\$x_{61}\}\{\$x_{62}\}\{\$x_{63}\}\{\$x_{64}\}\{\$x_{65}\}\{\$x_{66}\}\{\$x_{67}\}\{\$x_{68}\}\{\$x_{69}\}\{\$x_{70}\}\{\$x_{71}\}\{\$x_{72}\}\{\$x_{73}\}\{\$x_{74}\}\{\$x_{75}\}\{\$x_{76}\}\{\$x_{77}\}\{\$x_{78}\}\{\$x_{79}\}\{\$x_{80}\}\{\$x_{81}\}\{\$x_{82}\}\{\$x_{83}\}\{\$x_{84}\}\{\$x_{85}\}\{\$x_{86}\}\{\$x_{87}\}\{\$x_{88}\}\{\$x_{89}\}\{\$x_{90}\}\{\$x_{91}\}\{\$x_{92}\}\{\$x_{93}\}\{\$x_{94}\}\{\$x_{95}\}\{\$x_{96}\}\{\$x_{97}\}\{\$x_{98}\}\{\$x_{99}\}\{\$x_{100}\}\{\$x_{101}\}\{\$x_{102}\}\{\$x_{103}\}\{\$x_{104}\}\{\$x_{105}\}\{\$x_{106}\}\{\$x_{107}\}\{\$x_{108}\}\{\$x_{109\end{tikzpicture}

\texttt{<code.append>} \hspace{1em} You can extend the path of \texttt{tztangentat} by writing Ti\textsc{k}Z code in the last optional argument \texttt{<code.append>}.
Variations Since the slope of the tangent line is approximate, 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 $(\text{x-val})$. 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.

25.2.2 \ztangent

\ztangent uses a coordinate instead of a value of x to draw a tangent line to a curve. \ztangent accepts three mandatory arguments: $(\text{path})$, $(\text{coor})$, and $(\text{domain})$.

The value of y in $(\text{coor})$ is ignored. For example, \ztangent{(curve)}{(4,y)} is equivalent to \ztangentat{(curve)}{(4)} for any y.

Everything else is the same as in \ztangentat.
25.3 Slope lines and normal lines

25.3.1 \tzslopeat

\tzslopeat draws a slope line to a path with a specified length. The mandatory arguments are \{<path>\}, \{<x>\}, and \{<length>\}. The tangent point is the middle point of the slope line.

By default, the slope lines are drawn on the behind layer. You can change the layer, like \settztztslopelayer{main}.

Remark: The slope is approximate and you can manipulate the slope by changing the variation interval with the option \{<epsilon1>,<epsilon2>\}. The default is \((\varepsilon_1,\varepsilon_2) = (0.01, 0.01)\). You can also use \settzslopeepsilon{<epsilon1>}{<epsilon2>} before the macro \tzslopeat, which is valid until the end of the tikzpicture environment, unless changed again.
Normal lines  With the option \texttt{[<rotate>]} immediately after the last mandatory argument \texttt{[<length>]}, you can draw a normal line to a curve by rotating the slope lines 90° or -90°.
25.3.2 \texttt{\textbackslash tslopeat'}

The \emph{swap version} \texttt{\textbackslash tslopeat'} draws a slope line with the opposite direction of \texttt{\textbackslash tslopeat}.

\begin{tikzpicture}
\helplines*(5,4)
\coors*(0,0)(A)(2,2)(B){B}(3.5,1)(C)(5,3)(D);
\tos[thick]"AA"(A)[out=0,in=180]
  (B)[out=0,in=180]
  (C)[out=0,in=255]
  (D);
\tslopeat'[-,red,thick]{AA}{1}{3cm} XX
\tslopeat'[-,blue]{AA}{2}{2cm}
\tslopeat'[-]{AA}{4}{2cm} {$f'(4)$}[r] XX
\end{tikzpicture}

\textbf{Remark:} \texttt{\textbackslash tslopeat\{AA\}{2}{2cm}[-180]} is equivalent to \texttt{\textbackslash tslopeat'\{AA\}{2}{2cm}}.

\begin{itemize}
\item Without the arrows, \texttt{\textbackslash tslopeat} and \texttt{\textbackslash tslopeat'} give the same result except where the labels are placed.
\item The swap version \texttt{\textbackslash tslopeat'} does not change its direction with the option \texttt{[<rotate>]}.
\end{itemize}

25.3.3 \texttt{\textbackslash tslope}

\texttt{\textbackslash tslope} is the same as \texttt{\textbackslash tslopeat}, except for one thing. \texttt{\textbackslash tslope} uses a coordinate instead of a value of $x$ to draw slope lines. So the mandatory arguments of \texttt{\textbackslash tslope} is \texttt{\{<path>, (<coor>), and \{<length>\}}. The $y$ value of \texttt{\{<x,y>\}} is ignored.

\begin{tikzpicture}
\helplines*(5,4)
\coors*(0,0)(A)(2,2)(B){B}(3.5,1)(C)(5,3)(D);
\tos[thick]"AA"(A)[out=0,in=180]
  (B)[out=0,in=180]
  (C)[out=0,in=255]
  (D);
\slope[red]{AA}(1,0){3cm}
\getxyval(B){\Bx}{\By}
\slope[blue]{AA}({\Bx},2cm)
\slope[blue]{AA}({\Bx},0){cm} f'(4)[r]\slope[red]{AA}(4,0){2cm} {$f'(4)$}[r]
\normal[blue]{AA}(1,0){1cm} [90]
\normal[red]{AA}(2,0){1cm} [90]
\normal[red]{AA}(4,0){1cm} [-90]{normal}{b}
\end{tikzpicture}
\begin{tikzpicture}
\helplines (1,1)(5,4)
\coors (0,1)(A)(2,3)(B)(4,2)(C);
\parabola "AA"(A)(B)(C)
\foreach \x in {1,1.5,...,3.5}
{ \slope[thick,red]{AA}(\x,0){2cm} }
\end{tikzpicture}

You can add TikZ code with the last option \texttt{<code.append>}.

\begin{tikzpicture}[scale=.5]
\helplines (9,8)
\axes (9,8)
\plotcurve "AA"(1,7)(3,3)(8,1);
\coors (2,0)(K2)(4,0)(K4);
\slope[red,tzextend={1cm}{2cm}]{AA}(K2){1cm} %%
\tangent[blue]{AA}(K4)[1:7]{tangent}[red,b]
\slope[->]{AA}(K4){0.1pt}[90]<--(\[turn\]0:2cm) node \[r\] {gradient} >
\vXpoint*{AA}(K4)
\ticksx{1,7}
\end{tikzpicture}

25.3.4 \texttt{\textbackslash tslope'}

The swap version \texttt{\textbackslash tslope'} draws a slope line with the opposite direction of \texttt{\textbackslash tslope}.

\begin{tikzpicture}
\helplines (5,4)
\coors (0,0)(A)(2,2)(B){B}[b](3.5,1)(C)(5,3)(D);
\tos[thick] "AA"(A)[out=0,in=180]
(B)[out=0,in=180]
(C)[out=0,in=255]
(D);
\slope'[->,red,thick]{AA}(1,0){3cm}
\slope'[->,blue]{AA}(B){2cm}
\slope'[->]{AA}(4,0){2cm}{$f'(4)$}[r]
\end{tikzpicture}

Remark: \texttt{\tslope{AA}{2}{2cm}[-180]} is equivalent to \texttt{\tslope'{AA}{2}{2cm}}.

- Without the arrows, \texttt{\tslope} and \texttt{\tslope'} give the same result except where the labels are placed.
- The swap version \texttt{\tslope} and \texttt{\tslope'} does not change its direction with the option \texttt{[<rotate>]}.

25.4 Normal lines

25.4.1 \texttt{\textbackslash tznormalat}

\texttt{\tznormalat} draws a normal line \textit{from a point} on a graph at \( x = \text{x-val} \). The mandatory arguments are \{\texttt{\textbar path\}\}, \{\texttt{x-val}\}, and \{\texttt{length}\}.

By default, the normal lines are drawn on the \texttt{behind} layer. You can change the layer, like \texttt{\settztnormalayer{main}}.
Remark: The slope of a normal perpendicular to the slope line is approximate and can be changed by changing the variation interval with the option \(<\epsilon_1, \epsilon_2>\). The default is \((\epsilon_1, \epsilon_2) = (0.01, 0.01)\). You can also use \texttt{\settznormalepsilon{<\epsilon_1>}{<\epsilon_2>}} before the macro \texttt{\tznormalat}, which is valid until the end of the \texttt{tikzpicture} environment, unless changed again.

\begin{tikzpicture}
\helplines*(5,4)
\coors*(0,0)(A)(2,2)(B)[b](3.5,1)(C)(5,3)(D);
\tos[thick]"AA"(A)[out=0,in=180]
(B)[out=0,in=180]
(C)[out=0,in=255]
(D);
\slopeat[red,thick]{AA}{[3cm]}
\slopeat[blue]{AA}{[2cm]}
\slopeat[blue]{AA}{[2cm]}
\normalat[->,red,thick]{AA}{[3cm]}
\normalat[->,blue]{AA}{[2cm]}
\normalat[->,blue]{AA}{[2cm]}
\end{tikzpicture}

You can change its direction with the option \texttt{[\textit{rotate}]} after the argument \texttt{[length]}.

\begin{tikzpicture}
\helplines*(5,4)
\coors*(0,0)(A)(2,2)(B)[b](3.5,1)(C)(5,3)(D);
\tos[thick]"AA"(A)[out=0,in=180]
(B)[out=0,in=180]
(C)[out=0,in=255]
(D);
\slopeat[red]{AA}{[2cm]}
\normalat[->,red,thick]{AA}{[2cm]}
\normalat[->,blue]{AA}{[2cm]}
\normalat[->,blue]{AA}{[2cm]}
\end{tikzpicture}

25.4.2 \texttt{\tznormalat'}

The \textit{swap version} \texttt{\tznormalat'} draws a normal line with the opposite direction of \texttt{\tznormalat}.

\begin{tikzpicture}
\helplines*(5,4)
\coors*(0,0)(A)(2,2)(B)[b](3.5,1)(C)(5,3)(D);
\tos[thick]"AA"(A)[out=0,in=180]
(B)[out=0,in=180]
(C)[out=0,in=255]
(D);
\slopeat[red]{AA}{[3cm]}
\normalat[->,blue]{AA}{[2cm]}
\normalat[->,blue]{AA}{[2cm]}
\end{tikzpicture}
% defaults
[]{<m>}{<m>}{.01,.01}{<m>}{[-90]}{}</m>

% \tznormalat'  (swap version)
\begin{tikzpicture}
\tzhelplines*(5,4)
\tzcoors*(0,0)(A)(2,2)(B){B}[b](3.5,1)(C)(5,3)(D);
\tztos [thick] "AA"(A)[out=0,in=180]
(B)[out=0,in=180]
(C)[out=0,in=255]
(D);
\tzslopeat[red,thick]{AA}{1}{3cm}
\tzslopeat[blue]{AA}{2}{2cm}
\tzslopeat{AA}{4}{2cm}{$f'(4)$}[r]
\tznormalat'[-,red,thick]{AA}{1}{3cm} \ XX
\tznormalat'[-,blue]{AA}{2}{2cm}
\tznormalat'[-]{AA}{4}{2cm}{normal}[r] \ XX
\end{tikzpicture}

Remark: The swap version \texttt{\tznormalat'} has a fixed \texttt{[rotate]} value of [-90].

- \texttt{\tznormalat{AA}{1}{3cm}[-90]} is equivalent to \texttt{\tznormalat'\{AA\}{1}{3cm}}.

- \texttt{\tznomalat'} does not change its direction with the option \texttt{[rotate]}.

25.4.3 \texttt{\tznormalat*(')}

% syntax: minimum
\tznormalat*[<path>]{<x-val>}{<length>}
% syntax: full
\tznormalat*[<opt>]{<path>}{<x-val>}{<epsilon1>,<epsilon2>}{<length>}[<rotate>]
{<text>}[<node opt>]<code.append>
% defaults
* []{<m>}{<m>}{.01,.01}{<m>}{[90]}{}</m>
* ' []{<m>}{<m>}{.01,.01}{<m>}{[-90]}{}</m>

The starred version \texttt{\tznormalat*} works like \texttt{\tzslopeat}, but rotated [90]. And the swap version \texttt{\tznormalat*'} works like \texttt{\tzslopeat}, but rotated [-90].

Remark: The swap version \texttt{\tznormalat*'} has a fixed \texttt{[rotate]} value of [-90].

- \texttt{\tznormalat*\{AA\}{1}{3cm}[-90]} is equivalent to \texttt{\tznormalat*'\{AA\}{1}{3cm}}.

- \texttt{\tznomalat*'} does not change its direction with the option \texttt{[rotate]}.
\begin{tikzpicture}
\helplines*(5,4)
\coors*(0,0)(A)(2,2)(B)(3.5,1)(C)(5,3)(D);
\tostos[thick]"AA"(A)[out=0,in=180]
(B)[out=0,in=180]
(C)[out=0,in=255]
(D);
\slopeat[red,thick]{AA}{1}{3cm}
\slopeat[blue]{AA}{2}{2cm}
\slopeat{AA}{4}{2cm}{$f'(4)$}[r]
\normalat[->,red,thick]{AA}{1}{3cm}
\normalat[->,blue]{AA}{2}{2cm}
\normalat'[->]{AA}{4}{2cm}{normal}[r]
\end{tikzpicture}

You can change its direction with the option \texttt{[<rotate>]} after the argument \texttt{<length>}.

% syntax: minimum
% \normal\{<path>\}(<x,y>)\{<length>\}
% syntax: full
% \normal[<opt>]{<path>}(<x,y>)\{<epsilon1>,<epsilon2>\}{<length>}[<angle>]
% \texttt{\{<text>\}[<pos,opt>]<code.append>}
% defaults
% \{<m>\}(<m>)(.01,.01){<m>}[90]{}
%}

% \tznormal\{<path>\}(<x,y>){<length>}
% \tznormal[<opt>]{<path>}(<x,y>)\{<epsilon1>,<epsilon2>\}{<length>}[<angle>]
% \texttt{\{<text>\}[<pos,opt>]<code.append>}
% defaults
% \{<m>\}(<m>)(.01,.01){<m>}[90]{}
%}

25.4.4 \texttt{\tznormal}

\tznormal is the same as \texttt{\tznormalat}, except for one thing. \tznormal uses a coordinate instead of a value of \(x\) to draw normal lines. So the mandatory arguments of \tznormal are \{<path>\}, \{<coor>\}, and \{<length>\}. The \(y\) value of \((x,y)\) is ignored.
25.4.5 \texttt{\textbackslash tznornal'}

The swap version \texttt{\textbackslash tznornal'} draws a normal line with the opposite direction of \texttt{\textbackslash tznornal}.

\begin{tikzpicture}
\helplines*(5,4)
\coors*(0,0)(A)(2,2)(B){B}(3.5,1)(C)(5,3)(D);
\tos[thick]"AA"(A)[out=0,in=180]
  (B)[out=0,in=180]
  (C)[out=0,in=255]
  (D);
\slope[red]{AA}(B){3cm}
\normal[->,blue]{AA}(B){2cm}
\normal[->,blue]{AA}(B){2cm}[45\textdegree][ar]
\normal[->,blue]{AA}(B){2cm}[135]
\end{tikzpicture}

Remark: The swap version \texttt{\textbackslash tznornal'} has a fixed [\texttt{\textbf{\textbackslash rotate}}] value of [-90].

- \texttt{\textbackslash tznornal}{AA}{1}{3cm}[-90] is equivalent to \texttt{\textbackslash tznornal'}{AA}{1}{3cm}.
- \texttt{\textbackslash tznornal'} does not change its direction with the option [\texttt{\textbf{\textbackslash rotate}}].

\begin{tikzpicture}[\rotate=90]
\helplines*(5,4)
\coors*(0,0)(A)(2,2)(B){B}(3.5,1)(C)(5,3)(D);
\tos[thick]"AA"(A)[out=0,in=180]
  (B)[out=0,in=180]
  (C)[out=0,in=255]
  (D);
\slope[red]{AA}(1,0){3cm}
\getxyval(B){Bx}{By}
\slope[blue]{AA}{Bx}{2cm}
\slope[AA](4,0){2cm}{$f'(4)$}[r]
\normal[->,blue]{AA}{1,0}{2cm}
\normal[->,red]{AA}{B}{2cm}
\normal[->,red]{AA}{4,0}{2cm}{normal}[b]
\end{tikzpicture}

25.4.6 \texttt{\textbackslash tznornal*(')}

The starred version \texttt{\textbackslash tznornal*(')} works like \texttt{\textbackslash tzslope}, but rotated [90]. And its swap version \texttt{\textbackslash tznornal*(')} works like \texttt{\textbackslash tzslope}, but rotated [-90].

188
Remark: The swap version `$\text{\texttt{\textbackslash tznornal*'}}$` has a fixed `[\texttt{\textbackslash rotate}]` value of $[-90]$.

- `$\text{\texttt{\textbackslash tznornal*}}\{\text{AA}\}\{1\}\{3\text{cm}\}\{-90\}$` is equivalent to `$\text{\texttt{\textbackslash tznornal*'}}\{\text{AA}\}\{1\}\{3\text{cm}\}$`.
- `$\text{\texttt{\textbackslash tznornal*'}}$` does not change its direction with the option `[\texttt{\textbackslash rotate}]$. 

```latex
\begin{tikzpicture}
    \zhelplines*(5,4)
    \tzcoors*(0,0)(A)(2,2)(B){B}[b](3.5,1)(C)(5,3)(D);
    \tztos[thick]"AA"(A)[out=0,in=180]
    (B)[out=0,in=180]
    (C)[out=0,in=255]
    (D);
    \tzslopeat[red,thick]{AA}\{1\}\{3\text{cm}\}
    \tzslopeat[blue]{AA}\{2\text{cm}\}
    \tzslopeat(\text{\texttt{\textbackslash tznornal*'}})\{4\text{cm}\}\{f'(4)\}[r]
    \tznornal*[->,red,thick]{AA}\{1,0\}\{3\text{cm}\}
    \tznornal*[->,blue]{AA}\{B\}\{2\text{cm}\}
    \tznornal*'->[AA]\{4,0\}\{2\text{cm}\}\{\text{normal}\}[r]
\end{tikzpicture}
```
26 Miscellany

26.1 Middle arrows

Four styles for middle arrow tips are predefined: -->--, --o--, --x-- and --/--.

26.1.1 Controllable middle arrow tips: -->-- and \settzmidarrow

The middle arrow tip style -->-- prints a middle arrow tip of stealth by default. It accepts one argument that changes the position (0.5 by default) of a middle arrow tip, like -->--=.75.

\begin{tikzpicture}
% middle arrow tip: -->--
\tzhelplines(4,3)
\tzline[-->--](0,3)(4,3)
\tzline[-->--=.75,blue,thick](0,2.5)(4,2.5)
\tzto[-->--,->,out=45](0,0)(4,1)
\end{tikzpicture}

How to control middle arrow tips The default options for middle arrow tips are -, thin, solid, and shorten >=0, and bend right=0. The defaults can be changed by \settzmidarrow.

The macro \settzmidarrow controls the position, style, and other options of middle arrow tips. The effect is valid until the end of the tikzpicture environment, unless changed again.

% syntax:
\settzmidarrow<position>{<arrow tip style>}[<opt>]
% all arguments are optional
% defaults:
<0.5>{stealth}[-,thin,solid,shorten <=0,shorten >=0,bend right=0]

\begin{tikzpicture}
% various arrow tip styles: arrows.meta
\tzhelplines(4,3)
\settzmidarrow{Circle[open]}
\settzmidarrow{>>>}
\settzmidarrow<.25>{(Rays)}[blue,scale=2]
\tikzset{>=to}
\settzmidarrow[stealth]
\end{tikzpicture}

You can use various styles of arrow tips. (See Ti\textit{k}Z manual on arrows.meta library, for more details.)
26.1.2 Fixed middle arrow tip styles: --o--, --x--, --/--

Three styles for middle arrow tips are predefined: circle --o--, cross --x--, and diagonal --/-- middle arrow tips. These middle arrow tip styles are fixed and cannot be changed. \settzmidarrow can only be used with \[<opt>\] to control these middle arrow tips. The other options <position> and {<arrow tip style>} options are ignored for these styles.

Circle middle arrow tips: The circle middle arrow tip style --o-- takes one argument to change the position (0.5 by default).

The circle is drawn like this:

\begin{tikzpicture}
% circle middle arrow tip: --o--
\tzhelplines(4,3)
\tzline|--o--(0,3)(4,3)
\tzline|--o--=.75,blue,thick(0,2.5)(4,2.5)
\tzto|--o--,->,out=45(0,0)(4,1)
\end{tikzpicture}

\begin{tikzpicture}
% \settzmidarrow: difference
\tzhelplines(4,3)
\settzmidarrow<.25>{Circle[open,fill=green][scale=2]}
\tzline|--o--,thick(0,3)(4,3)
\tzline|--o--=.5,blue,thick(0,2.5)(4,2.5)
\tzline|--o--,blue,thick(0,2)(4,2)
\settzmidarrow[scale=2,thick,fill=green]
\tzto|--o--,->,out=45(0,0)(4,1)
\end{tikzpicture}

Cross middle arrow tips: The cross middle arrow tip style --x-- takes one argument to change the position (0.5 by default).

The cross mark is drawn like this:

\begin{tikzpicture}
% cross middle arrow tip: --x--
\tzhelplines(4,3)
\tzline|--x--(0,3)(4,3)
\tzline|--x--=.75,blue,thick(0,2.5)(4,2.5)
\tzto|--x--,->,out=45(0,0)(4,1)
\end{tikzpicture}
Diagonal middle arrow tips  The diagonal middle arrow tip style --/-- takes one argument to change the position (0.5 by default).

The diagonal line is drawn like this:

```
\draw [option] (2pt,2pt) to (-2pt,-2pt);
```

\texttt{\textbackslash settzmidarrow} controls the diagonal arrow tip line with [option].

26.2 \texttt{\textbackslash tzbrace(\textquoteleft\textquoteleft)}

\texttt{\textbackslash 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\>)\{<text>\}[<node opt>]
% defaults
[]\{5pt\}[amplitude=5pt]\<>(\<m\>)\{\}

The \texttt{raise} value of a brace is 5pt by default and the value can be changed by the first curly brace optional argument \{<raise>\}.

The \texttt{amplitude} of a brace is 5pt by default. You can control the amplitude by writing the option \texttt{amplitude=<dim>} in the second bracket option [\texttt{<decoration opt>}].

\begin{tikzpicture}[sloped]
\zhelplines(4,2)
\tzline(0,0)(3,1)
\tzbrace[very thick](0,0)(3,1){AAA}[a=15pt]
\tzbrace'(red,very thick)(0,0)(3,1){BBB}[b=20pt]
\end{tikzpicture}

You can change the style of the decorating brace by the second bracket optional argument \{\texttt{<decoration opt>}\}.

The color of the calligraphic brace can be changed by the option \texttt{pen colour} in the list of \{\texttt{<draw option>}\}.

\begin{tikzpicture}[sloped]
\zhelplines(4,3)
\tzline(0,0)(3,1)
\tzbrace[very thick,pen colour=blue]
[amplitude=10pt]
(0,0)(3,1){AAA}[a=15pt]
\tzbrace'(red,very thick)(0,0)(3,1){BBB}[b=20pt]
\end{tikzpicture}

You can also move a brace by specifying the option \texttt{<shift coor>} immediately before the first mandatory coordinate. The empty shift option \texttt{<>} is not allowed.

The \texttt{swap version} \texttt{\tzbrace'} swaps the coordinates. So it prints a mirror image of \texttt{\tzbrace}. For example, \texttt{\tzbrace'}(0,0)(3,1) is equivalent to \texttt{\tzbrace}(3,1)(0,0).

\begin{tikzpicture}[sloped]
\zhelplines(4,2)
\tzline(0,0)(3,1)
\tzbrace[very thick](0,0)(3,1){AAA}[a=15pt]
\tzbrace'(red,very thick)(0,0)(3,1){BBB}[b=20pt]
\end{tikzpicture}
26.3 \texttt{\textbackslash tzsnake(+)}: Snake lines (Experimental)

\texttt{\textbackslash tzsnake} connects two points with a snaked line, with many default values, using Ti\kZ’s to operation.

\begin{verbatim}
\tzsnake(0,1)(3,2) \% works like
\draw [ decorate , decoration={ % many defaults
   snake, segment length=5pt, % controlled by \tzsnake
   amplitude=2.5pt, pre length=5pt, post length=5pt
} ] (0,1) to (3,2) ;
\end{verbatim}

\begin{verbatim}
\begin{tikzpicture}
\tzhelplines(4,3)
\tzsnake(0,1)(3,2)
\tzsnake[->,blue,bend right](1,0)(4,3)
\end{tikzpicture}
\end{verbatim}

% syntax: minimum
\tzsnake(<coor>)(<coor>)
% syntax: full
\tzsnake[<opt>]{<segment length>}[<decoration opt>][<shift coor>]
   (<coor>){<text>}[<node opt>]
   (<coor>){<text>}[<node opt>]<code.append>
% defaults
[]{5pt}[many defaults above]<{<>}{<>}\{<>}{<>}

The key segment length is controlled by the first curly brace option and you should write down all the other keys in the second bracket option [\texttt{<decoration opt>}] to change the values. Not specifying the option \texttt{\{<segment length>\}}, when the first bracket option is empty, you need empty brackets [], like \texttt{\textbackslash tzsnake[]} [\texttt{<decoration opt>}]....
The plus version \( \texttt{\texttt{t}z\texttt{sn}a\texttt{ke}+} \) uses the second mandatory coordinate as a relative coordinate to the first. Everything else is the same as in \( \texttt{t}z\texttt{sn}a\texttt{ke} \).
Version history

- v2.1 (2022/09/28)
  - Uploaded to CTAN
  - document done
- v2.1 (2022/09/24)
  - fixed bugs: \tzplot, \tzplotcurve, \tzslope
  - redesigned \tzaxes to have one path with “liftpen” for naming paths
  - redesigned \tzaxesL to have one path with “liftpen” for naming paths
  - redesigned \tzslopeat and \tzslope
  - added the swap versions \tzslopeat' and \tzslope'
  - added \tznormallat(*)('), \tznormalepsion and \settznormallayer
- v2.1 (2022/04/15)
  - modified \tzaxisx and \tzaxisy to add the option "<path name>"
  - added \tzdistance to calculate the distance between two coordinates
- v2.0 (2022/02/28)
  - Uploaded to CTAN
- v1.0.1 (2021/03/20) uploaded to CTAN
  - revised the document with typo corrections
  - added the option <code.append> to \tzframe, \tzcircle, and \tzellipse
  - added aliases: \let\tzrectangle\tzframe and \let\zoval\tzellipse
- v1.0 (2021/02/28) uploaded to CTAN

Acknowledgement

Many thanks to Kangsoo Kim from KTUG (Korean TeX Users Group) who wrote many packages, including oblivoir.cls, for helping to implement the semicolon versions and to handle \TeX expansion issues using expl3.

References

Index

【Symbols】
--/-- 192
-->-- 190
--o-- 191
-->x-- 191
<m> 40, 42, 46, 138

【A】
abbreviations 2, 12

【B】
bounding box 23, 40, 139

【C】
clockwise 124, 128
counterclockwise 124, 128
current bounding box 28, 157, 159
current subpath start 156

【E】
elliptical arc 125
everse curve 169
error message 4, 15, 44, 161, 163
even odd rule 117, 120, 121, 123

【H】
horizontal intersection 173, 174

【I】
intersection 170
inverse function 152, 156, 161, 162

【L】
label node 11, 47, 49, 51, 68

【M】
main node 11, 12, 47, 49, 51, 68
middle arrow tip 5, 190

【N】
node coordinate 67, 68
normal distribution 165
normal line 182

【P】
plus version 80, 84, 88, 91, 92, 96, 102, 105, 108
probability density function 165

【S】
secant 175, 177
semicolon version 4, 15, 44, 94, 99
\settzAAlinestyle 131
\settzAR radius 131
\settzanglelayer 131, 133-135
\settzanglemarklayer 135
\settzcdotradius 43, 45
\settzAAlinestyle 131
\settzfillcolor 75, 101, 135
\settzfilloccurrence 75, 101, 135
\settzfnarealayer 166, 167
\settzfnarealinestyle 167
\settzlinkstyle 91, 92, 95
\settzmarksize 58
\settzmidarrow 5, 190
\settznormalepsilson 185
\settzpathlayer 100
\settzpathstyle 100
\settzRAlinestyle 134
\settzRAsize 134
\settzsecantlayer 175, 177
\settzslopeepsilson 181
\settztangentepsilson 180
\settztangentlayer 178
\settztnormallayer 184
\settztsloplayer 181
string replacement 5
swap version 124, 126, 128, 129, 132, 133, 143, 152, 156, 161, 162, 183–185, 188, 193

【T】
tangent 178, 180
THREE WAYS 46
(tzA\zMid) 132
\tzanglemark' 132
\tzanglemark 131
\tzanglemark* 133
\tzanglemark* 133
\tzangleONE 131
\tzangleresult 131
\tzangleTWO 131
\tzarc' 124
\tzarc 124
\tzarcfrom' 126
\tzarcfrom 126
\tzarcsfrom 127
\tzaxes 23, 138
\tzaxes* 23, 139, 157
\tzaxesL' 143
\tzaxesL 143
\tzaxisx 24, 140
\tzaxisy 24, 140
\tzbox 41, 157
\tzbezier 10, 14, 103
\tzbezier* 105
\tzbezierAtBegin 105
\tzbezierAtEnd 105
\tzbox 116
\tzbox* 116
\tzbox** 116
\tzbox+ 116

197
\texttt{tzwedge}  129
\texttt{tzXpoint}  29, 170
\texttt{tzXpoint*}  30, 171

\texttt{U}
\texttt{useasboundingbox}  157

\texttt{V}
vertical intersection  171, 172