Abstract

This is a \LaTeX{} package that provides Ti\LaTeX{} based macros to make it easy to draw graphs. The macros provided in the tzplot package are just abbreviations for Ti\LaTeX{} 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 .................................................... 3
    2.1.1 New macros ............................................... 3
    2.1.2 Extending paths: \texttt{\textbackslash{}tz<...>AtBegin and \textbackslash{}<...>AtEnd} ................... 4
    2.1.3 New coordinates ........................................ 4
    2.1.4 Error messages ......................................... 4
    2.1.5 Abridged strings to place labels for coordinates, dots, and points ....................... 4
    2.1.6 New styles for middle arrow tips ........................ 5
  2.2 Remarks ..................................................... 5

Part II Getting Started

3 An Intuitive Introduction I: Basics
  3.1 Lines: Basics: \texttt{\textbackslash{}tzline(0,0)(3,1)} ........... 6
  3.2 Dots: Basics ................................................ 6
    3.2.1 A circle dot: \texttt{\textbackslash{}tzcdot(0,0)} ............. 6
    3.2.2 A circle node dot: \texttt{\textbackslash{}tdot(0,0)} ............. 7
    3.2.3 Difference between \texttt{\textbackslash{}tzcdot} and \texttt{\textbackslash{}tzdot} ............. 7
  3.3 Coordinates: Basics: \texttt{\textbackslash{}tzcoor(0,0)(A)} .............. 8
  3.4 Curves: Basics ............................................... 9
    3.4.1 \texttt{\textbackslash{}tzto(0,0)(4,2)} ................. 9
    3.4.2 \texttt{\textbackslash{}tzbezier} ............................... 9
### 3.4.3 tzparabola

#### 3.5 Adding text: Nodes and placement

- 3.5.1 \tznode\{3,1\}{text}[right]
- 3.5.2 Review: Main nodes and label nodes in TikZ
- 3.5.3 Abbreviations of TikZ basic placement option styles: a, r, ar, bl, etc.

### 3.6 Labeling dots and coordinates

- 3.6.1 \tzdot, \tzcdot
- 3.6.2 \tzcoor
- 3.6.3 \tzcoor*

### 3.7 Adding text next to lines or curves

- 3.7.1 \tzticks
- 3.7.2 \tzto
- 3.7.3 \tzbezier
- 3.7.4 \tzparabola

### 4 An Intuitive Introduction II: Repetition of Coordinates

#### 4.1 Linking many coordinates: Semicolon versions

- 4.1.1 \tzlines: Connected line segments
- 4.1.2 \tzonegon, \tzpolygon*: Closed paths
- 4.1.3 \tzonepath*: Filling area

#### 4.2 Many dots: Semicolon versions

- 4.2.1 \tzcdots(*)
- 4.2.2 \tzcdots(*)

#### 4.3 Many coordinates: Semicolon versions

- 4.3.1 \tzcoors, \tzcoors*
- 4.3.2 \tzcoorsquick
- 4.3.3 \tzcoorsquick*

#### 4.4 \texttt{plot coordinates}: Semicolon versions

- 4.4.1 \tzplot*: Mark dots with \texttt{[mark=*]}
- 4.4.2 \tzplot: Lines with \texttt{[tension=0]}
- 4.4.3 \tzplot*[\texttt{[draw]}]: Lines with dots
- 4.4.4 \tzplotcurve: Curves with \texttt{[smooth,tension=1]}

### 5 An Intuitive Introduction III: Plotting Functions

#### 5.1 Axes

- 5.1.1 \tzaxes
- 5.1.2 \tzaxes*
- 5.1.3 \tzshoworigin, \tzshoworigin*
- 5.1.4 \tzaxisx, \tzaxisy

#### 5.2 Ticks

- 5.2.1 \tzticks
- 5.2.2 \tzticks*
- 5.2.3 \tzticksx(*), \tzticksy(*)

#### 5.3 Projections on the axes

- 5.3.1 \tzprojx(*), \tzprojy(*)
- 5.3.2 \tzproj(*)

#### 5.4 Plot functions

- 5.4.1 \tzfn
- 5.4.2 \tzfnat, \tzfnt: Horizontal lines
- 5.4.3 \tzvfnat, \tzvfn: Vertical lines
- 5.4.4 \tzLFn: Linear functions

#### 5.5 Intersection points

- 5.5.1 Naming paths
- 5.5.2 \tzXpoint(*): Intersection points of two paths
- 5.5.3 \tzvXpointat(*), \tzvXpoint(*): Vertical intersection points
- 5.5.4 \tzhXpointat(*), \tzhXpoint(*): Horizontal intersection points

#### 5.6 Tangent lines and secant lines

- \texttt{[smooth,tension=1]}
6 Examples: Economics

6.1 Markets
  6.1.1 Market equilibrium: step by step
  6.1.2 Tax incidence: step by step

6.2 Firms
  6.2.1 Cost curves
  6.2.2 Equilibrium of a competitive firm
  6.2.3 Monopoly equilibrium

6.3 Consumers: Budget lines and indifference curves

6.4 Production Possibility Curves

6.5 Edgeworth box

6.6 Growth

6.7 Liquidity trap

6.8 Miscellany

Part III Points, Lines, and Curves

7 Getting Ready

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

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

7.3 \texttt{tzbox}: A bounding box

8 Dots

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

8.2 \texttt{tzdots(*)}: Multiple circle dots

8.3 \texttt{tzdot(*)}: A single node dot
  8.3.1 Three Ways to change the size of node dots
  8.3.2 How to label
  8.3.3 How to change colors and shapes
  8.3.4 How to move: \texttt{shift}
  8.3.5 Comparison: \texttt{tzdot} and \texttt{tzdot*}

8.4 \texttt{tzdots(*)}: Multiple node dots

9 Coordinates

9.1 \texttt{tzcoor} and \texttt{tzcoor*}
  9.1.1 \texttt{tzcoor}
  9.1.2 \texttt{tzcoor*}

9.2 \texttt{tzcoors} and \texttt{tzcoors*}: Semicolon versions
  9.2.1 \texttt{tzcoors}
  9.2.2 \texttt{tzcoors*}

9.3 \texttt{tzcoorsquick} and \texttt{tzcoorsquick*}: Semicolon versions
  9.3.1 \texttt{tzcoorsquick}
  9.3.2 \texttt{tzcoorsquick*}

9.4 \texttt{tzgetxyval}

10 Plot Coordinates: \texttt{tzplot}: Semicolon Versions

10.1 \texttt{tzplot} and \texttt{tzplot*}: Syntax

10.2 \texttt{tzplot*}: Dots and marks

10.3 \texttt{tzplot}: Lines

10.4 \texttt{tzplot}: Curves

10.5 \texttt{tzplot}: Bars and combs

10.6 \texttt{tzplotcurve(*)}
<table>
<thead>
<tr>
<th>11 Nodes</th>
<th>66</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.1 \tznode and \tznode*</td>
<td>66</td>
</tr>
<tr>
<td>11.2 \tznodes and \tznodes*: Semicolon versions</td>
<td>68</td>
</tr>
<tr>
<td>11.3 \tznodedefine(*)</td>
<td>70</td>
</tr>
<tr>
<td>11.4 \tznodedefine(*): Semicolon versions</td>
<td>71</td>
</tr>
<tr>
<td>11.5 \tznodeframe and \tznodeframe*</td>
<td>73</td>
</tr>
<tr>
<td>11.6 \tznodecircle and \tznodecircle*: Semicolon versions</td>
<td>74</td>
</tr>
<tr>
<td>11.7 \tznodeellipse and \tznodeellipse*</td>
<td>75</td>
</tr>
<tr>
<td>12 Lines</td>
<td>76</td>
</tr>
<tr>
<td>12.1 \tzline: Connecting two points</td>
<td>76</td>
</tr>
<tr>
<td>12.1.1 Line styles</td>
<td>76</td>
</tr>
<tr>
<td>12.1.2 Adding text</td>
<td>77</td>
</tr>
<tr>
<td>12.1.3 Moving lines: \texttt{shift}</td>
<td>78</td>
</tr>
<tr>
<td>12.1.4 Extending paths</td>
<td>78</td>
</tr>
<tr>
<td>12.1.5 Naming paths: Intersection points</td>
<td>78</td>
</tr>
<tr>
<td>12.2 \tzline+: Relative coordinates</td>
<td>79</td>
</tr>
<tr>
<td>12.3 Styles: \tzshorten and \tzextend</td>
<td>80</td>
</tr>
<tr>
<td>12.4 \tzlines: Connecting multiple points: Semicolon version</td>
<td>80</td>
</tr>
<tr>
<td>12.5 \tzlines+: Relative coordinates: Semicolon version</td>
<td>83</td>
</tr>
<tr>
<td>13 Connecting Points</td>
<td>85</td>
</tr>
<tr>
<td>13.1 \tzto: Two points</td>
<td>85</td>
</tr>
<tr>
<td>13.2 \tzto+: Relative coordinates</td>
<td>87</td>
</tr>
<tr>
<td>13.3 \tztos: Multiple points: Semicolon version</td>
<td>87</td>
</tr>
<tr>
<td>13.4 \tztos+: Relative coordinates: Semicolon version</td>
<td>90</td>
</tr>
<tr>
<td>13.5 \tzlink: Two points</td>
<td>90</td>
</tr>
<tr>
<td>13.6 \tzlinks: All in one: Semicolon versions</td>
<td>93</td>
</tr>
<tr>
<td>13.6.1 \tzlinks*: Standard version</td>
<td>93</td>
</tr>
<tr>
<td>13.6.2 \tzlinks*, \tzlink*, \tzlinks++: Variants</td>
<td>95</td>
</tr>
<tr>
<td>13.6.3 Putting text, shift, intersections, and extending paths</td>
<td>96</td>
</tr>
<tr>
<td>14 Filling Area</td>
<td>98</td>
</tr>
<tr>
<td>14.1 \tzpath: Semicolon version</td>
<td>98</td>
</tr>
<tr>
<td>14.2 \tzpath*: Semicolon version</td>
<td>100</td>
</tr>
<tr>
<td>14.3 \tzpath+ and \tzpath++: Relative coordinates: Semicolon versions</td>
<td>101</td>
</tr>
<tr>
<td>15 Curves</td>
<td>102</td>
</tr>
<tr>
<td>15.1 Bézier curves</td>
<td>102</td>
</tr>
<tr>
<td>15.1.1 \tzbezier</td>
<td>102</td>
</tr>
<tr>
<td>15.1.2 \tzbezier+: Relative coordinates</td>
<td>104</td>
</tr>
<tr>
<td>15.2 Parabolas</td>
<td>105</td>
</tr>
<tr>
<td>15.2.1 \tzparabola</td>
<td>105</td>
</tr>
<tr>
<td>15.2.2 \tzparabola+: Relative coordinates</td>
<td>107</td>
</tr>
<tr>
<td>15.3 Edges</td>
<td>108</td>
</tr>
<tr>
<td>15.3.1 \tzedge(*)</td>
<td>108</td>
</tr>
<tr>
<td>15.3.2 \tzedges(*): Semicolon versions</td>
<td>109</td>
</tr>
<tr>
<td>15.4 More curves</td>
<td>111</td>
</tr>
<tr>
<td>15.4.1 \tzplotcurve, \tzplot</td>
<td>111</td>
</tr>
<tr>
<td>15.4.2 \tzto, \tztos</td>
<td>111</td>
</tr>
<tr>
<td>15.4.3 \tzlink, \tzlinks</td>
<td>111</td>
</tr>
<tr>
<td>15.4.4 \tzfn</td>
<td>112</td>
</tr>
<tr>
<td>16 Polygons and Circles</td>
<td>113</td>
</tr>
<tr>
<td>16.1 Polygons: \tzpolygon: Semicolon versions</td>
<td>113</td>
</tr>
<tr>
<td>16.1.1 \tzpolygon</td>
<td>113</td>
</tr>
<tr>
<td>16.1.2 \tzpolygon*: Semicolon versions</td>
<td>114</td>
</tr>
</tbody>
</table>
16.1.3 \tzpolygon+, \tzpolygon++: Relative coordinates: Semicolon versions . 114
16.2 Rectangles .............................................. 115
  16.2.1 \tframe and its variants ................................. 115
  16.2.2 \trectangling(*) ..................................... 116
16.3 Circles and rings ........................................ 118
  16.3.1 \tcircle(*) ........................................... 118
  16.3.2 \tring(*) ............................................ 119
16.4 Ellipses ................................................. 121
  16.4.1 \tlellipse(*) ......................................... 121
  16.4.2 \ttellipse(*) ....................................... 122

17 Arcs, Wedges, and Angle Marks 123
  17.1 \tzarc('): Centered arcs .................................. 123
     17.1.1 Arcs ............................................... 123
     17.1.2 Elliptical arcs ...................................... 124
  17.2 \tzarcfrom('): Arcs as in Ti\textup{k}Z ..................... 125
  17.3 \tzarcsfrom: Connected arcs: Semicolon version ............. 126
  17.4 Wedges ................................................ 127
     17.4.1 \tzwedge(') ........................................... 127
     17.4.2 \tzwedge*(') ....................................... 128
  17.5 Angle marks ............................................. 129
     17.5.1 \tzpointangle: Angles between points ................... 129
     17.5.2 \tzanglemark('): Angle marks ......................... 130
     17.5.3 \tzanglemark*('): Fill angle marks ...................... 132
     17.5.4 \trightanglemark: Right angle marks .................... 133
     17.5.5 \trightanglemark*: Fill right angle marks ............... 134

Part IV Plotting Graphs 136

18 Axes 136
  18.1 \tzaxes .................................................. 136
     18.1.1 \tzaxes .............................................. 136
     18.1.2 \tzaxes* ............................................ 137
  18.2 \tzaxisx and \tzaxisy ..................................... 137
  18.3 Display the origin ....................................... 138
     18.3.1 \tzshoworigin ........................................ 138
     18.3.2 \tzshoworigin* ....................................... 138
  18.4 \tzaxesL('): L-type axes ................................ 139

19 Ticks 140
  19.1 \tzticks: Tick labels .................................... 140
  19.2 \tzticks*: Tick marks .................................... 142
  19.3 \tzticksx(*) and \tzticksy(*) ............................... 143

20 Projections 144
  20.1 \tzproj(*): Projections on the axes ....................... 144
  20.2 \tzprojax(*) and \tzprojy(*) ............................... 145
  20.3 \tzprojax(*)  ........................................... 146
  20.4 \tzprojsx(*) and \tzprojsys(*) ............................. 147

21 Plot Functions 148
  21.1 \tzfn and \tzfn': Plot functions and inverse functions .... 148
     21.1.1 Syntax ............................................. 148
     21.1.2 Inverse functions: \tzfn' ............................ 149
     21.1.3 Define and name functions ........................... 149
     21.1.4 Name paths: name path ................................ 150
21.1.5 Move graphs: shift ........................................... 150
21.1.6 Extend paths: <code>append</code>, \texttt{tzfnAtBegin}, \texttt{tzfnAtEnd} .......... 151
21.2 \texttt{tzfnofy} and \texttt{tzfnofy'}: Functions of variable \( y \) .................. 152
21.3 Horizontal lines ............................................... 153
\hspace*{1em} 21.3.1 \texttt{tzhfnat} ........................................... 153
\hspace*{1em} 21.3.2 \texttt{tzhfn} ............................................. 154
21.4 Vertical lines .................................................. 155
\hspace*{1em} 21.4.1 \texttt{tzvfnat} ........................................... 155
\hspace*{1em} 21.4.2 \texttt{tzvfn} ............................................. 156

22 Plot Linear Functions .............................. 156
\hspace*{1em} 22.1 \texttt{tLFn}: Plot linear functions .................................. 156
\hspace*{1em} 22.1.1 \texttt{tLFn} and \texttt{tLFn'} .................................. 156
\hspace*{1em} 22.1.2 \texttt{tLFnofy} and \texttt{tLFnofy'} .................................. 158
\hspace*{1em} 22.2 \texttt{tzdefLFn} ........................................... 159
\hspace*{1em} 22.3 \texttt{tzdefLFnofy} ........................................... 160

23 Some More Functions ............................................ 161
\hspace*{1em} 23.1 \texttt{tpdfN}(*), \texttt{tpdfZ}: Normal distributions .................. 161
\hspace*{1em} 23.2 \texttt{tzfnarea}(*): Fill under the graph ......................... 162
\hspace*{1em} 23.2.1 \texttt{tzfnarea}(*), \texttt{tzfnarealine}(') .............. 162
\hspace*{1em} 23.3 Envelope curves .......................................... 165
\hspace*{1em} 23.3.1 \texttt{tzfnmax}: Upper envelope curves ...................... 165
\hspace*{1em} 23.3.2 \texttt{tzfnmin}: Lower envelope curves ...................... 166

24 Intersections ....................................................... 167
\hspace*{1em} 24.1 \texttt{txXpoint}(*): Intersection points ....................... 167
\hspace*{1em} 24.2 Vertical intersection points .................................. 168
\hspace*{1em} 24.2.1 \texttt{tvXpointat}(*), \texttt{tvXpoint}(*) ...................... 168
\hspace*{1em} 24.3 Horizontal intersection points .................................. 169
\hspace*{1em} 24.3.1 \texttt{zhXpointat}(*), \texttt{zhXpoint}(*) ...................... 169
\hspace*{1em} 24.4 \texttt{tLFnXpoint}(*): Intersection point of linear functions .... 171

25 Secant and Tangent Lines .......................... 172
\hspace*{1em} 25.1 Secant lines ......................................... 172
\hspace*{1em} 25.1.1 \texttt{tssecantat} ........................................... 172
\hspace*{1em} 25.1.2 \texttt{tssecant} ............................................ 173
\hspace*{1em} 25.2 Tangent lines ........................................... 174
\hspace*{1em} 25.2.1 \texttt{tztangentat} ........................................... 174
\hspace*{1em} 25.2.2 \texttt{tztangent} ............................................ 176
\hspace*{1em} 25.3 Slope lines and normal lines .............................. 177
\hspace*{1em} 25.3.1 \texttt{tslopeat} ............................................ 177
\hspace*{1em} 25.3.2 \texttt{tslope} .............................................. 179

26 Miscellany .............................................................. 180
\hspace*{1em} 26.1 Middle arrows ........................................... 180
\hspace*{1em} 26.1.1 Controllable middle arrow tips: \texttt{--\textendash --} and \texttt{settzmidarrow} .... 180
\hspace*{1em} 26.1.2 Fixed middle arrow tip styles: \texttt{--o--}, \texttt{--x--}, \texttt{--/---} .......... 181
\hspace*{1em} 26.2 \texttt{tzbrace}(') ............................................ 182
\hspace*{1em} 26.3 \texttt{tzsnake}(*): Snake lines (Experimental) .................. 184

Version history ............................................. 186

Acknowledgement .................................................. 186
Part I
Preliminary

1 Introduction

1.1 About tzplot.sty

The tzplot package is just a collection of macros based on Ti\textit{k}Z to save you time typing Ti\textit{k}Z code.

What you can do with tzplot In \textit{pstricks}, a line connecting two points (A) and (B) is drawn by $\texttt{\textbackslash psline(A)(B)}$. With the package tzplot, you can do it by $\texttt{\textbackslash tzline(A)(B)}$.

\begin{verbatim}
\tzline(A)(B) % is an abbreviation of:
\draw (A) -- (B);
\end{verbatim}

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

Some macros in this package represent one or a few lines of code, but some represent dozens of lines of Ti\textit{k}Z code.

All of the drawing macros of tzplot are prefixed by \texttt{\textbackslash tz}. Of course, it means Ti\textit{k}Z. The syntax of the tzplot macros comes from \textit{tikz} and \textit{pstricks}. However, the input mode is more like \textit{pstricks}.

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

\begin{verbatim}
\usepackage{tzplot}
\end{verbatim}

The package depends on the packages \textit{tikz}, \textit{xparse}, and \textit{expl3}. And it uses the following Ti\textit{k}Z libraries:

\begin{verbatim}
arrows, backgrounds, calc, intersections, patterns, plotmarks, positioning, shapes, decorations.pathreplacing, calligraphy
\end{verbatim}

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

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

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 \texttt{\textbackslash tikzset{\textasciitilde=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\textit{k}Z tutorial. \textit{To make good use of this package, you need to familiarize yourself with Ti\textit{k}Z.}
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
- tznoded
- 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:

% preoccupied (alias) styles for main node options
\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:

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

Layers The tzplot package also defines graphic layers as follows:

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

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

\begin{tikzpicture}
  \tzhelplines(4,3)
  <tzplot macros>
  \begin{pgfonlayer}{background}
  \tzhelplines[blue](0,0)(3,1)
  \end{pgfonlayer}
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

2.1.1 New macros

New macros have been added:

- \tzfn': (swap version) inverse function of \tzfn
- \tzfnofy, \tzfnofy': function of \y
- \tzdeffn, \tzdeffi: to define functions
- \tzdefLFn, \tzdeflFnoy: define linear functions
- \tzLFn('), \tzLFnoy('): linear functions
- \tzLFnXpoint(*): linear function intersection points
- \tzfmin('), \tzfmax('): envelope curves
- \tzfnarea(*), \settzfnarealayer, \tzfnarealine('): to fill the area under graphs
- \settzfnarealinestyle
- \tzpdfZ, \tzpdfN(*): probability density functions of a normal distribution
- \tzprojs(*), \tzprojsx(*), \tzprojsy(*): multiple projection lines
- \settzpathlayer: main layer by default
- \tzslope, \tzslopeat
- \settzslope layer, \settzslopeat layer, \settzslopeepsilon
- \tzlink(+), \tzlinks(*)(+), \settzlinkstyle, \settzpathstyle
- \tzedge(+), \tzedges(+)
- \tznodes(*): multiple nodes
- \tnodedots(*): multiple node dots
- \tnoderectangle(*), \tnodebox(*), \tnodeoval(*): aliases
- \tzring(*), \tzcirclering(*)
- \tzellipsering(*), \tzovalring(*)
- \tzrectanglering(*), \tzframering(*), \tzboxring(*)
- \tzbox(*)
- \tzpointangle: angle between points
- \tzanglemark(*)(*), \tztightanglemark(*): angle marks
- \tzanglemark, \tzangleONE, \tzangleTWO: after \tzanglemark
- \settAAlinestyle, \settAAradius, \settzRSAsize.
- \settzmidarrow: to control middle arrow tips
- \tznake: snake lines
2.1.2 Extending paths: $\texttt{tz<...>AtBegin}$ and $\texttt{<...>AtEnd}$

- $\texttt{tztoAtBegin}, \texttt{tztoAtEnd}, \texttt{tztosAtBegin}, \texttt{tztosAtEnd}$ (version 1)
- $\texttt{tzlinkAtBegin}, \texttt{tzlinkAtEnd}, \texttt{tzlinesAtBegin}, \texttt{tzlinesAtEnd}$
- $\texttt{tzbezierAtBegin}, \texttt{tzbezierAtEnd}$
- $\texttt{tparabolaAtBegin}, \texttt{tparabolaAtEnd}$
- $\texttt{tzvfn(at)AtBegin}, \texttt{tzvfn(at)AtEnd}$ (version 1)
- $\texttt{zhfn(at)AtBegin}, \texttt{zhfn(at)AtEnd}$
- $\texttt{zfminAtBegin}, \texttt{zfminAtEnd}, \texttt{zfmaxAtBegin}, \texttt{zfmaxAtEnd}$
- $\texttt{tzplotAtBegin}, \texttt{tzplotAtEnd}, \texttt{tzplotcurveAtBegin}, \texttt{tzplotcurveAtEnd}$

2.1.3 New coordinates

- $\texttt{tzAAmid}$: angle arc midpoint, depending on $\texttt{tzanglemark}$
- $\texttt{tzRAvertex}$: right angle mark vertex, depending on $\texttt{tzrightanglemark}$

2.1.4 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*:

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

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

\begin{verbatim}
! Package tzplot Error: Perhaps you are trying an 'infinite slope' here or above!
\end{verbatim}

2.1.5 Abridged strings to place labels for coordinates, dots, and points

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

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

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

The macros related to this issue are as follows:

- *dots*: $\texttt{tzdot(s)}, \texttt{tzcdots(s)}, \texttt{znodedot(s)}$
- *coordinates*: $\texttt{tzcoor(s)}, \texttt{tzcoorsquick}$
- *intersection points*: $\texttt{tzXpoint}, \texttt{tzvXpoint}, \texttt{tzhXpoint}, \texttt{tzLFnXpoint}$
- *plot coordinates*: $\texttt{tzplot}, \texttt{tzplotcurve}$
- and their starred versions.
2.1.6 New styles for middle arrow tips

Some styles for drawing the middle arrow tips on a path are predefined.

- **-->--**: (controllable) middle arrow tip
  - \settzmidarrow controls the positions and styles of middle arrow tips
- **--o--**: the circle middle arrow tip
- **--x--**: the cross middle arrow tip
- **--/---**: the diagonal middle arrow tip

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

```latex
\tzlines[-->--,red](0,0)(1,0)(3,1); % default=0.5
\tzlines[-->--=0.7](0,0)(1,0)(3,3); % work like
\draw [-->--,red](0,0) -- (1,0) -- (3,1);
\draw [-->--=0.7](0,0) -- (1,0) -- (3,1);
```

2.2 Remarks

Some macros have been modified in order to add new features. This does not cause any harm to existing users.

- **\tznode**: to add new option \texttt{<node.code>}
  - This allows you to use full features (including \texttt{foreach}) of the TikZ’s node operation.
- **\tzframe, \tzcircle, \tzellipse**: to add new option \texttt{<code.append>}
  - Now you can use 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 \texttt{istgame} to draw game trees predefines the styles of the middle arrow tips including ->-, -o->, and -x-.

- The style ->- defined in \texttt{istgame} and -->-- defined in \texttt{tzplot} are a little different from each other in the default values.
  - Still you may want to use the style ->- instead of -->--. In that case, you can do like this:
    ```latex
    \tikzset{->-.style={-->--}}
    ```
  - \textit{However}, it is important to understand that changing the style to ->- may override the style of ->- defined in other package, depending on which package is loaded first.
  - The best way to use ->- instead of -->-- is to upload the \texttt{istgame} and follow the instruction of the manual.
- The styles --o-- and --x-- differ in definition and usage from the styles -o- and -x- of the \texttt{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 \TeX 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 \TeX 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[\textit{->},\textit{blue},\textit{thick}](0,3)(4,1)
\end{tikzpicture}

\texttt{\tzline[dashed](0,1)(3,2)} % 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’ *, with the \textit{radius} 1.2pt by default, at the point (0,0). The starred version \texttt{\tzcdot*} prints a filled dot *.

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

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

\tzcdot*[green](3,2)(3pt) \% is an abbreviation for:
\draw [fill,green] (3,2) circle (3pt);

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

3.2.2 A circle node dot: \tzdot(0,0)

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

% \tzdot: node dot
\begin{tikzpicture}
% \tzhelplines(4,3)
% \tzcdot(0,0)
% \tzcdot*[3,1)
% \tzcdot[red](0,1)(4mm) \% minimum size=4mm
% \tzcdot*[green](3,2)(6pt) \% minimum size=6pt
% \tzline(0,0)(3,1)
% \tzline(0,1)(3,2)
\end{tikzpicture}

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

% \tzcdot
\begin{tikzpicture}
% \tzhelplines(4,3)
% \tzcdot(0,0)(5pt)
% \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}

\tzcdot*[green](0,0)(6pt) \% works like:
\draw [green] (0,0) node \tzdot[minimum size=6pt] {}; "tzdot style is predefined"

3.2.3 Difference between \tzcdot and \tzdot

A ‘circle dot’ drawn by \tzcdot is affected by xscale or yscale in Ti\kern.5ex Z, but a ‘circle node dot’ drawn by \tzdot is not. Note also that \tzcdot controls the radius of a circle dot (following Ti\kern.5ex Z practice), while \tzdot controls the diameter of a circle node dot.
**Remark:** The circle dots drawn by \texttt{tzdot} 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{verbatim}
\begin{tikzpicture}[scale=.5, yscale=1.1]
  \helplines(4,3)
  \tzdot(0,0) \tzdot*(3,1)
  \tzdot[red](0,1)(4mm) \tzdot*[green](3,2)(6pt)
  \tline(0,0)(3,1)
  \tline(0,1)(3,2)
\end{tikzpicture}
\end{verbatim}

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.

\begin{verbatim}
\tzcoor(0,0)(A) \% works like:
  \path (0,0) coordinate (A);
  \% or
  \coordinate (A) at (0,0);
\end{verbatim}

\begin{verbatim}
\begin{tikzpicture}
  \helplines(4,3)
  \tzcoor(0,0)(A) \tzcoor(60:3cm)(B)
  \tline[->](A)(B)
  \tzdot(A)(5pt)
  \tzcoor(0,1)(C) \tzcoor(4,2)(D)
  \tline[dashed](C)(D)
\end{tikzpicture}
\end{verbatim}

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

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

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

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

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

\texttt{tzto[bend right]}(0,1)(3,2) \% works like:
\texttt{\begin{tikzpicture}\helplines(4,3)\tzto[bend right](0,1)(3,2)\end{tikzpicture}}

3.4.2 \texttt{tzbezier}

\texttt{tzbezier} draws a Bézier curve with \textit{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[\rightarrow,,tzshowcontrols](0,2)(1,3)(3,0)(4,1)
\end{tikzpicture}

\texttt{tzbezier[blue]}(0,0)(2,0)(4,2) \% works like:
\texttt{\begin{tikzpicture}\helplines(4,3)\tzbezier[blue](0,0)(2,0)(4,2)\end{tikzpicture}}

\texttt{tzbezier}(0,2)(1,3)(3,0)(4,1) \% works like:
\texttt{\begin{tikzpicture}\helplines(4,3)\tzbezier(0,2)(1,3)(3,0)(4,1)\end{tikzpicture}}

3.4.3 \texttt{tzparabola}

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

\texttt{tzparabola}(0,0)(2,4) \% works like:
\texttt{\begin{tikzpicture}\helplines(4,3)\tzparabola(0,0)(2,4)\end{tikzpicture}}

\texttt{tzparabola}(2,0)(3,3)(4,1) \% works like:
\texttt{\begin{tikzpicture}\helplines(4,3)\tzparabola(2,0)(3,3)(4,1)\end{tikzpicture}}
3.5 Adding text: Nodes and placement

3.5.1 `\tznod{(3,1)}{\text}[right]`

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

\begin{tikzpicture}
\helplines(4,4)
\parabola(0,0)(2,4)
\parabola[bend at end,dashed](0,0)(2,4)
\parabola(2,0)(3,3)(4,1)
\parabola[bend pos=.33,dashed](1,0)(4,4)
\end{tikzpicture}

\begin{tikzpicture}
\helplines(4,3)
\node(3,1)at(0,0){text}[right]
\tznod*(1,1){text}
\tdot*(2,2)
\tznod(2,2){Text}[above right,blue]
\end{tikzpicture}

\begin{tikzpicture}
\helplines(4,3)
\tznod(0,0){A}
\tznod*(1,1){text}
\tdot*(2,2)
\tznod(2,2){Text}[above right,blue]
\end{tikzpicture}

3.5.2 Review: Main nodes and label nodes in TikZ

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

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

\begin{tikzpicture}
\helplines(4,3)
\tznod*(2,1){main node}
\tznod*(3,3){A}[label=above:a,\label=0:r]
\end{tikzpicture}

\begin{tikzpicture}
\helplines(4,3)
\tznod(0,0){A}
\tznod(2,2){Text}[above right,blue]
\draw(2,2)node[above right,blue]{Text};
\end{tikzpicture}

Instead of angles, you can use the corresponding placement words. In TikZ, for example, `above` is replaced by 90 degree, `right` by 0 degree, `below left` by -135 degree, and the like. Note that the angle expression cannot be used for placing main nodes.

Remark:
- In this package, macros related to ‘dots’ or ‘coordinates’ (like `\tzcdot`, `\tzdot`, and `\tzcoor`) can optionally have ‘label nodes,’ while macros related to ‘lines’ and ‘curves’ (like `\tzline`, `\tzto`, `\tztbezier` and `\tzparabola`) optionally have ‘main nodes’ or ‘text nodes.’
• There is one exception: `\tzshoworigin`. `\tzshoworigin` can have a label, but its location is controlled by positional words such as `below left` but not by `<angle>`. (See Section 18.3 on page 138 for more details.)

### 3.5.3 Abbreviations of Ti\kZ basic placement option styles: \texttt{a}, \texttt{r}, \texttt{ar}, \texttt{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 \textit{cannot} use angels to place main nodes.)

\begin{tikzpicture}[font=\ttfamily\textbf,text=blue]
\tzhelplines[thick](2,2)
\tznode(1,1){\texttt{centered}}[draw]
\tznode(1,2){\texttt{a}}[a]
\tznode(0,1){\texttt{b}}[b]
\tznode(0,2){\texttt{b}l}[bl]
\tznode(2,1){\texttt{r}}[r]
\tznode(0,0){\texttt{bl}}[bl]
\tznode(2,2){\texttt{ar}}[ar]
\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, \textit{right after a coordinate}, \{\texttt{<label>}\} followed by \{\texttt{<angle>}\} (90 degree or above by default in Ti\kZ).

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

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

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

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

\begin{tikzpicture}
\tzcoor(1,2){C}[180](2pt) \texttt{\% works like:}
\tzcoor(1,2){C}[180](2pt)
\end{tikzpicture}

\texttt{\% syntax: simplified}
\texttt{\% defaults}
\texttt{\% \texttt{<m>} means 'mandatory'}

You can see the full syntax of \texttt{tzcoor} in Section 9.1 on page 51.
3.6.3 \texttt{\textbackslash{tzcoor}}

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

\begin{verbatim}
% syntax: simplified
\texttt{\textbackslash{tzcoor}}[<dot opt>](<coor>){<label>}{<angle>}{<dot size>}
\end{verbatim}

\begin{tikzpicture}
\helplines(4,3)
\tzcoor*(1,2)(C){180}(4pt)
\end{tikzpicture}

3.7 Adding text next to lines or curves

3.7.1 \texttt{\textbackslash{tzline}}

\texttt{\textbackslash{tzline}} accepts two mandatory coordinates. To add text to a line segment, just specify the optional arguments \texttt{\{<text>\}} and \texttt{\{<node opt>\} in-between the two coordinates. The \texttt{\{<node opt>\}} is \texttt{\{above, midway\}}, by default.

\begin{tikzpicture}
\helplines(4,3)
\tzline(0,0){route A}{3,0}
\tzline[dashed](1,1){route B}{4,1}
\tzline[blue]
  (0,3) {plan C} [below,near end] (3,3)
\end{tikzpicture}
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 `tzto`, just specify the optional arguments `{<text>}` and `[<node opt>]` *in-between* the two coordinates. By default, the `[<node opt>]` is `[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}

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}

`tzbezier` accepts three or four coordinates as arguments. You can add text to the curve drawn by `tzbezier` using the optional arguments `{<text>}` and `[<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 \texttt{tzparabola}

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

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

\textbf{Remark:} Without the semicolon \textbackslash;;, an error occurs with the the \textit{error message}:

! Package tzplot Error: You may have forgotten a semicolon here or above!

\begin{verbatim}
% syntax: minimal
\tzlines[<opt>](<coor>)(<coor>).repeated.(<coor>);
% syntax: simplified
\tzlines[<opt>]*"<path name>"
 (\{<coor>\}{<label>}[<angle>].repeated.(\{<coor>\}{<label>}[<angle>];
% defaults
[]"*(\{<m>\}[].repeated.())[];
% <m> means mandatory
\end{verbatim}

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

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

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 (){}[]..repeated..(){}[];.

4.1.2 \texttt{tzpolygon}, \texttt{tzpolygon*}: Closed paths

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

The starred version \texttt{tzpolygon*} does the same thing as \texttt{tzpolygon} except for one thing. \texttt{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 113 for more details.)

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

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

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

DO NOT FORGET to indicate when the repetition ends by typing a semicolon.
4.1.3 \texttt{tikzpath\textast}: Filling area

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

\begin{tikzpicture}
\tzhelplines(4,3)
\tzdashed\poly\(0,0\){up} (1,2){down} (3,0){up} (4,1){Weight}[r] ;
\end{tikzpicture}

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

\begin{tikzpicture}
\tzhelplines(4,3)
\tzdashed\poly\(1,3\)(0,2)(2,0)
(3,1)(4,2)(2,3); % semicolon
\end{tikzpicture}

\begin{tikzpicture}
\tzhelplines(4,3)
\tblue\line\(0,3\)(4,0)
\t\poly\pattern=crosshatch\(1,3\)(0,2)(2,0)(3,1)(4,2)(2,3);
\t\poly\pattern=bricks,\preaction=\fill=brown\(0,0\) (1,2) (3,0) (4,1);
\end{tikzpicture}

\begin{tikzpicture}
\tzhelplines(4,3)
\tblue\line\(0,3\)(4,0)
\t\poly\fill=green\(1,3\)(0,2)(2,0)(3,1)(4,2)(2,3);
\t\poly\fill\(0,0\) (1,2) (3,0) (4,1);
\end{tikzpicture}

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

% syntax: simplified
\texttt{tikzpath\ast}[\langleopt\rangle]\langle\langlecoor\rangle\rangle\langle\langlelabel\rangle\rangle\langle\langleangle\rangle\rangle..repeated..\langle\langlecoor\rangle\rangle\langle\langlelabel\rangle\rangle\langle\langleangle\rangle\rangle;
% defaults
*\langle\langlefill=black!50, \langle\langletext opacity=1\rangle\rangle\rangle..repeated..\langle\langlefill=\langle\langle\langletext opacity=1\rangle\rangle\rangle\rangle..repeated\..\langle\langle\langletext opacity=1\rangle\rangle\rangle\rangle;
% \langle\langlem\rangle\rangle means mandatory

The macros \texttt{tikzpath} and \texttt{tikzpath\ast} are much more flexible. See Section 14.1 on page 98 for more details.
How to change the fill opacity with `\tzpath*` is not discussed in this introduction, but one example is given below. (See Section 14.2 on page 100 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 `\tzcdots(*)`

`\tzcdots` accepts an arbitrary number of coordinates to print circle dots, but the coordinate repetition must be ended by `;` (semicolon version). `\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); % semicolon
\end{tikzpicture}

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

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

4.2.2 `\tzdots(*)`

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

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

% syntax: minimum
\tzdots*{<coor>}{<coor>}{repeated}{<coor>};
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 \(<dot\ size>\) after the semicolon.

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

\begin{tikzpicture}
\zhelplines(4,3)
\tdots(0,0)(1,1)(2,1)(3,2)(4,3);(10pt)
\tdots*[fill=red](0,3)(1,3)(2,3)(3,3)(4,2)(blue)[0];(6pt) \hspace{1cm} \textit{diameter}
\end{tikzpicture}

4.3 Many coordinates: Semicolon versions

4.3.1 \texttt{\textbackslash{}tzoors}, \texttt{\textbackslash{}tzcoors*}

\texttt{\textbackslash{}tzoors} accepts a pair of mandatory arguments in parentheses: \((<\text{coord}>)(<\text{name}>)\). The semicolon version macro \texttt{\textbackslash{}tzcoors} accepts an arbitrary number of pairs to define multiple coordinates. A semicolon ‘;’ is necessary to indicate when the repetition ends.

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

The options \{<label>\} and \[<angle>\] following each pair of \((<\text{coord}>)(<\text{name}>)\) allow you to put \(<label>\) in the direction of \(<angle>\) around the coordinate. Here, the repeating pattern is the quadruple \((<\text{coord}>)(<\text{name}>)(<label>)(<angle>)\). The first two parenthesis arguments are mandatory and others are optional. The pattern is repeated until \texttt{\textbackslash{}tzcoors} meets a semicolon ;.
The starred version \texttt{\textbackslash tzcoors*} does one more thing: to print node dots.

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

4.3.2 \texttt{\textbackslash tzcoorsquick}

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

\begin{tikzpicture}
\zhelplines(4,3)
\tzcoorsquick(0,0)(A)
  (1,1)(Best)
  (2,1)(Case)
  (3,3)(Done)
  (4,2)(End); % semicolon
\end{tikzpicture}

4.3.3 \texttt{\textbackslash tzcoorsquick*}

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

\begin{tikzpicture}
\zhelplines(4,3)
\tzcoorsquick*(0,0)(A)
  (1,1)(Best)
  (2,1)(Case)
  (3,3)(Done)
  (4,2)(End){END!}[[blue]0];
\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{TikZ}) of 2pt, which is the initial value in \texttt{TikZ}. Since this is a semicolon version, the repetition of coordinates must be ended by ;.

\begin{verbatim}
\begin{tikzpicture}
\tzhelplines(4,3)
\tzplot*(0,0)
(1,1)
(2,1)
(3,3)
(4,2); % semicolon
\end{tikzpicture}
\end{verbatim}

\texttt{\textbackslash{}tzplot*}(0,0)(1,1)(2,1); % works like:
\texttt{\textbackslash{}draw} [mark=*] plot coordinates {(0,0)(1,1)(2,1)};

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 \texttt{last} parenthesis argument \{\texttt{mark size}\} after the semicolon.

\begin{verbatim}
\begin{tikzpicture}
\tzhelplines(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}
\end{verbatim}

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

\texttt{\textbackslash{}tzplot}(0,0)(1,1)(2,1); % works like:
\texttt{\textbackslash{}draw} plot coordinates {(0,0)(1,1)(2,1)};
4.4.3 \texttt{\zplot*[draw]}: Lines with dots
\texttt{\zplot*[draw]} prints bullet marks at the specified coordinates and draws line segments connecting them. The repetition of coordinates must be ended by \texttt{;} (semicolon version).

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

4.4.4 \texttt{\zplotcurve}: Curves with \texttt{[smooth,tension=1]}
\texttt{\zplotcurve} 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 \texttt{;} (semicolon version).

\begin{tikzpicture}
\zhelplines(4,3)
\zplotcurve(0,0)
 (1,1)
 (2,1)
 (3,3)
 (4,2){END!}{[blue]0} ; % semicolon
\end{tikzpicture}

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

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

\begin{tikzpicture}
\zhelplines(4,3)
\zcoor(.5,.5)(A)
\zplotcurve[blue]{2}{(1,3)(A)(4,0)};
\zplotcurve[thick]{(1,3)(A)(4,0)}; % default: tension=1
\zplotcurve[red]{.55}{(1,3)(A)(4,0)}; % TikZ default
\zplotcurve[dashed]{(0)(1,3)(A)(4,0)};
\end{tikzpicture}
5 An Intuitive Introduction III: Plotting Functions

5.1 Axes

5.1.1 \texttt{tzaxes}

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

\texttt{tzaxes} draws the x-axis from \texttt{<x1>} to \texttt{<x2>} and the y-axis from \texttt{<y1>} to \texttt{<y2>}.

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

If \texttt{(x1,y1)} is omitted, it is regarded as \texttt{(0,0)}. And optionally the names of x-axis and y-axis can be printed at a specified place (by default, \texttt{[right]} for x-axis and \texttt{[above]} for y-axis).

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

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

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

5.1.2 \texttt{tzaxes*}

The starred version \texttt{tzaxes*} is just to set the current state to a bounding box when the \texttt{tzaxes} macro execution is completed. Use \texttt{tzaxes*} before any larger graphics.
\begin{tikzpicture}[scale=.5]
\tzaxes*(8,5)\{f(x)\}\% bounding box
\tzhelplines(-2,-1)(10,8)
\tzto[out=90,in=-135,dashed]\{(-2,8)(12,-2)\}
\tzbezier[blue]\{(-1,-1)(3,-2)(7,12)(10,10)\}
\end{tikzpicture}

5.1.3 \texttt{tzshoworigin, tzshoworigin*}

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

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

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

\begin{tikzpicture}[scale=.5]
\tzshoworigin*(<coor>){<text>}[<node opt>]
\end{tikzpicture}

\texttt{tzshoworigin*}(<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 <angle>. Instead, you can use the abbreviations of TikZ basic placement options such as a, 1, br, ect. See Section 1.2, for more details.)

\begin{tikzpicture}[scale=.5]
\tzshoworigin*\{O_1\}
\tzaxes(-1,-1)(8,5)
\tzaxes\{blue\}(7,4)(-1,-1)
\tzshoworigin*(7,4){O_2}[ar]
\end{tikzpicture}

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

5.1.4 \texttt{tzaxisx, tzaxisy}

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

\begin{tikzpicture}[scale=.5]
% syntax: simplified
\tzaxis\{<opt>\}<y-shift>{<x1>}{<x2>}{<text>}[<node opt>]
% defaults
\{ -, >=stealth\}<0>{<m>}{<m>}{<text>}[right]
\end{tikzpicture}

\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\{<x-tick places>}\{<y-tick places>\} prints tick labels for x- and y-axis at specified places, which are comma separated. By default, tick labels are the numbers specified.

You can change the numbered labels, for example \{2,4,7\}, to any other form, by doing like, for example, \{2/mylabel,4,7\}. (Internally, \zticks uses the \texttt{foreach} operation of Ti\textsc{KZ}.)

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 $\texttt{\textbackslash \texttt{tzticksx}}(\ast), \texttt{\textbackslash \texttt{tzticksy}}(\ast)$

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

\begin{tikzpicture}[scale=.5,font=\scriptsize]
\tzhelplines(8,5)
\tzaxes(-1,-1)(8,5)
\tzticksx(1/x,3/y,5/z,7)
\tzticksy(-5pt:10pt){1,2,4/k}
\end{tikzpicture}

$\texttt{\textbackslash \texttt{tzticksx}}(\ast)$ and $\texttt{\textbackslash \texttt{tzticksy}}(\ast)$ suppress tick labels for their corresponding axes, like $\texttt{\textbackslash \texttt{tzticks}}(\ast)$.

\begin{tikzpicture}[scale=.5,font=\scriptsize]
%\tzhelplines(8,5)
\tzaxes(-1,-1)(8,5)
\tzticksx*(1/x,3/y,5/z,7)
\tzticksy*(-5pt:10pt){1,2,4/k}
\end{tikzpicture}

You can see more details on $\texttt{\textbackslash \texttt{tzticks}}$ and its friends in Chapter 19 on page 140.

5.3 Projections on the axes

5.3.1 $\texttt{\textbackslash \texttt{tzprojx}}(\ast), \texttt{\textbackslash \texttt{tzprojy}}(\ast)$

$\texttt{\textbackslash \texttt{tzprojx}}$ 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.

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

\begin{tikzpicture}
\tzhelplines(4,3)
\tzaxes(4,3)
\tzdot*(3,2)\{($(x_1,x_2)$)[45]
\tzprojx(3,2)\{$x_1$}
\tzprojy[solid,->,draw=blue](3,2)\{$x_2$}[bl]
\end{tikzpicture}

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

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

5.4 Plot functions

5.4.1 \texttt{\textbackslash{tzfn}}

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

You can name a path formed by \texttt{\textbackslash{tzfn}} by specifying the optional argument \texttt{"<path name>"} right before the mandatory curly brace argument \texttt{\{<fn of \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 \{\text{Foo}\}, the macro name Foo (without the backslash) is automatically assigned to the name of the path. (See Section 21.1.4 on page 150, for more details.)

5.4.2 \tzhfnat, \tzfn: Horizontal lines

\tzfnat 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 [\text{<from:to>}].

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

5.4.3 \tzvfnat, \tzvfn: Vertical lines

\tzvfnat 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}{\text{Two}}[1,at start]
\tzcoors(0,2)(A)(0,3)(B);
\tzhfn[blue]{A}[\text{Three}][b,near end]
\tzhfn{-}[B][\text{Four}][r]
\end{tikzpicture}
5.4.4 \tzLFn: Linear functions
\tzLFn(<coor1>)(<coor2>)\ldots draws a linear function passing through two points: (<coor1>) and (<coor2>). \tzLFn(<coor1>{<slope>}\ldots 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 intersection points when two named paths intersect. The name of a path is usually given by the option [name path=<path name>] in Ti\kZ.

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

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

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

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

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

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

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

$E$

\begin{tikzpicture}[scale=.7,font=\footnotesize]
\tzhelplines(5,5)
\tzaxes(5,5)
\def\bgt{4-x}
\def\IC{3/y}
\tzfn"bgt"{\bgt}{0:4}
\tzfn[blue]"IC"{\IC}{.75:4}
\tzXpoint*(bgt)(IC)(E)(88)[45] % first intersection
\tzdot(E-2)(88)[45](5pt) % second intersection
\end{tikzpicture}

Remark: You have to expect Ti\kZ to take a few seconds (or less) to find intersection points.

5.5.3 \texttt{tzhXpointat(*), tzhXpoint(*): Vertical intersection points}

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

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

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

\begin{tikzpicture}[scale=.7,font=\scriptsize]
\tzhelplines(5,5)
\tzaxes(5,5)
\def\Fx{((x-1)^2}
\tzfn\Fx[0:3] % name path = Fx (automatically)
\tzXpointat(Fx)(2.5)(A)
\tzXpoint*(Fx)(2.8,1)(B) % y=1 is ignored
\tzhproj*[->]{(A)}(2.5)[88][2.56] % f(2.5)
\tzdot(B)(88)[0]
\end{tikzpicture}

5.5.4 \texttt{tzhXpointat(*), tzhXpoint(*): Horizontal intersection points}

\texttt{tzhXpointat\{<path>\}{<y>}(A)} works like \texttt{tzhXpoint\{<path>\}{<x>}(A)}, but it uses the
value of $y$ instead of $x$. The starred version \texttt{tzhXpoint*} additionally prints a node dot at the (first)
intersection point.
\texttt{\textbackslash tzXpoint} uses \texttt{(<coor>), while \texttt{\textbackslash tzXpointat} uses the value of $y$. Here the $x$ coordinate of \texttt{(<coor>)} is ignored. The starred version \texttt{\textbackslash tzXpoint*} prints a node dot at the (first) intersection point.

\begin{tikzpicture}[scale=.7,font=\scriptsize]
\def\Fx{((x-1)^2)}
\def\Fz{x^2+y^2}
\tzhelplines(5,5)
\tztangentat{\Fx}{2.5}(A)
\tztangentat{\Fz}{1,2.8}(B)
\end{tikzpicture}

You can see more details on \texttt{\textbackslash tzXpoint} and its friends in Chapter 24 on page 167.

5.6 Tangent lines and secant lines

5.6.1 \texttt{\textbackslash tztangentat}

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

Remark: The slope of a tangent line drawn by \texttt{\textbackslash tztangentat} is just approximate.

The line is drawn on the \texttt{\textbackslash 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]
\tzhelplines(5,5)
\tztangentat{\textcolor{red}{(x,y)}}{(0,1)}{(2,3)}{(4,2)}
\tztangentat{\textcolor{red}{(x,y)}}{(1)}
\tztangentat{\textcolor{red}{(x,y)}}{(0:3)}
\tztangentat{\textcolor{red}{(x,y)}}{[1:5]{tangent at $x=3$}}[b]
\end{tikzpicture}

5.6.2 \texttt{\textbackslash tztangent}

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

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

\begin{tikzpicture}[scale=.7]
\def\Fx{x^2+y^2}
\def\Fz{(x-1)^2}
\def\Gz{x^2+y^2}
\tztangentat{\Fx}{0,1}{(2,3)}{(4,2)}
\tztangentat{\Fz}{1}
\tztangentat{\Gz}{(0:3)}
\end{tikzpicture}
\begin{tikzpicture}[scale=.7]
\tzhelplines(5,5)
\tzparabola{thick,blue}"curve"(0,1)(2,3)(4,2)
\tzvXpoint*(\text{\texttt{curve}})(0,1)(A){A}{[135]}
\tzvXpoint*(\text{\texttt{curve}})(3,0)(B){B}{[45]}
\tztangent(\text{\texttt{curve}})(A){0:3}[][0:3]
\tztangent[red](\text{\texttt{curve}})(B){1:5}{tangent at \$B\$}[b]
\end{tikzpicture}

See Section 25.2 on page 174 for more details on \texttt{tztangent} and \texttt{tztangentat}.

5.6.3 \texttt{tztsecant}, \texttt{tzsecant}

\texttt{tztsecant{\texttt{\textless path\textgt}},{\texttt{\textless x1\textgt}},{\texttt{\textless x2\textgt}}} draws a secant line segment of {\texttt{\textless path\textgt}} from \texttt{\textless x1\textgt} to \texttt{\textless x2\textgt} on the \texttt{behind} layer, by default.

\begin{tikzpicture}[scale=.7]
\tzhelplines(5,5)
\tzaxes(5,5)
\tzparabola"curve"(0,1)(2,3)(5,1)
\tzsecant{\text{\texttt{curve}}}{1}{2}[][0:2]
\tzsecant[red]{\text{\texttt{curve}}}{1}{3}[][0:3]
\tzsecant[blue]{\text{\texttt{curve}}}{1}{4}[0:4]
\end{tikzpicture}

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

\texttt{tzsecant{\texttt{\textless path\textgt}},{\texttt{\textless coor1\textgt}},{\texttt{\textless coor2\textgt}}} draws a secant line segment of \texttt{\textless path\textgt} from the \texttt{x} coordinate of \texttt{\textless coor1\textgt} to the \texttt{x} coordinate of \texttt{\textless coor2\textgt}, ignoring \texttt{y} values of the coordinates, on the \texttt{behind} layer by default.

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

You can extend or shorten a secant line by specifying the domain \texttt{[<from:to>]}, which is an optional argument. If you specify the domain, \texttt{tzsecant} draws a secant line over the domain. You can also add some text next to the secant line by specifying the optional arguments \texttt{\{<text>\}} and \texttt{\[<node opt>\]}.

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

See Section 25.1 on page 172 for more details on \texttt{tzsecant} and \texttt{tztsecant}. 31
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]
\tzhelplines[step=.5cm](110,110)
\ttxaxes(110,110){$Q$}{$P$}
\ttxto[bend right=15]"dem"(0,100)(100,0){$D$}[a]
\ttxto[bend right=15]"supp"(0,10)(100,90){$S$}[ar]
\ttxXpoint*{dem}{supp}(eqm){$E$}
\ttxproj(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]
\tzhelplines[step=.5cm](110,110)
\ttxaxes(110,110){$Q$}{$P$}
\ttxto[bend right=15]"dem"(0,100)(100,0){$D$}[a]
\ttxto[bend right=15]"supp"(0,10)(100,90){$S$}[ar]
\ttxXpoint*{dem}{supp}(eqm){$E$}
\ttxproj(eqm){$Q^*$}{$P^*$}
\end{tikzpicture}

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

\begin{tikzpicture}[x=0.05cm,y=0.05cm,scale=.7]
\tzhelplines[step=.5cm](110,110)
\ttxaxes(110,110){$Q$}{$P$}
\ttxto[bend right=15]"dem"(0,100)(100,0){$D$}[a]
\ttxto[bend right=15]"supp"(0,10)(100,90){$S$}[ar]
\ttxXpoint*{dem}{supp}(eqm){$E$}
\ttxproj(eqm){$Q^*$}{$P^*$}
\end{tikzpicture}

Step 4  If necessary, use \txproj to draw projection lines with text around the projection points.
6.1.2 Tax incidence: step by step

Step 1: Determine the size of the graph.

Step 2: Define functions and plot them. And then, find an intersection point.

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

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

Step 5: Project 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
6.2.2 Equilibrium of a competitive firm

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

6.2.3 Monopoly equilibrium

\begin{tikzpicture}[scale=.5,font=\scriptsize]
%\tzhelplines(10,10)
\tzaxes(10,10){\(Q\)}{\(P\)}
\def\DD{8-\(\times\)}
\def\MR{8-2\times\x}
\def\MC{\x}
\tzfn\DD[0:8]{\(D\)}[ar]
\tzfn\MR[0:4.5]{\(MR\)}[r]
\tzfn\MC[0:7]{\(MC\)}[r]
\tzXpoint*[MR]{MC}(E)
\tzvXpoint*[DD](E)(EE)
% \(\pi^M\)
\tzproj(EE){\(Q^M\)}{\(P^M\)}
% \(Q^M\) and \(P^M\)
\tzplotcurve"ICC"(3,9)(A)(12,3);
% trial and error
\tzproj*[red][C.S.](E-|0,0)(E)(EE-|0,0);
\tznode(1,6){\(\text{SWL}\)}[l]
\tznode(1,2){\(\text{P.S.}\)}
% C.S. and P.S.
\end{tikzpicture}

6.3 Consumers: Budget lines and indifference curves

\begin{tikzpicture}[scale=.3,font=\scriptsize]
%\tzhelplines(15,12)
\tzaxes(15,12){\(x\)}{\(y\)}
\def\bgt{-3/4\times+9}\ % \(3x+4y=36\)
\tzfn\bgt[0:12]
\tzXpoint*[bgt]{MC}(E)
\tzvXpoint*[DD](E)(EE)
\tzproj(EE){\(Q^M\)}{\(P^M\)}
% \(Q^M\) and \(P^M\)
\tzXpoint*[DD]{MC}(C)
% with \(C\) and \(\text{SWL}\)
\tznode*[C.S.](E)[\tiny SWL][1]
\tzpath*[blue][0:8](EE)(EE-|0,0);
% \(\text{SWL}\)
\tznode(1,6){\(\text{C.S.}\)}
% C.S.
\tzpath*[pattern=horizontal lines](0,0)(E)(E-|0,0);
% \(\text{P.S.}\)
\tznode(1,2){\(\text{P.S.}\)}
% P.S.
\end{tikzpicture}
6.4 Production Possibility Curves

\begin{tikzpicture}[scale=.15,font=\scriptsize]
%\tzhelplines(35,25)
\tzhelplines(35,25){\$x\$}{\$y\$}
\def\bgt{-2/3*\x+20} % 2x+3y=60
\def\IC{150/\x} % u(x,y)=xy
\tzfn\bgt[0:30]
\tzfn\IC[7:30]
\tzcoor*(15,10)(E){\$E\$}[45]
\tzproj(E)
\tzticks{15,30}{10,20}
\tzvXpointat*[red]\IC{12}(A)[5:20]
\tztangent[blue,densely dashed]\IC(A)[0:8]
\end{tikzpicture}

6.5 Edgeworth box

\begin{tikzpicture}[scale=.5,font=\scriptsize]
%\tzhelplines(9,9)
\tzhelplines(9,7)
\tzaxes(9,8){\$X_1\$}{\$Y_1\$}
\tzaxes<8,6>(8,6)(-1,-1){\$X_2\$}{\$Y_2\$}
\tshoworigin{\$O_1\$}
\tshoworigin(8,6){\$O_2\$}[ar]
\ttto[out=-10,in=105]"PPC"(0,7)(6,0){PPC}[ar]
\tzvXpointat*[PPC]{(2)}(E)
\tzvXpointat*[tan]{(4)}(F)
\tztangent[blue]\PPC(E)[0:8]
\tzplotcurve[densely dashed,red](2,7)(F)(8,4);
\end{tikzpicture}
6.6 Growth

\begin{tikzpicture}[scale=0.5,font=\scriptsize]
%\helplines(10,8)
\tsp\begin{scope}
% bounding box
\tsp\edef\pathname{sFk} \edef\pathname{Fk} \edef\pathname{ndk}
\tsp\\( (E) = (E-2) \)
%\end{scope}
\tsp\\begin{scope}
%\end{scope}
\tsp\\end{tikzpicture}

6.7 Liquidity trap

\begin{tikzpicture}[scale=0.5,font=\scriptsize]
%\helplines(10,8)
\tsp\begin{scope}
\tsp\\edef\pathname{ndk}(0:9){\pathname{ndk}(0:9){(n-\delta)k}}[r] \path name=ndk
\tsp\end{scope}
\tsp\begin{scope}
%\end{scope}
\tsp\end{tikzpicture}

LM curves are drawn with two paths. To shift the IS curve, <shift coor> is used. See Section 22.1 on page 156 for more details.
The money demand \((m_d)\) curve is drawn with one path. To do this, \(<\text{code.append}>\) is used. See Section 13.1 on page 85 for more details.

6.8 Miscellany
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
  },
  \% tzdashed/.style={
    dashed=none,dash pattern=on 5mm/(#1) off 5mm/(#1),
    tzdashed/.default=10
  },
  \% 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>)
\% (<m>): mandatory argument
\end{verbatim}

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

\texttt{\tzhelplines(4,3)} \% works similarly to:
\begin{verbatim}
\draw [help lines] (0,0) grid (4,3);
\end{verbatim}

\texttt{\tzhelplines(1,1)(4,3)} \% works similarly to:
\begin{verbatim}
\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 \texttt{<n>} dots per 1cm. (That is, the default value is \texttt{tzdotted=10}.)
\begin{tikzpicture}
\thelplines[2,2]
draw [help lines] (3,0) grid (5,2);
\end{tikzpicture}

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

With the option value, \[\texttt{tzdotted=}\langle n\rangle/\langle d\rangle\], \texttt{thelplines} prints \langle n\rangle dots per \langle d\rangle cm. Similarly for \texttt{tzdashed}.

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

\begin{tikzpicture}
\helplines[tzdashed](4,3)
\helplines[step=.5](4,3)
\end{tikzpicture}

\begin{tikzpicture}
\helplines[tzdashed=10/.7](4,3)
\helplines[step=5,tzdotted=10/.7](4,3)
\end{tikzpicture}

\begin{tikzpicture}
\helplines[tzdashed=10/.7](4,3)
\helplines[step=5,tzdotted=10/.7](4,3)
\end{tikzpicture}

% syntax
\tzbbox((<coor1>)(<coor2>)
% defaults
(0,0)(<m>)
% <m>: mandatory
\tzbbox(-1,-1)(4,3)  \% is an abbreviation of:
\path [use as bounding box] (-1,-1) rectangle (4,3);

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

8.1 \tzcdot(*): A small circle

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

\begin{itemize}
\item Syntax: minimum
\begin{verbatim}
\tzcdot<>(<m>)
\end{verbatim}
\item Syntax: medium
\begin{verbatim}
\tzcdot*<>(<coor>){<label>}{[<angle>]}(<radius>)
\end{verbatim}
\item Syntax: full
\begin{verbatim}
\tzcdot*[<opt>]{<coor>}{[<label>]}{[<label opt>angle]}(<radius>)
\end{verbatim}
\end{itemize}

% defaults
*{solid,thin,tzcdot=1.2pt }<>(<m>)\{\}(1.2pt)
% \tzcdot is a predefined key (in this package).
% <m>: mandatory

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

How to change the size There are Three Ways to change the radius of a circle dot drawn by \tzcdot.

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

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

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

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

\begin{verbatim}
\tzcdot*(1,1) % works like:
\draw [fill] (1,1) circle [radius=1.2pt]; % default radius=1.2pt
\end{verbatim}

\begin{verbatim}
\tzcdot*[tzcdot=3pt] % works like:
\draw [fill] (0,0) circle [radius=3pt];
\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{verbatim}
\% \settzcdotradius
\begin{tikzpicture}
\tzhelplines(4,2)
\settzcdotradius(4pt)
\tzcdot(0,0)
\tzcdot(1,1)
\tzcdot(2,1)(2pt)
\tzcdot(3,0)
\end{tikzpicture}
\end{verbatim}

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

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

\begin{verbatim}
\% \tzcdot(): color, fill
\begin{tikzpicture}
\draw [help lines] (0,0) grid (4,3);
\tzcdot(0,0)(A)
\tzcdot*(red,fill=green)(2,1)(green)[[blue]0](10pt)
\tzcdot*[fill=red,fill=green](2,2)[big][center]
\end{tikzpicture}
\end{verbatim}

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

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

\begin{verbatim}
\% \tzcdot(*): color, fill
\begin{tikzpicture}
\draw [help lines] (0,0) grid (4,3);
\tzcdot(0,0)(A)
\tzcdot*[red,fill=green](2,1)[green][[blue]0](10pt)
\tzcdot*[fill=red,fill=green](2,2)[big][center]
\end{tikzpicture}
\end{verbatim}

Shift Dots can be shifted by specifying the optional argument \texttt{\<shift coor>\} immediately before \texttt{(\<coor>\)}. The \texttt{empty} shift option <> is \texttt{not allowed}. 42
8.2 \texttt{tzcdots(*): Multiple circle dots}

The macro \texttt{tzcdots} takes an \textit{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 \textit{semicolon \textbackslash;}\. Let us call this kind of macro a \textit{semicolon version} macro.

\textbf{Remark:}
\begin{itemize}
  \item DO NOT FORGET to enter \textbackslash; at the end of iteration.
  \item Without the semicolon \textbackslash;\, an error occurs with the the \textit{error message}:
\end{itemize}

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

The starred version \texttt{tzcdots*} prints multiple filled dots.

\begin{verbatim}
% \tzcdot: repeated
\begin{tikzpicture}
\tzhelplines(4,2)
\foreach \x in {0,...,4}
{ \foreach \A in {2,4,6,8}
  { \tzcdot[blue] \langle \x,0 \rangle (\A pt) } }
\end{tikzpicture}
\end{verbatim}

\begin{verbatim}
% \tzcdots(*)
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcdots(0,0)(1,1)(2,1)(3,2)(4,0);
\tzcdots*[0,3](1,2)(2,2)(3,3)(4,3); % semicolon
\end{tikzpicture}
\end{verbatim}

43
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 \(<\text{coor}>\{<\text{label}>\}[<\text{angle}>]\). In TikZ, \(<\text{angle}>\) can have the label option like \(<\text{[label opt]}\text{angle}>\).

How to change the size of dots  There are Three Ways of changing the radius of dots.
1. The simplest way is to use the last parenthesis optional argument, after the semicolon.
2. Another way is to use the tzcdot key, like \(\text{tzcdots[tzcdot=3pt]}\). If both options are used the key-value option wins.
3. You can also use the macro \texttt{settzcdotradius}. The effect remains within the tikzpicture environment unless it is changed again.

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 \(\text{tzcdots[text=red]}\) as shown in the following example.

Shift  You can move the coordinates of dots by specifying \texttt{<shift coor>} option immediately before the first coordinate.
The macro `\zdots` prints a small circle node `*`, as a dot, with the diameter (or minimum size) of 2.4pt, by default.

The starred version `\zdots*` prints a filled dot `*`.

\zdots(*) accepts one mandatory argument, denoted by `(m)`. All others are optional. The style `\zdots` is predefined in this package as follows:

\zdots(*) has three ways to change the size of node dots:

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

2. The simplest way is to use the last parenthesis optional argument, like `\zdots(0,0)(5pt)`, which yields the same result as in `\zdots[tzdot=5pt](0,0)`. If both options are used, the `\zdots` (or minimum size) option overwrites the last parenthesis option.
3. You can use the macro \settzdotsize to change the size (or diameter) of all node dots drawn by \tdot*. It is effective within the tikzpicture environment unless changed again.

```
\settzdotsize{8pt}
\tdot(0,0) \tdot*[8pt](1,1) \tdot*[4pt](2,1) \tdot*[6pt](3,0)
```

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 (\<coor>\). You can also change the label position by the option \[<angle>\] or \[<\[label opt\]angle>\] following \{<label>\}.

```
\tdot(0,0){A}(3pt) % works like:
\path (0,0) node [tdot=3pt,label={:A}] {}; \\
\tdot*[0](1,1)\[0\](8pt)
```

```
\tdot*[0,0](A)[0](8pt) % works like:
\path (0,0) node [fill,tdot=3pt,label={[red]:0:A}] {}; \\
\tdot*[0,0](A)[0](8pt)
```

```
\tdot*[0,0](A)[0](8pt) % works like:
\path (0,0) node [fill,tdot=3pt,label={[red]:0:A}] {}; \\
\tdot*[0,0](A)[0](8pt)
```

```
\tdot*[0,0](A)[0](8pt) % works like:
\path (0,0) node [fill,tdot=3pt,label={[red]:0:A}] {}; \\
\tdot*[0,0](A)[0](8pt)
```

```
\tdot*[0,0](A)[0](8pt) % works like:
\path (0,0) node [fill,tdot=3pt,label={[red]:0:A}] {}; \\
\tdot*[0,0](A)[0](8pt)
```

String replacement  (See also Section 2.1.5 on page 4.)

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

Unlike \tdot, the \tdot\’s label position depends on the size of a circle node. In TikZ jargon, \{<label>\} is in a \textit{label node} for a \textit{main node} that is a circle node with no text in it, so \<label>\ moves accordingly as the main node dot gets bigger or smaller.
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} [2*10pt]
\tzdot[tzdot=4*7pt] (3,2) {big} [center]
\tzdot[fill=red] (4,3) {D} [4*3pt]
\end{tikzpicture}

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

\begin{tikzpicture}
\begin{tikzpicture}
\tzhelplines (4,2)
\foreach \x in {0,...,4}
{ \foreach \A in {1,2,3,4}
{ \tzdot[blue] (<x,0> (2*\A mm) } }
\foreach \x in {0,...,4}
{ \foreach \A in {1,2,3,4}
{ \tzdot[green] <1,1> (x,0) (2*\A mm) } } % shift
\end{tikzpicture}
\end{tikzpicture}
8.3.5 Comparison: \texttt{\textbackslash{}tzdot} and \texttt{\textbackslash{}tzcdot}

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

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

The following table further shows the differences between them.

% concept | % single | % multi | % size control | % [key=default size]
node [circle] | \tzdot | \tzdots | \setztzdotsize | [tzdot=2.4pt] | % diameter
circle | \tzcdot | \tzcdots | \setztzcdotradius | [tzcdot=1.2pt] | % radius

Remark:
- In Ti\textit{kZ}, a ‘node’ is ‘not’ affected by ‘scaling’ unless the Ti\textit{kZ} option \texttt{transform shape} is used together. \texttt{\textbackslash{}tzdot} is also useful for labelling a large dot.
  - In \texttt{\textbackslash{}tzdot}, <\texttt{label}> is a label in a \textsl{label node} for a node dot (as a \textsl{main node}). So if a main node dot gets larger or smaller, its label moves accordingly. (Unlike, the labels with \texttt{\textbackslash{}tzcdot} or \texttt{\textbackslash{}tzcdots}.)
  - The position of <\texttt{label}> in \texttt{\textbackslash{}tzcdot} does not depend on the size of dots.
- The package \texttt{tzplot} takes \texttt{\textbackslash{}tzdot} as a \textit{standard dot}, not \texttt{\textbackslash{}tzcdot}. So, you can apply the \textsc{Three Ways} (on page 45) to change the size of any standard dots.

8.4 \texttt{\textbackslash{}tzdots(*)}: Multiple node dots

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

This is a \textit{semicolon version} macro, with the repeating pattern \texttt{(<coor>){<label>}[<angle>]}, which means that you need to type a \textit{semicolon ‘;’} at the end of the coordinate repetition. The \textit{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:
  
  ! Package \texttt{tzplot} Error: You may have forgotten a semicolon here or above!

The starred version \texttt{\textbackslash{}tzdots*} prints multiple filled node dots.

% syntax: minimum
\tzdots*{<coor>}{<coor>}..repeated..(<coor>);
% syntax: full
\tzdots*[<node opt>]{<shift coor>}{<coor>}{<label>}[<label opt>][<angle>]
\hspace{1em}..repeated.. ()[] ; (<dot size>)
% defaults
*[tzdot=2.4pt]< (m){[]} ..repeated. (){}[] ; (2.4pt)
How to label  Each coordinate can be followed by the optional arguments \{<label>\} and [<angle>] to label dots. So the triple \((<coor>){<label>}[<angle>]\) is the whole repeating pattern. (To avoid frequent coding errors, you can also use *string replacement* instead of angles. See also Section 2.1.5 on page 4

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 45.

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

2. Another way is to use the style \texttt{tzdot}, like \texttt{\tzdots[tzdot=3pt]...} If both options are used the \texttt{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}\[->
\zhelplines(4,3)
\settzdotsize{6pt}
\tzdots*[red]
(0,0)(1,1){\textbf{Ben}}[[blue]-90](2,1)(3,0);
\tzdots*[thick,blue,fill=green]
(1,2){A}(2,2){Ben}[[red]-90](3,2){C}(4,2){D};(8pt)
\tzdots*[blue]
(1,3){A}(2,3){B}(3,3){C}(4,3){D};\[0pt]
\end{tikzpicture}

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

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

% every label/.style
\begin{tikzpicture}[every label/.style=(draw,text=red)]
\zhelplines(4,3)
\tzdots*[red]
(0,0){Ace}[[font=\large\ttfamily]-90](2,1){\textbf{Bob}}[[blue]135]
(3,2){$C_1$\\textbackslash\textbackslash$N_0$}[[align=center]0];
\end{tikzpicture}

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

% \tzdots: shift
\begin{tikzpicture}
\zhelplines(4,3)
\tzdots[red]
(0,0){A}(1,1){(2,1){C}(3,2){D}[0](4,0){E};
\tikzset{every label/.style=(red)}
\tzdots*[<0,1>0,0){A}(1,1){(2,1){C}(3,2){D}[0](4,0){E};
\end{tikzpicture}
9 Coordinates

9.1 \texttt{tzcoor} and \texttt{tzcoor*}

9.1.1 \texttt{tzcoor}

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

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

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

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

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

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

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

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

\begin{verbatim}
% \texttt{\texttt{tzcoor:shift}}
\begin{tikzpicture}
\tzhelplines(4,3)
\texttt{\texttt{tzcoor}(0,0)(A){$A_1$} \%}
\texttt{\texttt{tzcoor}(2,1)(B){$B_2$} \texttt{\% ending point}}
\texttt{\texttt{\draw (A) -- (B);}}
\end{tikzpicture}
\end{verbatim}
9.1.2 \texttt{\textbackslash tzcoor*}

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

\begin{verbatim}
\texttt{\textbackslash tzcoor*}(<coor>)(<coor name>)
\texttt{\textbackslash tzcoor*}(<coor>)(<coor name>){<label>}[<angle>]
\texttt{\textbackslash tzcoor*}[<dot opt>]<shift coor>
  (<coor>)(<name>){<label>}[<angle>]{<dot size>}
\texttt{\textbackslash defaults}
  *[]<>(<m>)(<m>)[](2.4pt)
\end{verbatim}

\texttt{\textbackslash tzcoor*(0,0)(A)} \% works like:
\begin{verbatim}
\path (0,0) coordinate (A);
\texttt{\textbackslash tzdot*(0,0)}
\end{verbatim}

\texttt{\textbackslash tzcoor(0,0)(A)\{A\}[right]} \% works like:
\begin{verbatim}
\path (0,0) coordinate (A);
\texttt{\textbackslash tzdot*(0,0)\{A\}[right]}
\end{verbatim}

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 \texttt{node} option. To change the size of dots, you can apply the \textsc{Three Ways} (see Subsection 8.3.1 on page 45).

\begin{verbatim}
\texttt{\begin{tikzpicture}
\texttt{\textbackslash helplines(4,3)}
\texttt{\textbackslash tzcoor*(0,0)(A\{A\}_1)} \% TikZ default: 90 or above
\texttt{\textbackslash tzcoor*(30:3cm)(B\{B\}_2)}[draw,blue]0
\texttt{\draw (A) -- (B);}
\texttt{\end{tikzpicture}}
\end{verbatim}

Shift  The optional argument <shift coor> works just like in \texttt{\textbackslash tzcoor}. The \texttt{empty} shift option <> is \textit{not allowed}.  

52
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).

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

This is a *semicolon version* macro. The quadruple `<coor><name>{<label}[[angle]>` is the whole repeating pattern. It is required to type a *semicolon ‘;’* to indicate when the coordinate repetition ends.

```latex
\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);
```

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

You can add a label to each specified coordinate by adding the optional arguments `{<label>}` and `[<angle>]` immediately after `<name>`.

```latex
% \tzcoors
\begin{tikzpicture}
\tzcoors (0,0)(A){Ace}{[font=\textbf]-90}
(2,1)(B){\textsf{Bob}}{[blue]135}
(3,2)(C){$C_1$} [red]80\end{tikzpicture}
```

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

The starred version \tzcors* 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 \langle<coor>\rangle\langle<name>\rangle\langle<label\rangle\rangle\langle<angle\rangle\rangle. It is required to type a semicolon ‘;’ to indicate when the iteration of coordinates ends.

You can label each dot by specifying the optional arguments \texttt{\{\langle<label\rangle\}\}} and \texttt{\{\langle<angle\rangle\}\}} after the pair \texttt{\langle<coor>\rangle\langle<name>\rangle\}}.

You can change the dot color by \texttt{\{\langle<dot opt\rangle\}\}} and the label color by \texttt{\{\langle<label opt\rangle\}\}}. You can apply the \texttt{\textsc{Three Ways}} (on page 45) 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{\{\langle<dot size\rangle\}\}}.

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

9.3.1 \texttt{\textbackslash tzcorsquick}

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

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

9.3.2 \texttt{\textbackslash tzcorsquick*}

The starred version \texttt{\textbackslash tzcorsquick*} 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 45). 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 label 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*\(<\text{coor}>\)(<name>)..repeated..\(<\text{coor}>\)(<name>)

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

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

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

A label can be suppressed by the empty braces {}. You can change the dot size using the **Three Ways** (on page 45). You can shift the coordinate by specifying `<shift coor>` immediately before the first coordinate. The *empty* shift option <> is not allowed.

```
```
```
10 Plot Coordinates: \tzplot: Semicolon Versions

10.1 \tzplot and \tzplot*: Syntax

\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.

% syntax: minimum
\tzplot(<coor>)(<coor>)..repeated..(<coor>) ;
% syntax: medium
\tzplot(<coor>)\{<label>\}[<angle>]..repeated..(<coor>)\{<label>\}[<angle>] ;
% syntax: full
\tzplot[<opt>]{<tension>}[<plot opt>]\{shift coor"<path name>"  
   (coor)\{<label>\}[<label opt>angle]\  
   ..repeated.. \}();  ;<mark size> <code.append>
% defaults
  [tzmark=2pt][0][smooth] <>" (<>)[] ..repeated.. ()[] ; (2pt) <>  
  - [plot opt] is [smooth] by default  
  - It can be changed to [smooth cycle]

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 \Tikz. The value of \tzmarksize can be changed by the macro \settzmarssize, like \settzmarssize{3pt}.
10.2 \texttt{tzplot*}: Dots and marks

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

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

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

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

There are THREE WAYS to change the mark size.
1. The simplest way is to use the parenthesis optional argument \(<\text{mark size}>\), \textit{immediately after the semicolon}.  
2. You can use the style \texttt{tzmark}, like \texttt{tzmark=3pt}.  
3. You can also use the macro \texttt{settzmarksize}, which is effective until the end of \texttt{tikzpicture} environment.

\begin{verbatim}
% \texttt{tzplot*}: label, size
\begin{tikzpicture}[scale=.8]
\tzhelplines(4,3)
\tzplot*(0,0){A}[-135](1,1)(2,1)(3,3){D}[blue0]; (1mm)
\tzplot*[mark=o,tzmark=6pt](0,3)(1,3)(2,2)(3,1);
\settzmarksize{4pt}
\tzplot*[red](0,2){A}(1,2){B}(2,0){C}[blue0](3,2){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 2.1.5 on page 4 for more details.)

\begin{verbatim}
% \texttt{tzplot*}: label: strings instead of angles
\begin{tikzpicture}[scale=.8]
\tzhelplines(4,3)
\tzplot*(0,0){A}[bl](1,1)(2,1)(3,3){D}[blue0]; (1mm)
\tzplot*[mark=o,tzmark=6pt](0,3)(1,3)(2,2)(3,1);
\settzmarksize{4pt}
\tzplot*[red](0,2){A}(1,2){B}(2,0){C}[blue0](3,2){D};
\end{tikzpicture}
\end{verbatim}
With `\tzplot*`, you can draw line segments by giving the TikZ’s option `draw` in the first bracket optional argument, like `\tzplot*[draw].`

\begin{tikzpicture}[scale=1]
\tzhelplines(4,3)
\settztmarksize{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);
\tzplot*[draw,dashed,red,mark=heart](0,2)(1,2)(2,0)(3,2);
\end{tikzpicture}

Remark: In TikZ, the `mark` shapes are affected by `scale`, `xscale`, and `yscale`.

\begin{tikzpicture}[yscale=.5]
\tzhelplines(4,3)
\settztmarksize{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);
\tzplot*[draw,dashed,red,mark=heart](0,2)(1,2)(2,0)(3,2);
\end{tikzpicture}

Shift You can move specified coordinates using 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.

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

Extending path You can use `<code.append>` as the last optional argument after a semicolon.

\begin{tikzpicture}
\tzhelplines(4,3)
\tzplot*[draw](0,0)(1,1)(2,2){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</code> extends the path after \tzplotAtEnd.

\begin{tikzpicture}
\zhelplines(4,3)
\tzplotAtBegin{(0,0.2) to [bend right]}
\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.)

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

\begin{tikzpicture}
\zhelplines(4,3)
\tzplot(0,0)(1,1)(2,1)(3,3);
\tzplot[dashed,mark=o,smooth cycle](0,3)(1,3)(2,2)(3,1);
\tzplot[red,mark=*,mark options={blue}][smooth cycle](0,1)(1,2)(2,0)(3,2);
\end{tikzpicture}

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

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

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>].

\begin{tikzpicture}
\zhelplines(4,3)
\tzplot(0,0)(1,1)(2,1)(3,3);
\tzplot[dashed,mark=o,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 \texttt{\textbackslash 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 \texttt{\textbackslash tzplot}, from the last coordinate, you can write Ti\textbackslash kZ code in the very last optional argument `<code.append>`, after the semicolon.

You can also use \texttt{\textbackslash tzplotAtBegin} and \texttt{\textbackslash tzplotAtEnd} to extend the path of \texttt{\textbackslash tzplot} at the beginning and at end, respectively. Specifying `<code.append>` extends the path after \texttt{\textbackslash tzplotAtEnd}.
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 \{\texttt{<tension>}\} before the coordinates or between the first and second bracket options (if they exist).

\begin{verbatim}
\texttt{\texttt{tzplot}[blue,smooth cycle]{1}(0,0)(1,2)(2,2)(3,3); % works like:}
\texttt{\texttt{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 \texttt{tension=1} version of \texttt{tzplot}. (See Section 10.6 on page 63.)

10.5 \texttt{tzplot}: Bars and combs

With \texttt{tzplot}, you can draw bars or combs, using the Ti\textsc{k}Z 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 \tzplotcurve(*)

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

The starred version \tzplotcurve* draws a curve and displays marks *, by default. Basically, this is equivalent to \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} <>"" (<>{}[] ..repeated.. (){}[] ; (2pt) <>
\end{verbatim}

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

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

Since \tzplotcurve is a \textit{semicolon version}, you need to enter a semicolon to indicate when the coordinate iteration ends. In repeating coordinates, each mandatory coordinate can have a label. So the whole repeating pattern is the triple `(coor){{text}}{pos}`. For example, `(A){here}{above}` represents `(A) node [above] {here}` in TikZ.

Options: lines, labels, colors, smooth cycle Use the first bracket option to control the colors of lines or labels.

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

You can change the color of all labels together by adding `[text=<color>]` to the first bracket option list.
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 \texttt{tzplotcurve(*)} from the last coordinate, you can directly write Ti\textit{kZ} code in the very last optional argument \texttt{<code.append>}, after the semicolon.

You can also use \texttt{<--cycle>} to close the path \emph{with a straight line} from the last coordinate to the first coordinate.

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

name path for intersection points  You can name the path of \texttt{tzplotcurve} by specifying the option \texttt{"<path name>"} immediately before the first coordinate.
11 Nodes

11.1 \tznode and \tznode*

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

\begin{tikzpicture}
\tzhelplines(4,3)
\tzplotcurve[blue] "IC"(.5,3)(1.5,1.2)(4,.5){$u$}[0];
\tzplot[draw=red] "bgt"(0,3)(4,0){budget}[-90];
\end{tikzpicture}

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

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

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

\begin{itemize}
\item \texttt{\tznode*[draw] (0,0)(A){text}[above right]} \% works like:
\item \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 [<opt>] or the second bracket option [<node opt>] to put text with different colors, fonts, and so on.

Abbreviations  You can use abbreviations (or aliases) such as a for above, l for left, ar for above right, bl for below left, and so on to indicate where the text of a main node is placed. (See also Section 1.2 on page 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 (<coor>) by specifying (<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 <shift coor> immediately before the coordinate (<coor>). The empty shift option <> is not allowed.
Repetition: foreach The last optional argument \texttt{<node.code>} can be used to iterate over to place multiple nodes.

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

\begin{tikzpicture}
\helplines\node\(\x\),0\){\x}\foreach\x\in\{1,2,3\}\node\[fill=blue,text=yellow\]<1,2>(\x),0\){\x}\[circle,blue,fill=yellow\]
\node\foreach\x\in\{1,2,3\}\node\foreach\y\in\{1,2,3\} at \(\x\),0\){\x};
\end{tikzpicture}

11.2 \texttt{tznodes} and \texttt{tznodes*}: Semicolon versions

\texttt{tznodes} accepts any number of \textit{mandatory} pairs of coordinates and their names to defined node coordinates at specified coordinates.

\begin{tikzpicture}
\helplines\node\(\x\),0\){\x}\foreach\x\in\{1,2,3\}\node\[fill=blue,text=yellow\]<1,2>(\x),0\){\x}\[circle,blue,fill=yellow\]
\node\foreach\x\in\{1,2,3\}\node\foreach\y\in\{1,2,3\} at \(\x\),0\){\x};
\end{tikzpicture}

\begin{tikzpicture}
\helplines\node\(\x\),\y\){\(P_{\x\y}\)}\foreach\x\in\{1,2,3,4\}\node\foreach\y\in\{1,2,3\} at \(\x\),0\){\x};
\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. \texttt{\textbackslash tznodes} works similarly to \texttt{\textbackslash tzcoors}, but the former uses main nodes and the latter uses label nodes.

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

---

\begin{tikzpicture}
\helplines(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};
\tzlines(A)(B)(C)(D)(E);
\end{tikzpicture}

\begin{tikzpicture}
\helplines(4,4)
\tznodes*[rectangle,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];
\tzlines(A)(B)(C)(D)(E);
\end{tikzpicture}

---

% \texttt{\textbackslash tznodes}
\begin{tikzpicture}
\helplines(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};
\end{tikzpicture}

% syntax: minimum
\tznodes{<coor>}{<node name>}[ repeated.. ]{<coor>}{<node name>};
% syntax: medium
\tznodes{<coor>}{<node name>}{<text>}[ node opt ] [ repeated.. ]{<coor>}{<node name>};
% syntax: full
\tznodes{<every node opt>}{<shift coor>}{<node name>}{<text>}[ node opt ] [ repeated.. ]{<coor>}{<node name>};
% defaults
{}<>{<m>}{<m>}[ repeated.. ]{<coor>}{<node name>};
*{draw}{}<>{<m>}{<m>}[ repeated.. ]{<coor>}{<node name>};
11.3 \texttt{\textbackslash znodedot(\ast\ast)}

\texttt{\textbackslash znodedot} names a node and prints a circle node dot (of the size 2.4pt, by default). \texttt{\textbackslash znodedot} is basically the same as \texttt{\textbackslash zdot}, except for one thing. \texttt{\textbackslash znodedot} names a node.

The starred version \texttt{\textbackslash znodedot\ast\ast} prints a filled circle node dot (of the size 2.4pt, by default), just like \texttt{\textbackslash zdot\ast\ast}. But it optionally names a node.

Since \texttt{\textbackslash znodedot\ast\ast} prints a node dot, its \texttt{\textless label\textgreater} is placed by \texttt{\textless angle\textgreater}.

\texttt{\textbackslash znodedot\ast\ast}(1,1)(A)[Ace][180] \texttt{\textbackslash path (1,1) node (A) [\textbackslash zdot,fill,label=(180:Ace)] {} ;}

You can apply the Three Ways (on page 45) to change the size of node dots. The simplest way is to use the last parenthesis option \texttt{\textless dot size\textgreater).
You can move the coordinates of dots by specifying the `<shift coor>` option immediately before the coordinate. The empty shift option `<>` is not allowed.

You can use the abridges strings instead of angles. (See also Section 2.1.5 on page 4.)

11.4 \znodedots(*): Semicolon versions

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

The starred version \znodedots* prints filled node circle dots.
% \tznodenedots: label
\begin{tikzpicture}
\tzhelplines(4,3)
\tznodenedots(0,0)(A){A}
(1,1)(B)
(2,1)(C){C}[0]
(3,2)(D){D}[-90]
(4,0)(E){E}; % semicolon
\tzlines[red](A)(B)(C)(D)(E);
\tznodenedots*
(0,3)(A)(1,2)(B){B}[45]
(2,2)(C){C}[45](3,3)(D)(4,3)(E);
\tzlines(A)(B)(C)(D)(E);
\end{tikzpicture}

The simple first bracket option of \tznodenedots(*) controls dots rather than labels. If you want to control all the labels, you can use TikZ’s every label style as follows:

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

\textbf{Remark:} Comparison of connecting nodes and coordinates:

% connecting \tznodenedots
\begin{tikzpicture}
\tzhelplines(4,3)
\tznodenedots(0,0)(A){A}
(1,1)(B)
(2,1)(C){C}[red]-90
(3,2)(D){D}[draw,blue]a
(4,0)(E){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.

% 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){D}[draw,blue]a
(4,0)(E){E}; (6pt)
\tzlines[thick](A)(B)(C)(D)(E); % connects coordinates
\end{tikzpicture}
11.5  \texttt{\textbackslash tznodename} and \texttt{\textbackslash tznodename*}

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

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

% \texttt{\textbackslash tznodename}(2,1)(A){Here} % works like 
\texttt{\node [draw,rectangle] (A) at (2,1) [above] \{Here\};

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

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

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

\textbf{Remark:} \texttt{\textbackslash tznodename} works very similar to \texttt{\textbackslash tznod}{\textbackslash node}, but their ‘starred versions’ work differently. While \texttt{\textbackslash tznodename*} draws the perimeter of a node, \texttt{\textbackslash tznodename*} fills a node with color.
\tznoderectangle and \tznodobox are aliases of \tznodoframe. \tznoderectangle* and \tznodobox* are aliases of \tznodoframe*.

You can change the fill opacity of \tznodoframe* by specifying the last curly brace option \{<fill opacity>\}. You can also move the node by specifying the \<shift coor>\ option immediately before the coordinate. (The empty shift option <> is not allowed.)

\begin{tikzpicture}
\helplines
\framednode[0,0](A)
\framednode[4,2](B)[\circle]
\line[blue,thick](A)(B)
\framednode*[\green,scale=1.5](2,1)
{\node{Node frame}}
\pin[180:Left, pin=-45:pin]
\end{tikzpicture}

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

\begin{itemize}
\item The default fill opacity can be changed by the macro \settzfillopacity.
\item The default fill color is black!50. You can use the macro \settzfillcolor to change the default fill color.
\item With \tznodoframe, you can change the color of the perimeter and text with the second bracket option [\node opt].
\end{itemize}

11.6 \tznodocircle and \tznodocirc*  
\tznodocircle works just like \tznodoframe but with a circle node.

\begin{tikzpicture}
\helplines\[solid\]
\framednodebox(0,0)
\framednode*[fill=yellow](2,2)(A)
\framednode*[red](3,2)(C)
\node[draw=green]<-1,-2>(4,2)(D)
\to[->,bend left=45](A.north)(C.90)
\to[->,bend right=45,dashed](B.-135)(D.west)
\end{tikzpicture}

\begin{tikzpicture}
\helplines\[solid\]
\framednodebox*[\circle,\fill=\black!50,\fill opacity=.3,\text opacity=1]<>{\m}<{}[\text=\black]
\end{tikzpicture}

\begin{tikzpicture}
\helplines\[solid\]
\framednode*[\circle,\fill=\black!50,\fill opacity=.3,\text opacity=1]<>{\m}<{}[\text=\black]
\end{tikzpicture}

74
You can change the \texttt{fill opacity} of \texttt{\tznodcircle} by specifying the last curly brace option \{\texttt{<fill opacity>}\} or using \texttt{\settzfillopacity}. You can also move the node by specifying the \texttt{<shift coor>} option immediately before the coordinate. The \texttt{empty} shift coordinate option \texttt{<>} is \texttt{not allowed}.

11.7 \texttt{\tznodellipse} and \texttt{\tznodellipse*}

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

The starred version \texttt{\tznodellipse*} works just like \texttt{\tznodframe*} but with an ellipse node.
\begin{tikzpicture}
  \node[draw, circle, fill=red] (C) at (3,2) {Cat};
  \node[draw, fill=green] (D) at (4,2) {Dog};
\end{tikzpicture}

You can change the fill opacity of \texttt{\texttikz{\textnodeellipse*}} 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.

\section{Lines}

\subsection{\texttt{\texttikz{\textzline}}: Connecting two points}
\texttt{\textzline} connects two points with a straight line.

\subsection{Line styles}
You can use the first optional argument \texttt{\{opt\}} to control the line styles.
12.1.2 Adding text

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

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

% \tzline: text next to line
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline[blue,->](0,0)(3,3)
\tzline[<->](0,2)(3,0)
\tzline[dashed](1,2)(4,0)
\tdots*(1,2)(4,0);
\end{tikzpicture}

You can use the abbreviations (or aliases) of Ti\kern-.1667em kZ basic placement options such as a for above, bl for below left, and so on. (For more details, see page 2.)

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

% \tzline: text at or around last coordinate
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline(0,0){Above}(4,0) % default [above]
\tzline(0,1){Near end}[b,near end](4,1) % below
\tzline[-](0,2){Sloped}[sloped](3,3) % default [above]
\end{tikzpicture}
12.1.3 Moving lines: shift

You can move the line generated by \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}
\helplines(4,3)
\tzone[blue,->] (0,0)(3,3){A}[r]
\tzone[blue] <1,0>(0,0)(3,3){A}[r]
\tzone[dashed] (1,2)(4,0){C}[r]
\tzone[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 \tzone from the last coordinate, by writing Ti\textit{kZ} code in the last optional argument \texttt{<code.append>}.\n
\begin{tikzpicture}
\helplines(4,3)
\tzone[blue,->] (0,0)(3,3)<arc(0:-60:1cm)>
\tzone[blue] {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 \tzone at the beginning and at end, respectively. Specifying \texttt{<code.append>} extends the path after \texttt{\tzoneAtEnd}.

\begin{tikzpicture}
\helplines(4,3)
\tzone[blue,->] (0,0)(3,3)<arc(0:-60:1cm)>
\tzoneAtBegin{(2,3) -|}
\tzoneAtEnd{ arc (-90:90:.5) node [a] \{here?\} }
\tzone[blue] {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 \texttt{[<opt>]} . You can use this name of path to find intersection points.
12.2 \texttt{\textbackslash tzline+:} Relative coordinates

The \texttt{\textbackslash tzline+:} \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}

% \texttt{\textbackslash tzline+:} name path
\begin{tikzpicture}
\tzhelplines(4,3)
\texttt{\textbackslash tzline+[blue,->]} "dem"(0,0)(3,3){demand}[r]
\texttt{\textbackslash tzlineAtBegin}(0,1) to [bend left1] 
\texttt{\textbackslash tzline+[dashed,->]}"supp"(1,2)(3,-2){supply}[ar]
% intersection point\texttt{\textbackslash xpoint*}{dem}{supply}
\end{tikzpicture}
12.3 Styles: \texttt{tzshorten} and \texttt{tzextend}

The styles \texttt{tzshorten} and \texttt{tzextend} are defined as follows:

\begin{verbatim}
\% 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)}
\end{verbatim}

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

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

\begin{verbatim}
\% tzshorten, tzextend
\begin{tikzpicture}
  \zhelplines(4,3)
  \tzdots(0,0)(3,0);
  \tzline(0,0)(3,0)
  \tzdots(0,1)(3,1);
  \tzline[->,thick,blue,tzshorten={2mm}{1mm}](0,1)(3,1)
  \tzdots(1,2)(3,2);
  \tzline[<->,thick,red,tzextend={1cm}{1cm}](1,2)(3,2)
\end{tikzpicture}
\end{verbatim}

12.4 \texttt{\textbackslash tzlines}: Connecting multiple points: Semicolon version

\texttt{\textbackslash tzlines} connects two or more points with connected line segments. Since \texttt{\textbackslash tzlines} takes an arbitrary number of coordinates as arguments, it is a \textit{semicolon version}. So, you need to enter a semicolon ‘;’ to indicate when the coordinate iteration ends.

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

\texttt{\tzlines(1,1)(2,2)(3,1)(4,3)}; \texttt{\% works like:}

\texttt{\draw (1,1) -- (2,2) -- (3,1) -- (4,3)};

The whole repeating pattern in \texttt{\tzlines} is the triple \texttt{(coor){text}[node opt]}.  

80
Line styles  Use the first optional argument \(<\text{opt}>\) to control the style of the connected line drawn by \texttt{\tzlines}.

\begin{verbatim}
\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}
\end{verbatim}

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

\begin{verbatim}
\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}
\end{verbatim}

Shift  You can move the connected line by specifying \(<\text{shift coor}>\) before the first coordinate or immediately before the option "<\text{path name}>" if it exists. (The empty shift option \(<\text{}>\) is not allowed.)

\begin{verbatim}
\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}
\end{verbatim}

Naming paths  You can name the path of \texttt{\tzlines} by specifying the option "<\text{path name}>" immediately before the first coordinate.

81
Extending paths  You can extend the path of \texttt{\tzlines} by writing TikZ code in the \textit{last} (even \textit{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 plus version \texttt{tzlines+} connects two or more points with connected line segments, but each coordinate (except the first one) is relative to the previous coordinate.

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

\begin{verbatim}
\% syntax: minimum
\tzlines+ (<coor>) (<coor>) ..repeat.. (<coor>) ;
\% syntax: full
\tzlines+[<opt>]<shift coor>"<path name>" <coor>\{<text>\}<node opt> ..repeated.. (){}[] ; <code.append>
\% defaults
[]<>" "<>\{\} ..repeated.. (){}[] ; <>
\end{verbatim}

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

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

\begin{tikzpicture}
\tzhelplines*(4,3)
\tzlines+[red,thick,<->,text=blue]
  <1,1> (0,1){Up}
  (1,1){Down}[a,sloped]
  (2,-1){Up}
  (1,2){line A} ;
\tzcoors(0,0)(A)(1,1)(B)(1,0)(C)(1,-1)(D);
\tzlines+[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 \tzto: Two points

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

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

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

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

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

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.
% 	zto: adding text
\begin{tikzpicture}
\zhelplines(4,3)
\tzto[blue,thick,->](0,1){line}[sloped](3,3)
\tzto[out=0](0,0){curve}[pos=.8](4,2){line B}[r]
\tzto[bend right,dashed](2,0)(4,1){line C}[red,r]
\end{tikzpicture}

**Shift** You can move the line by specifying the option \texttt{<shift coor>} before the first coordinate or immediately before the option "\texttt{<path name>}" if it exists. (The empty shift option <> is not allowed.)

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

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

% 	zto: name path
\begin{tikzpicture}
\zhelplines(4,3)
\tzto[bend right=45,blue,thick,->]"curveA"(1,3)(4,0)
\tzto[out=0]"curveB"(0,0)(4,2){line B}[r]
\% intersection point
\tzxpoint*[{curveA}]{curveB}
\end{tikzpicture}

**Extending the path** You can extend the path of \texttt{\tzto} by writing TikZ code in the last optional argument \texttt{<code.append>}. You can also use \texttt{\tztoAtBegin} and \texttt{\tztoAtEnd} to extend a path of \texttt{\tzto} at the beginning and at the end, respectively. Specifying \texttt{<code.append>} extends the path after \texttt{\tztoAtEnd}. 

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

86
13.2 \texttt{\textbackslash tzto}: Relative coordinates

The \textit{plus version} \texttt{\textbackslash tzto +} uses the second coordinate relative to the first coordinate. 

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

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

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

\texttt{\textbackslash tztos} takes an arbitrary number of coordinates as arguments to connect them by the \textit{TikZ's to} operation. \texttt{\textbackslash tztos} is much more flexible than \texttt{\textbackslash tzlines}. Since this is a \textit{semicolon version}, you need to enter a \textit{semicolon} to indicate when the coordinate iteration ends.
% syntax
\tztos[<opt>]<shift coor>"<path name>"
  (<coor1>[<to opt>]<text>[<node opt>])..repeated..(){}{} ; <code.append>
% defaults
[]<>""(<m>)[]{};<>% works like:
\begin{tikzpicture}
\draw (0,0) to (1,2) to (3,1);
\end{tikzpicture}

The quadruple (<coor>)[<to opt>]{<text>}{<node opt>} is the whole repeating pattern. Here, [<to opt>] is for the options of TikZ’s to operation such as [bend right], [bend left], [bend left=<angle>], [out=<angle>,in=<angle>] and so on.

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

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

Adding text You can add text next to lines or curves by specifying the options {<text>} and [<node opt>] in-between coordinates or after the option [<to opt>], if it exists. You can also add text at or around the last coordinate by the last options {<text>} and [<node opt>].
% \tztos: adding text
\begin{tikzpicture}
\tzhelplines(4,3)
\tztos(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}

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 <> is not allowed.)

% \tztos: shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzdot(0,0)
\tztos[blue,->] (0,1)(1,2)(3,1)(4,3);
\tztos[red,thick] <1,1>(0,0)[bend right] (1,1) (2,1)[out=-135,in=45] (3,0);
\end{tikzpicture}

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

% \tztos: name path, intersection points
\begin{tikzpicture}
\tzhelplines(4,3)
\tzdot(0,0)
\tztos[blue,->]"AA"(0,1)(1,2)(3,1)(4,3);
\tztos[red,thick] <1,1>"BB"(0,0)[bend right] (1,1) (2,1)[out=-135,in=45] (3,0);
% intersection points
\tzXpoint*(AA){BB}(X)
\tzdot(X-2)(4pt)
\end{tikzpicture}

Extending paths You can extend the path of \tztos from the last coordinate, by writing TikZ code in the last (after the semicolon) optional argument \texttt{<code.append>}.  

89
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{<code.append>\texttt{}} extends the path after \tztosAtEnd.

\begin{tikzpicture}
\helplines(4,3)
\tztos[blue](0,1)(1,2)(3,1)(4,3);\--cycle
\tztos[red,thick]
\hspace{0.5cm}
(0,0)[{bend right}]
(1,1)
(2,1)[{out=-135,in=45}]
(3,0); < to ((turn)0:1.5cm) >
\end{tikzpicture}

13.4 \tztos+: Relative coordinates: Semicolon version

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

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

\begin{tikzpicture}
\helplines(4,3)
\tztos[blue](0,1)(1,2)(3,1)(4,3);\--cycle
\tztosAtBegin\hspace{0.5cm}
(1,3)\-|-\}
\tztosAtEnd\hspace{0.5cm}
to (([turn]0:1.5cm) }
\tztos[red,thick]
\hspace{0.5cm}
(0,0)[{bend right}]
(1,1)
(2,1)[{out=-135,in=45}]
(3,0);
\end{tikzpicture}

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 \texttt{to}, which can be changed with the \texttt{second} optional argument \texttt{[<code1>]} between two coordinates like, for example, \texttt{(coor1)[-]}\texttt{(coor2)}. Another way is to use the \texttt{first} curly brace option \texttt{\{}\texttt{<link style>\}}. You can also change it by \texttt{\settztlinkstyle\texttt{\{}\texttt{<link style>\}}} like, for example, \texttt{\settztlinkstyle{-}}\texttt{\}}, which is effective until the end of the \texttt{\tikzpicture} environment.
You can change the link style (to by default) using the optional argument \([\text{<code1>}]\) immediately after the first coordinate.

\begin{tikzpicture}
\zhelplines(4,3)
\tzlink{to}[bend left](0,0)(1,0)
\tzlink{to, bend right}(0,2)(3,0)
\tzlink{to, [above]}(1,2)(4,1)
\end{tikzpicture}

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

\begin{tikzpicture}
\zhelplines(4,3)
\settzlinkstyle{to}[bend left] \% local change
\tzlink{to, bend left}(0,0)(1,0)
\tzlink{to, [above]}(0,2)(3,0)
\tzlink{to, [above]}(1,2)(4,1)
\end{tikzpicture}

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

\begin{tikzpicture}
\zhelplines(4,3)
\tzlink{1,0}(3,1) \% works like:
\draw (1,0) to (3,1);
\end{tikzpicture}

\begin{tikzpicture}
\zhelplines(4,3)
\tzlink{1,0}[-|](3,1) \% works like:
\draw (1,0) -| ++ (3,1);
\end{tikzpicture}
Adding text  You can add text next to the line or curve of \texttt{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 \texttt{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.

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

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.

13.6 \tzlinks: All in one: Semicolon versions

13.6.1 \tzlinks: Standard version

The macro \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.

% \tzlink(+): path name: intersection
\begin{tikzpicture}
\tet{\helplines(4,3)}
\tzlink\[blue\]\"BB\"(0,3)(line B)[sloped](3,0)
\tzlink[-,red,bend right]\"CC\"(1,0)(3,2)(Line C)[r]
\tzXpoint\{(BB\})(OC\}(E) \% intersection
\tzdot*(E)(E)[a]
\end{tikzpicture}

% \tzlink(+): path name: intersection
\begin{tikzpicture}
\tet{\helplines(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]
<-| ++(-1,1) node \[a\] \{End!\} >
\tzXpoint\{(BB\})(OC\}(E)
\tzdot*(E)(E)[a]
\end{tikzpicture}

% syntax: minimum
\tzlinks \{(coor)\}(\{coor\}) ..repeated.. (\{coor\}) ;
% syntax: full
\tzlinks\{\path style\}[\{opt\}]<shift coor>"\path name"
\{(coor)\}\{\{text\}\}{\{node opt\}\{code2\}}\{\fill opacity\}\{code.append\}
% remark:
- <\text> must be link style, such as --, to, -|, etc.
  (possibly followed by other code)
- <\text> 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*+
\{to\}[]""\{\text\}\{node opt\}\{code\}..repeated..\{\text\}\{node opt\}\{code\} ; \{ \}

93
How to change link styles

There are three ways of changing link styles:

- The first 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}
\tzhelplines(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=-90,in=45]]
(3,0);
\end{tikzpicture}

% \tzlinks: various link styles
\begin{tikzpicture}
\tzhelplines(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 [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 100 for more details).

\begin{tikzpicture}
\tzhelplines(4,3)
\tzlinks*(1,1)(2,2)(3,1)(4,3); \% works like:
\draw (1,1) to (2,2) -- (3,1) to (4,3);
\end{tikzpicture}

\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}

\tzlinks++ is the plus version of \tzlinks*.  

95
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>` if 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>]`.

```latex
\tzlinks+[dashed]={AA}(1,1){(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};
```

```latex
\begin{tikzpicture}
\tzhelplines(4,3)
\tztlinks*[blue]{{Ace}(1,1){Beth}[sloped,a]
{3,0}[C] [near start]
{4,2}[End!][draw,a=3pt];}
\end{tikzpicture}
```
% \tzlinks: adding text
\begin{tikzpicture}
\tzhelplines(4,3)
\tzlinks[->](1,1){start}{sloped,b}++>
  (1,1){[rounded corners=10pt]--}K{at end,a}
  (0,3){[rounded]scale=0.7,font=\textscshape,at end,ar}
  (0,0){[sharp corners]--}
  (4,0)++>
  (-1,2){End!}[draw,blue,a=3pt];
\end{tikzpicture}

\begin{tikzpicture}
\tzhelplines(4,3)
\tzlinks<.5,-1>(0,1){start}++>
  (1,2){[edge[dashed,bend right]}Go![sloped,b]<(3,1)>
  (3,1)
  (4,3){\End\here!}[align=center,r,draw,red] ;
\end{tikzpicture}

\begin{tikzpicture}
\tzhelplines(4,3)
\tzlinks<.5,-1>(0,1){start}[at start,b] % shift
  (1,2){[to[dashed,bend right]}Go![sloped,b]<(3,1)>
  (3,1)
  (4,3){\End\here!}[align=center,r,draw,red] ;
\end{tikzpicture}

% \tzlinks: edge
\begin{tikzpicture}
\tzhelplines(4,3)
\tzlinks"AA"{(0,1){start}[at start,b] % shift
  (1,2){[edge[dashed,bend right]}Go![sloped,b]<(3,1)>
  (3,1)
  (4,3){\End\here!}[align=center,r,draw,red] ;
\end{tikzpicture}

% \tzlinks: edge
\begin{tikzpicture}
\tzhelplines(4,3)
\tzlinks"AA"{(0,1){start}[at start,b] % shift
  (1,2){[edge[dashed,bend right]}Go![sloped,b]<(3,1)>
  (3,1)
  (4,3){\End\here!}[align=center,r,draw,red] ;
\end{tikzpicture}

\begin{tikzpicture}
\tzhelplines(4,3)
\tzlinks<.5,-1>(0,1){start}[at start,b] % shift
  (1,2){[to[dashed,bend right]}Go![sloped,b]<(3,1)>
  (3,1)
  (4,3){\End\here!}[align=center,r,draw,red] ;
\end{tikzpicture}

\begin{tikzpicture}
\tzhelplines(4,3)
\tzlinks"AA"{(0,1){start}[at start,b] % shift
  (1,2){[edge[dashed,bend right]}Go![sloped,b]<(3,1)>
  (3,1)
  (4,3){\End\here!}[align=center,r,draw,red] ;
\end{tikzpicture}

Remark: Note that edge is not part of the main path.

Shift  You can move the path of \tzlinks with the option \texttt{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.

% \tzlinks:
\begin{tikzpicture}
\tzhelplines(4,3)
\tzlinks<.5,-1>(0,1){start}[at start,b] % shift
  (1,2){[to[dashed,bend right]}Go![sloped,b]<(3,1)>
  (3,1)
  (4,3){\End\here!}[align=center,r,draw,red] ;
\end{tikzpicture}

In the previous example, since edge is not part of the main path of \tzlinks, the second horizontal intersection point (A-2) is not on the edge curve.

Extending paths  You can extend the path of \tzlinks by writing Ti\textit{k}Z code in the last (after the semicolon) optional argument \texttt{code.append}. So \texttt{--cycle} closes the path with a straight line.
You can also use \tzlinksAtBegin and \tzlinksAtEnd to extend the path of \tzlinks at the beginning and at the end, respectively.Specifying the option \texttt{<code.append>} extends the path after \tzlinksAtEnd.

\begin{tikzpicture}
\helplines(4,3)
\tzlinks*\[blue\](0,1)(1,2)\[to\][bend right=60]\(0,1\)\(--\)\cycle
\tzlinks*\[fill=red\]
\(0,0\)\[to\][bend right]\(1,1\)
\(2,1\)\[to\][out=-135,in=45]\(3,0\); \texttt{not closed}
\end{tikzpicture}

\begin{tikzpicture}
\helplines(4,3)
\tzlinksAtBegin\{(0,2) \[to\][bend left]\}
\tzlinksAtEnd\{\[to\][out=210]\(1,3\)\}
\tzlinks*\[blue\](0,1)(1,2)\[to\][bend right]\(3,1\)\[out=-135,in=45]\(4,3\);
\tzlinks*\[fill=red\]
\(0,0\)\[to\][bend right]\(1,1\)
\(2,1\)\[to\][out=-135,in=45]\(3,0\); \(--\cycle\)
\end{tikzpicture}

14 Filling Area

14.1 \texttt{tzpath}: Semicolon version

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

You can visualize the path with \texttt{tzpath[draw]}, which is the same as \texttt{tzlinks}.

\begin{tikzpicture}
\helplines(4,3)
\tzlinks*\{\[to\][out=120]\(1,3\)\}
\tzlinks*\{\[to\][out=90]\(1,3\)\}
\end{tikzpicture}

\begin{tikzpicture}
\helplines(4,3)
\tzlinks*\{\[to\][out=120]\(1,3\)\}
\tzlinks*\{\[to\][out=90]\(1,3\)\}
\end{tikzpicture}

Remark:
• The main purpose of `\tzpath` is to fill an enclosed area with color or pattern. You can use `\tzpath[fill]` or `\tzpath[pattern=<...>]` to do it.

• Use `\settzpathlayer`, like `\settzpathlayer{behind}`, to change the layer of `\tzpath` (default: main).

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

`\tzpath` allows you to choose how to construct a path using the `<path style>` option in-between coordinates. **Path extension operation** can be selected from ‘--’, ‘to’, ‘|–’, ‘–|’, etc. You can use the first brace option `{<path style>}` to change all the `<path style>` in-between coordinates.

```latex
\begin{tikzpicture}
\tzhelplines(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 `\settzpathstyle`, like, for example, `\settzpathstyle{--}`. `\settzpathstyle` is an alias of `\settzlinkstyle`. The effect remains valid until the end of `tikzpicture` environment unless changed again.

```latex
\begin{tikzpicture}
\tzhelplines(4,3)
\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}
```

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

The starred version \texttt{\textbackslash path*} is the (almost) same as \texttt{\textbackslash tzlinks*\textbackslash [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*} and \texttt{\textbackslash tzlinks*\textbackslash [draw=none]} is that the layer of \texttt{\textbackslash path*} can be changed by \texttt{settzpathlayer}, like \texttt{settzpathlayer\textbackslash [behind]}.

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

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

You can also change the fill opacity by specifying the \texttt{last} curly brace optional argument \texttt{\{fill opacity\}}, \texttt{after the semicolon}. 

100
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.

14.3 \tzpath+ and \tzpath++: Relative coordinates: Semicolon versions

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

Everything else is the same as in \tzpath.

\tzpath++ is simply the plus version of \tzpath*.

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

There are many ways to draw curves.

15.1 Bézier curves

15.1.1 \texttt{\textbackslash bezier}

\texttt{\textbackslash bezier} 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, \texttt{(<cntl-coor>)}.

% syntax: minimal
\texttt{\textbackslash bezier(<start-coor>)(<cntl-coor>)(<last-coor>)}
\texttt{\textbackslash bezier(<start-coor>)(<cntl-coor>)(<cntl-coor>)(<last-coor>)}
% syntax: full
\texttt{\textbackslash bezier[<draw opt>]<shift coor>"<name path>"}
\texttt{(<start-coor>)(<cntl-coor>)(<cntl-coor>)(<last-coor>)}
{<text>}{<node opt>}{<code.append>}
% defaults
[]<>”(<m>)(){m}{}<>
The style \texttt{tzshowcontrols} displays the control points by drawing dotted lines, by default. You can also change the dotted line style, like \texttt{tzshowcontrols=(dashed,green)}.

```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)
\tzbezier[blue,dashed]<1,1>(0,2)(1,5)(4,0)
\tzbezier[blue,r]<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}
```

**Shift** You can move the curve of \texttt{tzbezier} by specifying the option \texttt{<shift coor>} before the first coordinate or immediately before the option \"<path name>\", if it exists. The \texttt{empty} shift option <> is not allowed.

```latex
\begin{tikzpicture}
\tzhelplines(4,4)
\tzbezier[blue,shift<1,1>](0,2)(1,5)(4,0){curve}[draw,black,r]
\tzbezier[blue,dashed]
\tzbezier[red]
\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 \texttt{tzbezier} by specifying the option \"<path name>\" immediately before the first coordinate.

```latex
\begin{tikzpicture}
\tzhelplines(4,4)
\tzbezier[blue]
  (0,2)(1,5)(4,0){curve}[draw,black,r]
\tzbezier[blue,dashed]
\tzbezier[red]
  (0,3)(1,2)(2,4)(4,3){bezier}[near end,a]
\tzbezier[red,dashed]
\end{tikzpicture}
```
Extending paths  You can extend the path of \texttt{tikzpicture} from the last coordinate, by writing Ti\textsc{k}Z code in the last optional argument \texttt{<code.append>}.

\begin{verbatim}
\begin{tikzpicture}
thelplines(4,4)
tbez[red](0,3)(1,2)(2,4)(4,3)(bezier)[near end,a] < to [bend right] ++(-1,-1) node [below] {A} >
tbez[blue,->]()(0,2)(1,5)(4,0)
\end{tikzpicture}
\end{verbatim}

You can also use \texttt{bezAtBegin} and \texttt{bezAtEnd} to extend the path of \texttt{tikzpicture} at the beginning and at the end, respectively. Specifying the option \texttt{<code.append>} extends the path after \texttt{bezAtEnd}.

\begin{verbatim}
\begin{tikzpicture}
thelplines(4,4)
bezAtEnd
{ to [bend right] ++(-1,-1) node [below] {A} }
bez[red](0,3)(1,2)(2,4)(4,3)(bezier)[near end,a]
bezAtBegin((0,0) to [out=45,in=-30] )
bezAtEnd([-]++(-2,1)}
bez[blue,->]()()(0,2)(1,5)(4,0){[b]}
arc (0:90:1) node [a] {End!} >
\end{tikzpicture}
\end{verbatim}

15.1.2 \texttt{bez+}: Relative coordinates

For the plus version \texttt{bez+}, the last coordinate is relative to the first coordinate. And the first control point is relative to the first coordinate.

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

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

Therefore, for \texttt{bez+}(A)(B)(C)(D), (B) and (D) are relative to (A), and the second control point (C) is relative to the last coordinate (D).
% four coordinates
\tbezert(0,1)(1,-1)(-2,1)(4,3) % works like:
\draw (0,1) ..controls +(1,-1) and +(-2,1).. (4,3);%

% \tbezert+
\begin{tikzpicture}
\helplines(4,4)
\bezert(0,3)(1,-1)(-2,1)(4,0)
\bezert[blue,->](0,2)(1,-1.5)(4,-2)
\end{tikzpicture}

% \tbezertAtBegin, \tbezertAtEnd
\begin{tikzpicture}
\helplines(4,4)
\bezertAtEnd
\bezertAtBegin(1,0) to [out=150,in=30]
\bezertAtEnd\bezert[blue,->](0,2)(1,-1.5)(4,-2)
\end{tikzpicture}

15.2 Parabolas

15.2.1 \tzparabola

\tzparabola accepts 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.

% two coordinates
\tzparabola(0,0)(3,2) % works like:
\draw (0,0) parabola (3,2);

% three coordinates
\tzparabola(0,0)(1,1)(3,2) % works like:
\draw (0,0) parabola bend (1,1) (3,2);

% syntax: minimal
\tzparabola(<coor>)(<coor>)
\tzparabola(<coor>)(<coor>)(<coor>)
% syntax: full
\tzparabola[<opt>]<shift coor>"<path name>" 
\draw (<coor>)(<coor>)(<coor>){<text}>{<node opt>}{code.append}
% default
[]<>"{<opt}>(){<coor>}{<coor>}{<coor>}{<text>}{<node opt>}{code.append}
Parabolas \texttt{tzplot} draws the graph of a quadratic function \( f(x) = ax^2 + bx + c \) for appropriate values of \( a \), \( b \), and \( c \).

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

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

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

15.2.2 \tzparabola+: Relative coordinates

The \textit{plus version} \tzparabola+ use the second and the third coordinates \texttt{xemrelative} to the first coordinate.

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


15.3 Edges

15.3.1 \tzedge(+) 

The macro \tzedge connects two coordinates with a straight or curved line, using the Ti\kZ\ edge operation.

% syntax
\tzedge[<opt>]<shift coor>
  (ppo){{text}}{(ppo)}{{text}}{<opt>}{<opt>}{code.append}
% defaults
[]<>(<m>)[]<>(<m>)[]<> 

Remark: In Ti\kZ, 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.

\tzedge(1,1)\,(3,2) \% works like:
\draw (1,1) edge (3,2);

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

% \tzedge
\begin{tikzpicture}
\tzhelplines(4,3)
\tzedge[->](0,1)\,(2,3)
\tzedge[on=0,dashed]
  (0,0){edge}[near start,sloped,a](4,2)
\tzedge{in=90,blue}(2,0)\,(4,1){C}[r]
\end{tikzpicture}

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

\tzedge+(1,1)\,(3,2) \% works like:
\draw (1,1) edge ++(3,2);

\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)
Remark: edge in the option <code.append> works with the second coordinate, but it works with the last node if the second option {<text>} is not empty. (In TikZ, if edge is preceded by node, the node is its start point.)

15.3.2 \tzedges(+): Semicolon versions

The macro \tzedges accepts any number of coordinates to draw edges from the first coordinate to each of the next using the TikZ edge operation. That is, the first coordinate is the unique start coordinate, and all others are target coordinates. \tzedges is a semicolon version, so you need to type ; to indicate when the coordinate iteration ends.

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

- Each edge is drawn independently from a main path as well as any other edge’s. (See TikZ manual for more details.)
- The edge operation of TikZ does not change change anything about a main path, so the current point is not changed by \tzedges. This means that the last node (with {<text>} and [node opt)]) works with the first (namely, start) coordinate.
\begin{tikzpicture}
  \helplines(4,3)
  \zedges(0,1){A}(1,2)
  [blue,thick]{edge}[b](3,1)
  (4,3);
\end{tikzpicture}

\zedges(0,1){A}(1,2)[blue,thick]{edge}[a](3,1)(4,3); % works like
\draw (0,1) edge node {A} (1,2)
   edge [blue,thick] node [above] {edge} (3,1)
   edge (4,3);

\begin{tikzpicture}
  \helplines(4,3)
  \zedges[bend right]<1,-1> % shift
  (0,1)[->,thick,blue]{A}
  (1,2)[dotted]{Beta}[b]
  (3,1)[dashed]{Cate}[near end,sloped,a]
  (4,3);
\end{tikzpicture}

The plus version \texttt{\textbackslash tzedges+} uses the second and next coordinates relative to the first coordinate.

\zedges+(0,1)(2,1)(3,0)(4,2); % works like
\draw (0,1) edge ++ (2,1)
   edge ++ (3,0)
   edge ++ (4,2);

\begin{tikzpicture}
  \helplines(4,3)
  \zedges+[bend right](0,1)[bend right=0]{A}
  (1,1)[dotted]{Beta}[b]
  (3,0)[dashed]{Cate}[near end,sloped,a]
  (4,2);
\end{tikzpicture}

\textbf{Remark:} The option \texttt{\textbackslash 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)
\edges+(0,1)[bend right=0,->]{A}{\fill=white}
(1,1)[bend right,dotted]{Beta}[midway]
(3,0)[bend right,dashed]{Cate}[near end]
(4,2);
< edge [bend left,thick,blue] (1,3) >
\end{tikzpicture}

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

See Section 10.6 on page 63, for more details on \tzplotcurve.
See Section 10.1 on page 57, for more details on \tzplot.

15.4.2 \tzto, \ztos
You can draw curves with \tzto and \ztos. \ztos are quite useful to draw various curves.

\begin{tikzpicture}
\helplines(4,3)
\coors*(0,0)(A)(1,2)(B)(2,2)(3,3)(4,1){E};
\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 85, for more details on \tzto.
See Section 13.3 on page 87, for more details on \ztos.

15.4.3 \tzlink, \tzlinks
You can also draw curves with \tzlink and \tzlinks.

111
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 148, 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>"
\tzpolygon[<opt>] ( <coor> ){<text>}[<node opt>]
..repeated.. (){}[] ;<code.append>
% defaults
[]<>""(<m>){}
\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.

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

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

\begin{tikzpicture}
\helplines(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 \tzpolygon*

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

% 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>"
\hfill{(coor)}{<text>}{<node opt>}.repeated..{(coor)}{<text>}{<node opt>}<code.append>

% defaults
*{[fill=black!50,fill opacity=.3,text opacity=1]}<>{(m)}{(m)}.. repeated..(){}[] ;

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

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

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

16.1.3 \tzpolygon+, \tzpolygon**: Relative coordinates: Semicolon versions

The plus version \tzpolygon+ uses each coordinate (except the first one) relative (with ++) to the previous coordinate.

Everything else is the same as in \tzpolygon.

\tzpolygon++ is just a plus version of \tzpolygon*.

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

\tzpolygon+[dashed]"AA"(1,1)(2,2){A}(3,1){B}[below]; % works like:
\draw [dashed,name path=AA] (1,1)
\hfill -- ++(2,2)
\hfill -- ++(3,1) node {A}
\hfill -- ++(4,3) node [below] {B}
\hfill -- cycle;
\% \tzpolygon: shift
\begin{tikzpicture}
\zhelplines(4,3)
\tzpolygon*[blue,auto]
\{0,1\}(Side A)[sloped,red]
\{1,2\}(B)
\{1,0\}
\{1,-1\}(Side D)[swap,sloped]
\{-2,-2\}
\tzpolygon*[green,auto,dashed,text=black]<1,-.5>
\{0,1\}\{1,2\}\{1,0\}\{1,-1\}\{-2,-2\} \{.7\}
\end{tikzpicture}

16.2 Rectangles

16.2.1 \tzframe and its variants
\tzframe accepts two coordinates draws a rectangle.

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

\tzrectangle and \tbox are aliases of \tzframe.

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

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

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

\% \tzframe, \tzframe+
\begin{tikzpicture}
\zhelplines(4,3)
\tzframe\{0,0\}(3,2)
\tzframe+(blue,rounded corners=2mm)(1,3)(1,-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 \tbox* are aliases of \tzframe*.)
\tzframe* has also its starred version \tzframe++. (\tzrectangle++ and \tbox++ are aliases of \tzframe++.)

\% syntax
\tzframe*\{<opt>\}<shift coor>"<path name>"\{<coo1>\}(<coor2>)\{fill opacity\}<code.append>
With the starred versions, you can change the fill opacity using the last option `{fill opacity}`.

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

You can use the last option `<code.append>` to add more TiKZ code.

16.2.2 `\tzrectanglering(*)`

`\tzrectanglering*` 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`. 
\tzrectanglering*(A1)(A2)(B1)(B2) \% works like:
\draw [fill=black!50, fill opacity=.3, text opacity=1, even odd rule]
(A1) rectangle (A2)
(B1) rectangle (B2);

\% \tzrectanglering*
\begin{tikzpicture}
\tzhelplines(4,2)
\tzrectanglering*[blue](0,0)(3,2)(.5,.5)(2.5,1.5)
\end{tikzpicture}

\% \tzrectangling*: shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzrectangling*[blue](0,0)(3,2)(.5,.5)(2.5,1.5)
\tzrectangling*[red]<.9,.9>(0,0)(3,2)(.5,.5)(2.5,1.5)
\end{tikzpicture}
\tzrectanglering draws two rectangles with the default option even odd rule.

\% syntax: minimal
\tzrectangling(<coorA1>)(<coorA2>)(<coorB1>)(<coorB2>)
% syntax: full
\tzrectangling*[<opt>]<shift coor>(<coorA1>)(<coorA2>)
\( (<coorB1>)(<coorB2>)\)<code.append>
% defaults
[even odd rule]<>(<m>)(<m>)()()<>

\tzrectangling(0,0)(2,2)(.5,.5)(1.5,1.5) \% works like:
\draw (0,0) rectangle (2,2)
(.5,.5) rectangle (1.5 and 1.5);

\% \tzrectanglering: shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzrectanglering[blue](0,0)(3,2)(.5,.5)(2.5,1.5)
\tzrectanglering[red,fill=green]<.9,.9>
(0,0)(3,2)(.5,.5)(2.5,1.5)
\end{tikzpicture}

\% \tzrectangling* 
\begin{tikzpicture}
\tzhelplines(4,3)
\tzrectangling*[blue](0,0)(3,2)(1.5,1.5)(4,3)
\end{tikzpicture}
With the last option <code.append> you can add some more TiKZ code.

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

With the last option <code.append> you can add some more TiKZ code.

\begin{tikzpicture}
\helplines(4,4)
\framering[blue](0,0)(3,4)
\boxring[pattern=bricks](1,1)(4,3)
\end{tikzpicture}

\framering and \boxring are aliases of \rectangle.
\framering* and \boxring* are aliases of \rectangle*.

16.3 Circles and rings

16.3.1 \circle

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

\begin{tikzpicture}
\helplines(4,3)
\circle(1,1)(1cm)
\circle[blue,dashed]"AA"(2,2)(1cm)
\vXpointat{AA}{2.5}(A)
\dots*(A-1)\$A_1\$[r](A-2)\$A_2\$[r];
\end{tikzpicture}

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

% \tzcircle(*): shift
\begin{tikzpicture}
\tzhelplines(4,4)
\tzcoors(2,2)(A)(3,1)(B);
\tzcircle(2,2)(1)
\tzcircle[blue,dashed]<-.5,0>(A)(1)
\tzcircle*[3,1](.5cm)
\tzcircle*[blue]<.5,0>(B)(.5cm)
\end{tikzpicture}

With the last option `<code.append>`, you can add some Ti\textit{k}Z code.

% \tzcircle(*): <code.append>
\begin{tikzpicture}
\tzhelplines(4,4)
\tzcoors(2,2)(A)(3,1)(B);
\tzcircle*[blue,even odd rule](A)(1cm)
  (A)(1cm)
\tzcircle*[fill=green,even odd rule](B)(1cm)
  (B)(1cm)
\end{tikzpicture}

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`.

% syntax: minimal
\tzring*\{<coor>\}(\radius)(<coor>\}(\radius)
% syntax: full
\tzring*[<opt>]<shift coor>
  (\coor)(\radius)(\coor)(\radius)\{fill opacity\}<code.append>
% defaults:
*\{even odd rule,fill=black!50,fill opacity=.3,text opacity=1\}<>(m)\{(m)\{(m)\{(m)\{(m)\{.3\}<>}

\tzring*(0,0)(1cm)(0,0)(1.5cm) % works like:
\draw [fill=black!50,fill opacity=.3,text opacity=1, even odd rule]
  (0,0) circle (1cm) (0,0) circle (1.5cm);
\tzring* draws two circles with the default option even odd rule.

\begin{tikzpicture}
\zhelplines(4,4)
\tzring*[green](2,2)(1)(2,2)(1.5) \%
\tzring*[fill=blue](3,1)(1)(3,1)(.7) \%
\end{tikzpicture}

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

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

\begin{tikzpicture}
\zhelplines(4,3)
\tzframe(0,0)(4,3)
\tzring*[blue](1.5,1.5)(1)(2.5,1.5)(1) \%
\end{tikzpicture}

% syntax: minimal
\tzring*\langle\text{coor}\rangle\langle\text{radius}\rangle\langle\text{coor}\rangle\langle\text{radius}\rangle
% syntax: full
\tzring*[\text{opt}]\langle\text{shift coor}\rangle
\langle\text{coor}\rangle\langle\text{radius}\rangle\langle\text{coor}\rangle\langle\text{radius}\rangle\langle\text{code.append}\rangle
% defaults:
[\text{even odd rule}]\langle\langle\text{m}\rangle\rangle\langle\langle\text{m}\rangle\rangle\langle\rangle\langle\rangle

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

You can add some Ti\textit{k}Z code with the last option <\text{code.append}>.
\begin{tikzpicture}
\zhelplines(4,3)
\tzring*[blue](2,2)(1) (1.5,2) circle (.3 and .5) >
\tzring*[pattern=bricks](3,1)(1)
\hspace{2cm} < (2.5,.5) rectangle ++(1,1) >
\end{tikzpicture}

\begin{tikzpicture}
\zhelplines(4,3)
\tzring*[fill=red,draw=none,nonzero rule](1,1)(1)
\hspace{2cm} (3,.5) rectangle (1.5,3)>
\tzcircularering[fill=white](1,1)(1)
\hspace{2cm} (3,.5) rectangle (1.5,3)>
\end{tikzpicture}
\tzcircularering and \tzcircularering* are aliases of \tzring and \tzring*, respectively.

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. \tzellise(*) is basically the same as \tzcircle(*).

% syntax
\tzellipse*[<opt>]<shift coor>"<path name>"
\hspace{2cm} (<coor>)(<x and y radius>){<fill opacity>}{<code.append>}
% defaults: \tzellipse*
*{fill=black!50,fill opacity=.3,text opacity=1}<>""(<m>)(<m>)<>
% defaults: \tzellipse
*{}<>""(<m>)(<m>)<>

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

You can move the ellipse by specifying the option <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.

% \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)
\tzellipse*[red]<0,-.5>(3,1)(1cm and .5cm) % shift
\end{tikzpicture}

You can add some TikZ code with the option <code.append>.
\tzoval is an alias of \tzellipse and \tzoval* is an alias of \tzellipsering.*

16.4.2 \tzellipsering(*)

\tzellipsering* draws two ellipses and draws an ellipse ring by filling the interior with the default options even odd rule, fill=black!50, fill opacity=.3, and text opacity=1.

\tzellipse draws two ellipses with the default option even odd rule.

\tzellipsering(*) is basically the same as \tzring(*).

% syntax: minimal
\tzellipsering* (<coor>)(<x and y radius>)(<coor>)(<x and y radius>)
% syntax: full
\tzellipsering*[<opt>]<shift coor>
    (<coor>)(<x and y radius>)(<coor>)(<x and y radius>)
    {<fill opacity>}<code.append>
% defaults: \tzellipsering*
*{[even odd rule,fill=black!50,fill opacity=.3,text opacity=1]}
    <>(<m>){(<m>)}{(.3)}<>
% defaults: \tzellipse
*{[even odd rule]}{<>}{(<m>){(<m>)}{()}}<>
\begin{tikzpicture}
\tzhelplines(4,4)
\tzcoors(2,2)(A)(3,1)(B);
\tzellipsering*[blue](2,2)(1 and 1.5)
\hspace{1.5pt} < (1.5,2.5) circle (3mm) >
\tzellipsering[pattern=bricks](2,2)(1 and 1.5)
\hspace{1.5pt} < (2.5,.5) rectangle ++(1,1) >
\end{tikzpicture}

\begin{tikzpicture}
\tzhelplines(4,4)
\tzcoors(2,2)(A)(3,1)(B);
\tzovalring*[fill=red,draw=none,nonzero rule]
\hspace{1.5pt} (2,2)(1 and 1.5)(3,1)(1 and 1.5){1}
\tzovalring[fill=white]
\hspace{1.5pt} (2,2)(1 and 1.5)(3,1)(1 and 1.5)
\end{tikzpicture}

\tzovalring is an alias of \zellipsering and \tzovalring* is an alias \zellipsering*.

17 Arcs, Wedges, and Angle Marks

17.1 \texttt{\textasciitilde zarc\texttt{}}: Centered arcs

17.1.1 Arcs

\tarc draws an arc around a specified center coordinate.

\begin{verbatim}
\tarc(1,1)(30:120:1) \% works like:
\draw (1,1) ++(30:1) arc (30:120:1);
\end{verbatim}

The swap version \texttt{\textasciitilde zarc}’ switches its drawing direction from counterclockwise to clockwise and vice versa.

\begin{verbatim}
\tarc’(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 \{<text>\} and \{<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 \texttt{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 \textit{elliptical arc} if you specify x-radius and y-radius.
17.2 \texttt{\tzarcfrom}('): Arcs as in \texttt{TikZ}

\texttt{\tzarcfrom} draws an arc starting from a specified point, like \texttt{TikZ} 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)
\twzots*(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)
\twzots*(1,1)(2,2);
\tzarc[blue,fill=green,fill opacity=.3](2,2)(0:-270:1 and 2)<--(2,2)--cycle \texttt{\% code.append}
\tzarc[->,blue,dashed]<1,0>(2,2)(0:-270:1 and 2) \texttt{\% shift}
\end{tikzpicture}
```

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

```
\begin{tikzpicture}
\helplines(4,3)
\ttwzots*(1,1)(3,2);
\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}
```

% \texttt{\tzarc} is the swap version of \texttt{\tzarcfrom}.

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

\texttt{\tzarcfrom'} is the swap version of \texttt{\tzarcfrom}.
17.3 \texttt{\textbackslash tzarcsfrom}: Connected arcs: Semicolon version

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

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

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

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

% flag: step 2
\begin{tikzpicture}
\edef\x{atan(2/3)}
\tzarcsfrom[red](0,0)
\{\texttt{red 1}\} (\texttt{-\x:-180:1})[midway]
\tzarcsfrom[blue](0,0)
\{\texttt{blue 1}\} (\texttt{-\x-180:2})[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 \code{append}.

% syntax
\tzwedge[<opt>][<shift coor>]<"<path name>""[<coor>]<<angA>:<angB>:<radius>><<text>><[<node opt>]
% defaults
\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*(')

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

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

\begin{tikzpicture}
\tzcoor*(2,1)(A)
\zhelplines(4,3)
\tzwedge[blue,very thick]<.1,.1>(A)(30:120:1.5) % shift
\tzarc[->,lightgray]<.1,.1>(A)(30:120:1.3) % shift
\tzwedge'[dashed](A)(0:135:1.5)[clockwise][sloped,red]
\tzarc'[->,lightgray](A)(0:135:1.3)
\tzwedge'[fill]<2,0>(A)(-60:240:1) % shift
\end{tikzpicture}

\begin{tikzpicture}
\tzcoor*(2,1)(A)
\zhelplines(3,3)
\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 between two points and allows you to use it.
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 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 with with \texttt{\settzAAlinestyle}.

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

\begin{tikzpicture}[font=\scriptsize]
\tzhelplines(5,4)
\tzcoors*(4,2)(A){(1,1)}(B){[180](2,3)}(C){[135]};
\tzline[tzextend={1cm}{1cm}](B)(A)
\tzline[tzextend={1cm}{1cm}](B)(C)
\tzpointangle(B)(A){\myAngA}
\tzpointangle(B)(C){\myAngC}
\tzarc(1,1){\myAngA}:{\myAngC}:10pt\%(\theta)[ar,midway]
\tzarc(1,1){\myAngA}:{\myAngC}:11pt
\end{tikzpicture}

\begin{tikzpicture}
\tzcoors*(4,2)(A){(1,1)}(B){(2,3)}(C){};
\tzlines(A)(B)(C);
\tzanglemark(A)(B)(C){\theta} % angle mark
\end{tikzpicture}

\begin{tikzpicture}
\tzcoors(4,2)(A){(1,1)}(B){(2,3)}(C){};
\tzlines(A)(B)(C);
\tzlines[thick,blue](A){(2.5,0)};
\tzanglemark(A)(B){\theta}
\settzAAradius{20pt}
\tzanglemark(C){(2.5,0)}(A){\alpha}[pos=.7]
\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 only after running \texttt{tzanglemark}.

- \texttt{tzanglemark} draws an angle mark between two angles, \textit{counterclockwise}, from small to large.
- \texttt{tzanglemark(‘)} draws an angle mark between two angles, \textit{clockwise}, from small to large.

\setcounter{tocdepth}{2}

\section*{Angle marks}

\subsection*{\texttt{\settzAAradius}}

You can change the radius of the angle arc by \texttt{\settzAAradius}.

\subsection*{\texttt{\settzanglelayer}}

You can change the layer of the angle mark by \texttt{\settzanglelayer}.

\subsection*{\texttt{\settzAAlinestyle}}

You can change the line width of the angle mark by \texttt{\settzAAlinestyle}.

\subsection*{\texttt{\settzanglemarklayer}}

You can change the layer of the angle mark by \texttt{\settzanglemarklayer}.

\subsection*{\texttt{\settzanglemarklayer}}

You can change the layer of the angle mark by \texttt{\settzanglemarklayer}.

\subsection*{\texttt{\settzanglemarkwidth}}

You can change the width of the angle mark by \texttt{\settzanglemarkwidth}.

\subsection*{\texttt{\settzanglemarktext}}

You can add angle text by \texttt{\settzanglemarktext}.

\subsection*{\texttt{\settzanglemarktextalpha}}

You can add angle text by \texttt{\settzanglemarktextalpha}.

\subsection*{\texttt{\settzanglemarktextdir}}

You can add angle text by \texttt{\settzanglemarktextdir}.

\subsection*{\texttt{\settzanglemarktextfont}}

You can add angle text by \texttt{\settzanglemarktextfont}.

\subsection*{\texttt{\settzanglemarktextsize}}

You can add angle text by \texttt{\settzanglemarktextsize}.

\subsection*{\texttt{\settzanglemarktextstyle}}

You can add angle text by \texttt{\settzanglemarktextstyle}.

\subsection*{\texttt{\settzanglemarktextweight}}

You can add angle text by \texttt{\settzanglemarktextweight}.

\subsection*{\texttt{\settzanglemarktextxy}}

You can add angle text by \texttt{\settzanglemarktextxy}.

\subsection*{\texttt{\settzanglemarktextxydir}}

You can add angle text by \texttt{\settzanglemarktextxydir}.

\subsection*{\texttt{\settzanglemarktextxyfont}}

You can add angle text by \texttt{\settzanglemarktextxyfont}.

\subsection*{\texttt{\settzanglemarktextxysize}}

You can add angle text by \texttt{\settzanglemarktextxysize}.

\subsection*{\texttt{\settzanglemarktextxystyle}}

You can add angle text by \texttt{\settzanglemarktextxystyle}.

\subsection*{\texttt{\settzanglemarktextxyweight}}

You can add angle text by \texttt{\settzanglemarktextxyweight}.

\subsection*{\texttt{\settzanglemarktextxyxy}}

You can add angle text by \texttt{\settzanglemarktextxyxy}.

\subsection*{\texttt{\settzanglemarktextxyxydir}}

You can add angle text by \texttt{\settzanglemarktextxyxydir}.

\subsection*{\texttt{\settzanglemarktextxyxyfont}}

You can add angle text by \texttt{\settzanglemarktextxyxyfont}.

\subsection*{\texttt{\settzanglemarktextxyxysize}}

You can add angle text by \texttt{\settzanglemarktextxyxysize}.

\subsection*{\texttt{\settzanglemarktextxyxystyle}}

You can add angle text by \texttt{\settzanglemarktextxyxystyle}.

\subsection*{\texttt{\settzanglemarktextxyxyweight}}

You can add angle text by \texttt{\settzanglemarktextxyxyweight}.
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoors*(4,2){(A){(1,1)(B){[180](2,3)(C){(C)}}
\tzlines(A){(B){(C)}};
\tzanglemark[->]{(A){(B){(C)}}{$\theta$}}
\tznode{scale=.7}{(3,1}{ONE: \tzangleONE}{r}
\tznode{scale=.7}{(3,.5}{TWO: \tzangleTWO}{r}
\tznode{scale=.7}{(3,0}{\theta}{\tzangleresult}{r}
\end{tikzpicture}

Remark: Simple to use:

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

\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoors*(4,2){(A){(1,1)(B){[180](2,3)(C){(C)}}
\tzlines(A){(B){(C)}};
\tzanglemark[->]{(C){(B){(A)}}{$\theta$}}
\tznode{scale=.7}{(3,1}{ONE: \tzangleONE}{r}
\tznode{scale=.7}{(3,.5}{TWO: \tzangleTWO}{r}
\tznode{scale=.7}{(3,0}{\theta}{\tzangleresult}{r}
\end{tikzpicture}

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

\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoors*(4,3)(A){(1,2)(B){(3,0)(C){(C){[0]}}
\tzlines[thick]{(A){(B){(C)}};
\tzanglemark[->]{(A){(B){(C)}}{$\theta$}}
\tznode{scale=.7}{(3,2}{ONE: \tzangleONE}{r}
\tznode{scale=.7}{(3,1.5}{TWO: \tzangleTWO}{r}
\tznode{scale=.7}{(3,1}{\theta}{\tzangleresult}{r}
\tzanglemark[->]{(A){(B){(C)}}{$\theta'$}{(15pt)} % swap
\end{tikzpicture}

Angle mark text position The midpoint of an angle arc is stored under the coordinate name tzAAmid. The angle mark text is put on the line that goes through the middle point and (tzAAmid). The default (<arc radius>) is 10pt and the default position of angle text is pos=1.5 in [<node opt>].
\begin{tikzpicture}
\zhelplines(4,3)
\tzcoors*(4,2)(A){(1,1)(B){[180]{2,3}}(C){[0]}}	
\tzlines(A)(B)(C);
\tzanglemark(C)(B){$\theta$}[pos=.65](20pt) %
\tzdot*[tzAAmid]
\tzline[red,dashed,tzextend={1cm}{2cm}](B)(tzAAmid)
\end{tikzpicture}

\begin{tikzpicture}
\zhelplines(4,3)
\tzcoors*(4,2)(A){(1,1)(B){[180]{2,3}}(C){[0]}}	
\tzlines(A)(B)(C);
\tzanglemark*[red](C)(B){$\theta$}
\end{tikzpicture}

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

17.5.3 \tzanglemark*('): Fill angle marks

\tzanglemark* 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.

% syntax: minimum
\tzanglemark*\{}(coorA){(coorB)}\{}(coorC)\{
% syntax: full
\tzanglemark*[\{opt\}\{}(coorA){(coorB)}\{}(coorC)\}{\{text\}\{}[\{node opt\}\{}(arc radius)\{
% defaults
[very thin](<m>)(<m>)(<m>)\{}[pos=1.5](10pt)
% syntax
\tzanglemark*[\{opt\}\{}(coor1){(coor2)}\{}(coor3)\}{\{text\}\{}[\{node opt\}\{}(arc radius)\}{\{fill opacity\}\{}% defaults
*\{very thin,draw=none,fill=black!50,fill opacity=.3,\}
text opacity=1\}(<m>)(<m>)(<m>)\}\{}[pos=1.5](10pt)\{.3

% \tzanglemark* is the swap version of \tzanglemark*.

% \tzanglemark*' is the swap version of \tzanglemark*.'
\begin{tikzpicture}
\zhelplines(4,3)
\tzcoors*(4,2)(A){(1,1)(B){[180]{2,3}}(C){[0]}}	
\tzlines(A)(B)(C);
\tzanglemark*[red](C)(B){$\theta$}
\end{tikzpicture}

\tzanglemark*'\{}(blue)(A)(B)(C){$\theta'$}(15pt) % swap
\end{tikzpicture}
\begin{tikzpicture}
\settzAAlinestyle{thick}
\tzhelplines(4,3)
\tzcoors*(4,3)(A){A}(1,2)(B){B}(3,0)(C){C}[0] ;
\tzlines[thick,blue](A){A}(B){B}(C)
\tzanglemark*[red](A){A}(B){B}(C){\$\theta\$}
\tzanglemark(A){A}(B){B}(C)
\settzAAlinestyle{very thin} % default
\tzanglemark' (A){A}(B){B}(C){14pt}
\tzanglemark'(C){C}(B){B}(A){A}{15pt}{\$\theta'$}
\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 \texttt{behind} layer by default, which can be changed by \texttt{\settzanglelayer}.

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

\begin{tikzpicture}[font=\scriptsize]
\tzhelplines(4,3)
\tzcoors(1,2){A}(1,0){B}(3,0){C}[0];
\tzlines(A){A}(B){B}(C)
\tzrightanglemark(A){A}(B){B}(C){90\textdegree}
\end{tikzpicture}

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 \texttt{(tzRAsvertex)} as the coordinate of the right angle mark vertex. The angle text is placed on the line going through the second coordinate and \texttt{(tzRAsvertex)}. 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 \tZErightanglemark.

17.5.5 \tZErightanglemark*: Fill right angle marks

The starred version \tZErightanglemark* 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 \settztanglelayer. Its alias is \settztanglemarklayer.

With \settztzfillcolor and \settztztzfillopacity, you can also change the default fill color and fill opacity.
% \setzRAsize
begin{tikzpicture}[scale=.8,font=\scriptsize]
tzhelplines(5,5)
tzcoorsquick(0,5)(A)(4,0)(B)(0,1)(C)(5,5)(D);
tzline[dotted]"AB"(A)(B)
tzline[dotted]"CD"(C)(D)
tzxpoint[AB]{CD}(E)
\setzRAsize{20pt} \%%
tzrightanglemark*[red](A)(E)(D)\($\rho$\)[pos=1.3]
tzrightanglemark[blue,thin](A)(E)(D)
tzrightanglemark*[draw=blue,thin,fill=green](B)(E)(C)
end{tikzpicture}
Part IV
Plotting Graphs

18 Axes

18.1 Draw axes

18.1.1 \texttt{tzaxes}

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

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

% syntax: minimal
\texttt{tzaxes(<x2,y2>){x-axis name}{y-axis name}}

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

% defaults
\begin{itemize}
  \item [-1]<0,0>(0,0){cm}{right}{above}
\end{itemize}

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

Here, \texttt{(cm)} stands for a mandatory argument.

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

\begin{tikzpicture}[scale=.5]
\tzhelplines(-1,-1)(7,7)
\tzshoworigin*
\tzaxes[thick,blue]<1,2>(7,7){$x$}{$y$}
\tzaxes[-,dashed](7,7){$X$}{below}{$Y$}{left}
\tzaxes[6,1]{(6,1)}{(3,6)}{$a$}{left}{$b$}
\end{tikzpicture}

\textbf{Shift} By default, the \texttt{x} and \texttt{y} axes intersect at \texttt{(0,0)}. Specifying the option \texttt{<x-shift,y-shift>} moves the axes to intersect at \texttt{(x-shift,y-shift)}.
18.1.2 \texttt{\textbackslash tzaxes*}

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

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

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

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

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

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

\begin{verbatim}
% syntax
\tzaxisy[<opt>]<x-shift>{<from>}{<to>}{<y-axis name>}[<node opt>]
% defaults
[-]>0{(<m>){(<m>)}}{[above]}
% arguments:
[#1]: line style, arrow type (for y-axis)
<#2>: x-shift of y axis
{#3}: y-axis starts from \% mandatory
{#4}: y-axis runs to \% mandatory
{#5}: y-axis name
[#6]: y-axis name option \% node option
\end{verbatim}
18.3 Display the origin

18.3.1 \tzshoworigin

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

All arguments of \tzshoworigin are optional.

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 \tzshoworigin*

\tzshoworigin* prints a node dot at the origin with no text by default. Internally the dot is processed by \tzdot*. All arguments are optional.
You can add text with the option \{<text>\}. The default size of the dot is 2.4pt, and it can be changed with the last option \{<dot size>\}. You can change the dot style using the first optional argument \{<dot opt>\}. You can also move the dot by specifying the option \{<shift coor>\}.

\textbf{Remark:} For \texttt{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 \{angle\}.)

\textbf{18.4 tzaxesL(‘): L-type axes}

\texttt{tzaxesL} is similar to \texttt{tzaxes}, but it draws only the ‘L’ type axes with (\langle x_1, y_1 \rangle) as the origin and (\langle x_2, y_2 \rangle) as the opposite corner of the rectangle. Those two coordinates are mandatory.
The swap version \texttt{\textbackslash tzaxesL'} swaps \texttt{(<x1,y1>)} and \texttt{(<x2,y2>)}. That is, \texttt{\textbackslash tzaxesL'}(A)(B) is equivalent to \texttt{\textbackslash tzaxesL}(B)(A).

\begin{tikzpicture} [scale=.5]
\tzhelplines(8,7)
\tzshoworigin
\tzaxes(8,7){x}{y}
\tzaxesL[red,thick](2,2)(6,5){a}{b}
\tzaxesL'[blue,dashed,->](2,2)(6,5){c}{d}
\tzaxesL'[-](7,1)(4,7){m}[draw,r]{n}[draw,circle]
\end{tikzpicture}

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

\begin{tikzpicture} [scale=.5]
\tzhelplines(8,7)
\tzshoworigin
\tzaxes(8,7){x}{y}
\tzaxesL[red,thick](2,2)(6,5){a}{b}
\tzaxesL[red,thick]<.5,.5>(2,2)(6,5){a'}{b'}
\tzaxesL'[blue,dashed,->](2,2)(6,5){c}{d}
\tzaxesL'[blue,dashed,->]<-1,-1>(2,2)(6,5){c'}{d'}
\end{tikzpicture}

19 \hspace{3cm} 19.1 \texttt{\textbackslash tzticks}: Tick labels

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

\begin{tikzpicture}
\begin{scope}[local bounding box=ticks]
\tzticks\{<x-ticks pos>\}{<y-ticks pos>}
\end{scope}
\end{tikzpicture}

The \texttt{\textbackslash tzticks} command prints tick labels and draws zero length tick marks, i.e. from \texttt{(0pt)} to \texttt{(0pt)}. There are three different syntaxes:

1. \texttt{\textbackslash tzticks}\{<x-ticks pos>\}{<y-ticks pos>}
2. \texttt{\textbackslash\textbackslash tzticks}\{<x-from:x-to>\}{<y-from:y-to>}[\{node opt\}]
3. \texttt{\textbackslash\textbackslash tzticks}\{\{x-shift,y-shift\}\{<x-from:x-to>\}{<y-from:y-to>}[\{node opt\}]

\textbf{Tick labels} Internally, \texttt{\textbackslash tzticks} uses \texttt{Ti\kern-.1667em kZ}'s \texttt{foreach} operation. So you need to provide comma separated lists to print tick labels. If only one comma separated list is specified, it is for \textit{x} tick labels.
You can change the numbered labels to a different format with slashes and other text, as follows: `<number>/<other text>`.

**Tick marks** By specifying the options `<x-from:x-to>` for $x$ ticks and/or `<y-from:y-to>` for $y$ ticks, you can print tick marks. (The default is (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 `[<node opt>]`.

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

```latex
\begin{tikzpicture}[scale=.4,font=\scriptsize]
\helplines\axes(-1,-1)(10,10)
\ticks[blue]{1/x_1,2,5/x_2\sqrt{x}}{2/\sqrt{x}}{3/y,4/m,5,7/k}[red]
\end{tikzpicture}
```

```latex
\begin{tikzpicture}[scale=.4,font=\scriptsize]
\helplines\axes(-1,-1)(10,10)
\ticks[draw=red,thick]{-15pt:10pt}{1,...,6,7/k,7/\alpha,8/\beta}[b=5pt]
\end{tikzpicture}
```
Shift  You can move (or shift) tick marks and labels together by specifying the option \texttt{<x-shift,y-shift>}, where \texttt{<x-shift>} is for \texttt{y-ticks} and \texttt{<y-shift>} is for \texttt{x-ticks}.

\begin{tikzpicture}[scale=0.4,font=\scriptsize]
\helplines(10,10)
\showorigin
\axes(-1,-1)(10,10)
\axes[dashed]<4,2>(-1,-1)(10,10)
\ticks[draw=red]<4,2>
\foreach \i in {5,...,8} {\tick\i}
\foreach \i in {3,...,7} {\tick\i}
\end{tikzpicture}

19.2 \texttt{\textbackslash tzticks*}: Tick marks

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

\begin{tikzpicture}[scale=0.4,font=\scriptsize]
\helplines(10,10)
\showorigin
\axes(-1,-1)(10,10)
\ticks*[draw=red,thick]
\foreach \i in {1,...,7,8/$\alpha$/} {\tick\i label ignored}
\foreach \i in {2,...,6,7/$\beta$/} {\tick\i label ignored}
\end{tikzpicture}
19.3 \texttt{\textbackslash{tz}ticksx(*) and \textbackslash{tz}ticksy(*)}

You can handle $x$ ticks and $y$ ticks independently.

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

\begin{verbatim}
% syntax
\textbackslash{tz}ticksx\{<opt>\}<y-shift>(<from>:<to>)\{<x-tick pos/labels>\} [<node opt>]
% defaults
[]<>(0pt:0pt)\{<m>\}[text height=1.25ex,text depth=.25ex,below]
\end{verbatim}

\texttt{\textbackslash{tz}ticksx*} only prints $x$-tick marks from 0pt to 3pt, by default, suppressing tick labels.

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

**Y ticks** \texttt{\textbackslash{tz}ticksy} only prints $y$-tick labels but not tick marks, by default. To prints tick marks you need to specify \texttt{(<x-from>:<x-to>)}.

\texttt{\textbackslash{tz}ticksy*} only prints $y$-ticks from 0pt to 3pt by default, suppressing tick labels.

\begin{verbatim}
% syntax:
\textbackslash{tz}ticksy\{<opt>\}<x-shift>(<from:to>)\{<y-ticks pos/labels>\} [<node opt>]
% defaults
[]<>(0pt:0pt)\{<m>\}[]
\end{verbatim}

\begin{verbatim}
% syntax
\textbackslash{tz}ticksy*[<opt>]<x-shift>(<from:to>)\{<yticks pos}\}
% defaults
[]<>(0pt:0pt)\{<m>\}
% starred(*) version suppresses tick labels
\end{verbatim}
Shift  The options \texttt{<y-shift>} and \texttt{<x-shift>} move x-ticks and y-ticks, respectively.

\begin{tikzpicture}[]
\scriptsize
\helplines (10,10)
\showorigin
\draw[red,thick] (-5pt:1cm) node[anchor=south]{$\alpha$} (1,...,7,8)
\draw[blue,thick] (0pt:3cm) node[anchor=east]{$\beta$} (2,...,6,7)
\end{tikzpicture}

20 Projections

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

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

\begin{tikzpicture}[]
\scriptsize
\draw[dotted] (0,0) node[anchor=south]{$\alpha$} (1,...,7,8)
\draw[blue,thick] (0,0) node[anchor=east]{$\beta$} (2,...,6,7)
\end{tikzpicture}

\tzproj* additionally prints a ‘black node dot’ of the size 2.4pt, by default. Internally, the node dot is processed by \texttt{\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 THREE WAYS on page 45. If you want to control fill or color of dots, use \texttt{\tzdot*} separately.
Adding text You can also add text around the projection point on each axis by the option `{<text>}`. The position and color of the text is controlled by the option `[<node option>]`. The default position is (approximately) [below] for the x axis and [left] for the y axis.

Projection shift Specifying the option `<x-shift,y-shift>` moves the projection point and text on each axis.

20.2 \texttt{\tzprojx(*)} and \texttt{\tzprojy(*)}

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

\texttt{\tzprojy} draws a dotted line, which is perpendicular to the y axis. \texttt{\tzprojy*} 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 (\texttt{\textless{}dot size\textgreater{}}). If you want to control fill or color of dots, use \texttt{\textbackslash{}tzdot*} separately. You can also add text around the projection point on each axis by specifying the option \{\texttt{x-text}\} or \{\texttt{y-text}\} followed by the option \{\texttt{node option}\}.

\begin{tikzpicture}[scale=.5,font=\scriptsize]
\tzhelplines(-1,-2)(6,6)
\tzshoworigin
\tzaxes(-1,-1)(6,6)
\tzproj*\{dashed\}(4,5){\{4\}}{\{5\}}{\{3pt\}}
\tzproj*[\textcolor{green}\text{green,thick,solid}]{\{3\}}{\{4\}}{\{\$x=3\$\}}{\{\textcolor{blue}\text{blue}\}}
\tzproj*[\textcolor{thick}\text{thick}]{\{5\}}{\{2\}}{\{5pt\}}
\end{tikzpicture}

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

\begin{tikzpicture}[scale=.5,font=\scriptsize]
\tzhelplines(-1,-2)(6,6)
\tzshoworigin
\tzaxes(-1,-1)(6,6)
\tzprojx*[\textcolor{green}\text{green,thick,solid}]{\{2\}}{\{1\}}{\{3\}}{\{4\}}{\{\$x=3\$\}}{\{\textcolor{blue}\text{blue}\}}
\tzprojy*[\textcolor{thick}\text{thick}]{\{2\}}{\{1\}}{\{5\}}{\{2\}}{\{5pt\}}
\end{tikzpicture}

20.3 \texttt{\textbackslash{}tzprojs(*)}: Semicolon versions

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

\begin{tikzpicture}[scale=.5,font=\scriptsize]
\tzhelplines(-1,-2)(6,6)
\tzshoworigin
\tzaxes(-1,-1)(6,6)
\tzprojx*[\textcolor{green}\text{green,thick,solid}]{\{2\}}{\{1\}}{\{3\}}{\{4\}}{\{\$x=3\$\}}{\{\textcolor{blue}\text{blue}\}}
\tzprojy*[\textcolor{thick}\text{thick}]{\{2\}}{\{1\}}{\{5\}}{\{2\}}{\{5pt\}}
\end{tikzpicture}

\texttt{\textbackslash{}tzprojs*} additionally prints \texttt{\textbackslash{}tzdots*} of the 2.4pt (by default) on the coordinates. The first option \{\texttt{\textbackslash{}opt}\} does not control the node dots.
You can move the projection points and text accordingly, using the option `<x-shift,y-shift>` before the first coordinate. The *empty* shift option `<>` is *not allowed*. You can also change the dot size with the last option `(dot size)` after the semicolon, as in \texttt{\texttt{tzproj}(*).}

\begin{figure}[h]
\centering
\begin{tikzpicture}
\node at (2,3) {$x$};
\node at (4,2) {$y$};
\node at (5,4) {$\alpha_1$};
\node at (6,5) {$\alpha_2$};
\draw[dashed, text=blue] (2,3) -- (5,4);
\draw[green] (2,3) -- (6,5);
\draw[red] (2,3) -- (7,6);
\end{tikzpicture}
\end{figure}

\subsection{\texttt{tzprojsx}(*), \texttt{tzprojsy}(*): Semicolon versions}

\texttt{tzprojsx} is a semicolon version of \texttt{tzprojx}. It draws dotted lines, which are perpendicular to the x axis from the specified coordinates, by default.

\texttt{tzprojsx*} additionally prints \texttt{tzdots*} of the size 2.4pt, by default.

\begin{figure}[h]
\centering
\begin{tikzpicture}
\node at (2,3) {$x$};
\node at (4,2) {$y$};
\node at (5,4) {$\alpha_1$};
\node at (6,5) {$\alpha_2$};
\draw[dashed, text=blue] (2,3) -- (5,4);
\draw[green] (2,3) -- (6,5);
\draw[red] (2,3) -- (7,6);
\end{tikzpicture}
\end{figure}

\texttt{tzprojsy} and \texttt{tzprojsy*} work similarly as \texttt{tzprojsx} and \texttt{tzprojsx*} do but to the y axis.
Specifying the option `<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 Syntax

`\tzfn` plots a function of `\x`.

\begin{verbatim}
\% syntax: minimum
\tzfn{<fn of \x>}[<domain>]
\% syntax: medium
\tzfn{<fn of \x>}{<domain>}{<text>}{<pos>}
\% syntax: full
\tzfn[<opt>]<shift coor>"<path name>"
  \{<fn of \x>}{<domain>}{<text>}{<node opt>}
% defaults
% [<domain>] should be of the form [<from num:to num>]
% [<from num:to num>] defaults [samples=201]<""{<\x>}{<\x>}{\{}{}\}
\end{verbatim}

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

\begin{verbatim}
\tzfn{.5*(\x)^2-1}[1:5] \% works like:
\draw [samples=201,domain=1:5] plot ({\x},{.5*(\x)^2-1});
\end{verbatim}
21.1.2 Inverse functions: `tzfn'`

The swap version `tzfn' draws the inverse function of `tzfn'.

\[ tzfn'(.5*(\langle\rangle^2-1)[1:5] \text{ works like:}\]
\draw [samples=201,domain=1:5] plot ({.5*(\langle\rangle^2-1)},\langle\rangle);

You can add text at the end of the graph, by specifying \{<text>\} and \{<node opt>\} immediately after the domain.

\[ tzfn'(0.5*(\langle\rangle^2-1)[1:5]
\text{inverse} \]
\draw (x,5) -- (x,7) node [right] {inverse demand};
\draw (5, y) -- (7, y) node [right] {demand inverse};
\draw (x,5) -- (x,7) node [right] {inverse demand};
\draw (5, y) -- (7, y) node [right] {demand inverse};

21.1.3 Define and name functions

To use `tzfn' you need to express a function as a function of `x'.

\[ tzfn
\begin{tikzpicture}[scale=.5]
\def\Dx(7-x)
\tzfn(\Dx)[0:7](5,5)
\tzfn[blue](5-5/7+x)[0:7]
\tzfticks(5,7)
\end{tikzpicture} \]
You can also use the predefined functions of Ti\text{\kern-.1667em Z} such as \texttt{sin}, \texttt{cos}, \texttt{ln}, \texttt{log10}, \texttt{log2}, \texttt{exp}, \texttt{sqrt}, and so on. (See Ti\text{\kern-.1667em Z} manual.)

\begin{tikzpicture}[scale=.5] 
\tzhelplines(-2,-2)(8,6) 
\tzaxes*(-2,-2)(8,6) 
\def\Fx{\sin(\x r)+3} \def\Gx{\exp(\x)} \def\Hx{\ln(\x)} \tzfn\Fx[-2:2*pi]{\sin \ x + 3} \[blue,r]\tzfn\Gx[-2:2]{\ e^x\ }[red,r]\tzfn[dashed]\Hx[.2:7]{\ln \ x}\[r]\end{tikzpicture} 

(See Ti\text{\kern-.1667em Z} manual.)

21.1.4 Name paths: name path

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

\begin{tikzpicture} 
\tzhelplines(8,8) \tzaxes(8,8) \def\Dx{7-\x} \def\Sx{1+\x} \tzfn[Dx=\Dx][0:7]{\{\textcolor{red}{D}\}}[ar] % name path = Dx \tzfn[Sx=\Sx][0:7]{\{\textcolor{blue}{S}\}}[blue,r] % name path = Sx \tzXpoint*[\Dx=Sx](E){E} % intersection \end{tikzpicture} 

\% \texttt{tzfn: name path: intersection point} \begin{tikzpicture}[scale=.5] \tzhelplines(8,8) \tzaxes(8,8) \def\Dx{7-\x} \def\Sx{1+\x} \tzfn[Dx=\Dx][0:7]{\{D\}}[ar] % name path = Dx \tzfn[Sx=\Sx][0:7]{\{S\}}[blue,r] % name path = Sx \tzXpoint*[\Dx=Sx](E){E} % intersection \end{tikzpicture} 

\textbf{Remark:} Advantage of defining functions: Suppose that the function’s expression \{\texttt{<fn of \ x>}\} consists only of a macro name, say, \texttt{\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{tzfn}\texttt{\Fx} is equivalent to \texttt{tzfn"Fx"\Fx}. (You don’t need to type the same thing twice.)

\begin{tikzpicture} \tzhelplines(8,8) \tzaxes(8,8) \def\Dx{7-\x} \def\Sx{1+\x} \tzfn[Dx=\Dx][0:7]{\{\textcolor{red}{D}\}}[ar] % name path = Dx \tzfn[Sx=\Sx][0:7]{\{\textcolor{blue}{S}\}}[blue,r] % name path = Sx \tzXpoint*[\Dx=Sx](E){E} % intersection \end{tikzpicture} 

21.1.5 Move graphs: shift

You can move the graph of \texttt{tzfn} 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.
21.1.6 Extend paths: `<code.append>`, \texttt{\textbackslash tzfnAtBegin}, \texttt{\textbackslash tzfnAtEnd}

You can extend the path of \texttt{\textbackslash tzfn}, by writing TikZ code in the last optional argument `<code.append>` and \texttt{[<node opt>]}.

\begin{tikzpicture}[scale=0.6]
\helplines(8,8)
\def\Dx{7-\x}
\def\Sx{1+\x}
\tzfn\Dx[0:7]{$D$}[right] % name path = Dx
\tzfn"supply"\Sx[0:7]{$S$}[right] % name path = supply
\Xpoint{Dx}{supply}(E)
\tzfn[dashed]<1,1>"demandA"\Dx[0:7]{\D'}[r]
\tzfn[dashed]<1,-1>"supplyA"\Sx[0:7]{\S'}[r]
\Xpoint{demandA}{supplyA}(E1)
\end{tikzpicture}

\texttt{\textbackslash tzfnAtEnd} You can also extend the path of \texttt{\textbackslash tzfn} at the end using \texttt{\textbackslash tzfnAtEnd}. Internally it adds TikZ code immediately before the options \texttt{[<text>]} and \texttt{[<node opt>]}. But you have to use \texttt{\textbackslash tzfnAtEnd} (immediately) before each \texttt{\textbackslash tzfn}.

\begin{tikzpicture}[scale=0.5]
\helplines(-2,-2)(8,6)
\axes(-2,-2)(8,6)
\def\Fx{\sin(\x r)+3}
\tzfn[->]\Fx[-2:2*pi]{$\sin x + 3$}[blue,r]
\tzfnAtEnd[to [bend right] ++(2,-2) node [b] {End!}]\Fx[-2:2*pi]{$\sin x + 3$}[blue,r]
\end{tikzpicture}

Specifying the option `<code.append>` extends the path after \texttt{\textbackslash tzfnEnd} if it exists.

\begin{tikzpicture}[scale=0.5]
\helplines(-2,-2)(8,6)
\axes(-2,-2)(8,6)
\def\Fx{\sin(\x r)+3}
\tzfnEnd[to [bend right] ++(2,-2) node [b] {End!}]\Fx[-2:2*pi]{$\sin x + 3$}[blue,r]
\tzfnAtEnd[to [bend right] ++(2,-2) node [b] {End!}]\Fx[-2:2*pi]{$\sin x + 3$}[blue,r]
\end{tikzpicture}
You can use `\tzfnAtBegin` (immediately) before each `\tzfn` to insert TikZ code at the beginning of the path of `\tzfn`.

```
% \tzfnAtBegin (before \tzfn)
\begin{tikzpicture}[scale=.5]
\tzhelplines(-2,-2)(8,6)
\tzaxes*(-2,-2)(8,6)
\def\Fx{\sin(\x r)+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 `|-`) 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 $y$.

```
% syntax: minimum
\tzfnofy{<fn of \y>}[<domain>]
% syntax: medium
\tzfnofy{<fn of \y>}[<domain>]{<text>}[<pos>]
% syntax: full
\tzfnofy[<opt>]<shift coor>"<path name>"
% [<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});
You can also extend the path of \texttt{tzfnoy} using the option \texttt{code.append} or the macros \texttt{tzfnoyAtBegin} and \texttt{tzfnoyAtEnd}.

\begin{verbatim}
\% \texttt{tzfnoy: shift: intersections}
\begin{tikzpicture}[scale=.6]
\def\Fy{7-\y}
\def\Gy{1+\y}
\tzfnofy\Fy[0:7][right]\ % name path = Fy
\tzfnofy\Gy[0:7][right]\ % name path = Gy
\tzXpoint*{\Fy}{\Gy}(E){E}\ % name path = E
\tzfnofy[dashed]<1,1>"Fyy\Fy[0:7]{shifted}[r]
\tzfnofy[dashed]<1,-1>"Gyy\Gy[0:7]{shifted}[r]
\tzXpoint*{\Fyy}{\Gyy}(E1){E'}
\end{tikzpicture}
\end{verbatim}

\section{Horizontal lines}

\subsection{\texttt{tzfnat}}
\texttt{tzfnat} draws a horizontal line at a specified value of $y$.

\begin{verbatim}
\% syntax: minimal
\tzfnat{<y-val>}{<domain>}
\% syntax: full
\tzfnat{<opt>}{<shift coor>"<path name>"
  \{<y-val>\}{<domain>}{<text>}\{<node opt>\}<code.append>
\% defaults
{}\{\texttt{\{west:east (of current bounding box)\}}\}
\end{verbatim}
\texttt{tzfnat} accepts only one mandatory argument \{<y-val>\}. The domain is optional and should be of the form \{<from num:to num>\}. The default domain is from left to right of the current bounding box.

Remark: Internally, the default domain of \texttt{tzfnat} depends on the current bounding box.

- Each \texttt{tzfnat} may draw a line with a (slightly) different length.
- If an appropriate current bounding box is not formed before \texttt{tzfnat} is executed, you will probably get an unexpected result.
- In that case, you can fix a bounding box in the beginning of the \texttt{tikzpicture} environment using macros such as \texttt{tzbbox}, \texttt{tzhelplines*}, \texttt{tzaxes*}, or TikZ’s \texttt{useasboundingbox}.

You can name the path of \texttt{tzfnat} 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}
\tzhelplines(4,3)
\tzfnat[blue,thick]{0}
\tzfnat[red,thick]{1}[1:3]{line A}[r]
\tzfnat[blue]{2}{line B}[r]
\tzfnat[red]{3}[0:3]{line C}[draw=blue,red,r]
\end{tikzpicture}

\begin{tikzpicture}
\tzhelplines*(4,4) % bounding box
\tzfn"XX"\{\x\}[0:4]
\tzfnat[blue,thick]{0}
\tzfnat[red,thick]<-.5,.5>"AA"{1}[1:3]{line A'}[r]
\tzxpoint*{AA}{XX}
\tzfnat[blue,->]{2}{line B}[r]
< arc (0:120:1.5) node [b,draw] {End!} >
\end{tikzpicture}

You can also use \texttt{tzfnatAtBegin} and \texttt{tzfnatAtEnd} to extend the path of \texttt{tzfnat} at the beginning and at the end. Specifying the option \{<code.append>\} extends the path after \texttt{tzfnatAtEnd} if it exists. (See other examples of using \texttt{tz<..>AtBegin} and \texttt{tz<...>AtEnd}.)

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

\textit{Everything else is the same as in \texttt{tzfnat}.}
You can also use \texttt{\tzhfnAtBegin} and \texttt{\tzhfnAtEnd} to extend the path of \texttt{\tzhfn} at the beginning and at the end. Specifying the option \texttt{<code.append>} extends the path after \texttt{\tzhfnAtEnd} if it exists. (See other examples of using \texttt{\tz<...>AtBegin} and \texttt{\tz<...>AtEnd}.)

21.4 Vertical lines

21.4.1 \texttt{\tzvfnat}

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

You can name the path of \texttt{\tzvfnat} by the option \texttt{"<path name>"}. You can move the line by the option \texttt{<shift coor>}. You can also extend the path from the end of the line by writing TikZ code in the last optional argument \texttt{<code.append>}.
In the previous example, `\tzhelplines*` is used to fix a bounding box. (See Section 7.2 on page 39, for more details.)

You can also use `\tzvfnAtBegin` and `\tzvfnAtEnd` to extend the path of `\tzvfn` at the beginning and at the end. Specifying the option `<code.append>` extends the path after `\tzvfnAtEnd` if it exists. (See other examples of using `\tz<..>AtBegin` and `\tz<...>AtEnd`.)

### 21.4.2 `\tzvfn`

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

Everything else is the same as in `\tzvfnat`.

---

#### 22 Plot Linear Functions

### 22.1 `\tLFn`: Plot linear functions

#### 22.1.1 `\tLFn` and `\tLFn'`

Knowing two coordinates or one coordinate with a slope, you can draw a linear function with the macro `\tLFn`, without writing the explicit definition of a linear function.

- `\tLFn(<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{tzLFnoy}(x_1,x_2)(y_1,y_2) draws the graph of \(f(x) = \frac{y_2-y_1}{x_2-x_1}(x-x_1) + y_1\).

- \texttt{tzLFn(<coor1>){<slope>}} is prepared for when you know one coordinate and the slope of a line.
- If you specify all the three arguments \(<\texttt{coor1}>\)(<coor2>){<slope>}, 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.

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.

\texttt{tzLFn'} : inverse function \texttt{tzLFn} draws the inverse function of \texttt{tzLFn}.

**Remark:** If you inadvertently try an infinite slope, you will get an error message.

**Inverse function** The swap version \texttt{tzLFn'} draws the inverse function of \texttt{tzLFn}.
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}[scale=.8]
\zhelplines\(4,4\)
\tzcoors*(1,1)(A)(1,2)(B)(3,1)(C);
\tZLFn[blue]"Gx"(B)(C)[0:4]{$g(x)$}[r]
\tZLFn[<arc>\(0:140:2\) node \[below\] \{End!\} >
\end{tikzpicture}
\texttt{tZLFn} at \texttt{Begin} and \texttt{tZLFn} at \texttt{End} 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{tZLFn} at \texttt{End}, if it exist.\%
\begin{tikzpicture}[scale=.8]
\zhelplines\(4,4\)
\tzcoors*(1,1)(A)(1,2)(B)(3,1)(C);
\tZLFn[blue]"Gx"(B)(C)[0:4]{$g(x)$}[r]
\tZLFnAtBegin{(1,0) to [bend left] \(1,1\)}
\tZLFnAtEnd{\(arc \(0:140:2\)\)}
\end{tikzpicture}\%
\subsection{tZLFn}ofy and tZLFn}ofy′
\texttt{tZLFn}ofy draws a line as a function of \(y\). \texttt{tZLFn}ofy works just like \texttt{tZLFn} but for the variable \(y\). If you provide two points \((x_1, y_1)\) and \((x_2, y_2)\), \texttt{tZLFn}ofy\((x_1,x_2)(y_1,y_2)\) draws the graph of
\[
    f(y) = \frac{x_2 - x_1}{y_2 - y_1} (y - y_1) + x_1.
\]
Everything else is the same as \texttt{tZLFn}.
\%
\begin{tikzpicture}[scale=.8]
\zhelplines\(4,4\)
\tzcoors*(1,1)(A)(1,2)(B)(3,1)(C);
\tZLFn[blue]"Gx"(B)(C)[0:4]{$g(x)$}[r]
\tZLPPoint*{Fx}{Gx}{E}{E}(3pt)
\end{tikzpicture}\%
\textbf{Remark:} If you inadvertently try an infinite slope, you will get an error message.
Inverse function  The swap version \texttt{\tzLfnofy'} draws the inverse function of \texttt{\tzLfnofy}.

\begin{tikzpicture}[scale=.8]
\tzhelplines(4,4)
\tzcoors*(1,1)(A)(1,2)(B)(3,1)(C);
\tzLfnofy'[red]"Fx"(A){.5}{y}[a]
\tzLfnofy'[blue]"Gx"(B)(C){.5:2.5}{y}[r]
\tzXpoint*[fill=none]"Fx"{Gx}(E){3pt}
\end{tikzpicture}

You can use \texttt{\tzLfnofyAtBegin} and \texttt{\tzLfnofyAtEnd} to extend the path of \texttt{\tzLfnofy} at the beginning and at the end. Specifying the optional argument \texttt{<code.append>} extends the path after \texttt{\tzLfnofyAtEnd} if it exists. (See other examples of using \texttt{\tz<...>AtBegin} and \texttt{\tz<...>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, without writing an explicit definition of a linear function. (Of course, you can directly use \texttt{\tzLfn}.)

\begin{tikzpicture}[scale=.8]
\tzhelplines(4,4)
\tzaxes(4,4)\node\(x\)\node\(y\)
\tzticks{1,2,3,4}{1,2,3,4}
\tzcoors*(1,1)(A)\node\(A\)(-90)(1,2)(B)\node\(B\)(3,1)(C)\node\(C\)\(45\);
\tzdefLFn\Fx(A){.5}
\tzdefLFn\Gx(B)(C)
\tzfn[red]\Fx{0:4}{x}{y}[r]
\tzfn[blue]\Gx{0:4}{y}[r]
\end{tikzpicture}

Remark: If you inadvertently try an infinite slope, you will get an error message.

\begin{tikzpicture}[scale=.8]
\tzhelplines(4,4)
\tzaxes(4,4){x}{y}
\tzticks{1,2,3,4}{1,2,3,4}
\tzcoors*(1,1)(A){-90}(1,2)(B)(3,1)(C){45};
\tzdefLFn\Fx(A){.5}
\tzdefLFn\Gx(B)(C)
\tzfn[red]\Fx{0:4}x{y}[r]
\tzfn[blue]\Gx{0:4}y[r]
\end{tikzpicture}

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 (A) or (B).
\begin{tikzpicture}
\helplines (4,4)
\axes (4,4) {$x$} {$y$}
\ticks {1,2,3,4} {1,2,3,4}
\coors* (1,1) (A)
\coors* (1,2) (B)
\coors* (3,1) (C)
\coors* (1,2) (D)
\coors* (4,1) (E)
\def\Fx{(A)}{.5}
\def\Gx{(B),(C)}{.5}
\fn{red}{\Fx[0:4]}{$f(x)$}[x]
\fn{blue}{\Gx[0:4]}{$g(x)$}[x]
\end{tikzpicture}

\subsection{\texttt{\textsc{tzdefLFnofy}}}

\texttt{\textsc{tzdefLFnofy}} defines a function of \texttt{y}.

\begin{verbatim}
% syntax
\texttt{\textsc{tzdefLFnofy}} (<fn csname>) (<coor1>) (<coor2>) (<slope>)
% defaults
{} (<m>) { (<m>) } () (1)
\end{verbatim}

If (<coor2>) is specified, (<slope>) is ignored. If (<coor2>) is missing \texttt{<slope>} is considered as the slope of the line (by default 1).

\begin{verbatim}
\texttt{\textsc{tzdefLFnofy}} {Fy}(1,1) {.5} % works like
\texttt{\textsc{def}} \texttt{Fy} {.5* (y-1) + 1}

\texttt{\textsc{tzdefLFnofy}} {Gy}(1,2)(3,1) % works like
\texttt{\textsc{def}} \texttt{Gy} {-2*(y-2) + 1}
\end{verbatim}

\textbf{Remark:} If you inadvertently try an infinite slope, you will get an error message.

You can use \texttt{\textsc{tzdefLFnofy}} together with \texttt{\textsc{tzFnofy}} to graph a linear function of \texttt{y}. (Of course, you can directly use \texttt{\textsc{tzLFnofy}}.)

\begin{verbatim}
% \texttt{\textsc{tzdefLFnofy}} and \texttt{\textsc{tzLFnofy}}
\begin{tikzpicture}
\helplines(4,4)
\axes*(4,4) {$x$} {$y$}
\ticks {1,2,3,4} {1,2,3,4}
\coors* (1,1) (A)
\coors* (1,2) (B)
\coors* (3,1) (C)
\coors* (4,1) (D)
\coors* (1,2) (E)
\def\Fx{Fy}(A) {.5}
\def\Gx{Gy}(B) {C}
\def\Fy{red}{Fy}[0:4]{$f(y)$}[x]
\def\Gy{blue}{Gy}[0:4]{$g(y)$}[a]
\end{tikzpicture}
\end{verbatim}

\textbf{Remark:} The swap version \texttt{\textsc{tzfnofy}'} simply draws the graph of the inverse function of \texttt{\textsc{tzfnofy}}. So \texttt{\textsc{tzfnofy}'} (A) (B) and \texttt{\textsc{tzfnofy}'} (A) {.5} do not guarantee passing through the coordinate (A) or (B).
23 Some More Functions

23.1 \tzpdfN(*) and \tzpdfZ: Normal distributions

\tzpdfN, \tzpdfN* and \tzpdfZ are predefined functions to plot the probability density function (pdf) of a normal distribution \( N(\mu, \sigma^2) \).

Normal distributions \( \tzpdfN \) accepts two mandatory arguments, \(<\text{mean}>\) \( \mu \) and \(<\text{variance}>\) \( \sigma^2 \), to define the pdf function:

\[
\frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{x-\mu}{\sigma} \right)^2}
\]

\( \tzpdfN* \) uses a standard deviation \( \sigma \) instead of a variance.

With these predefined functions together with \tzfn you can plot the pdf of normal distributions.

Remark: If \tikzpicture is scaled when plotting the inverse function using \tzfn', you may want to change the scale accordingly.
Standard normal distributions \texttt{\textbackslash \textpdfZ} (with no arguments) represents the pdf of the standard normal distribution.

23.2 \texttt{\textfnarea(*)}: Fill under the graph

\texttt{\textfnarea(*)} 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{\textsetfnarealayer}, you can change the layer of \texttt{\textfnarea}. \texttt{\textfnarea} accepts two mandatory arguments: \{\texttt{<fn of \textbackslash x>}\} and \{\texttt{<domain>}\}, like \texttt{\textfn}.
The starred version \texttt{\textbackslash 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{\textbackslash settzfillcolor}, \texttt{\textbackslash settzfillopacity}, and the option \texttt{\{<fill opacity>\}}.

\begin{tikzpicture}
\begin{scope}
\begin{scope}
\fill[red] plot[domain=1:3, smooth, variable=t] (t,{t^2-1}) \closedcycle;
\end{scope}
\end{scope}
\end{tikzpicture}

\begin{tikzpicture}[\scriptsize]
\draw[help lines] (-4,-.1) grid (6,.5);
\draw[<->] (0,0) -- (0,4.5) node[above] {$y$};
\draw[<->] (-4,0) -- (4.5,0) node[right] {$x$};
\draw[blue] plot[domain=-3:3, smooth, variable=t] (t, {t^2-1}) node[above right] {$f(x)$};
\fill[red] (1,0) -- (3,0) -- (3,1) -- (1,1) -- cycle;
\fill[gray!50] (1,0) -- (3,0) -- (3,-1) -- (1,-1) -- cycle;
\fill[pattern=north east lines] (1,-1) -- (3,-1) -- (3,1) -- (1,1) -- cycle;
\node[black] at (2,-1) {Area};
\node[black] at (2,3) {Area};
\end{tikzpicture}

\begin{tikzpicture}
\begin{scope}
\settzfnarealayer{main}
\fill[red] plot[domain=1:3, smooth, variable=t] (t,{t^2-1}) \closedcycle;
\end{scope}
\end{tikzpicture}

\begin{tikzpicture}
\begin{scope}
\draw[help lines] (0,-2) grid (4,3);
\draw[<->] (0,0) -- (0,3.5) node[above] {$y$};
\draw[<->] (-4,0) -- (4.5,0) node[right] {$x$};
\draw[blue, thick] plot[domain=-3:3, smooth, variable=t] (t, {t^2-1}) node[above right] {$f(x)$};
\draw[very thick] (1,0) -- (1,1);
\draw[very thick] (3,0) -- (3,1);
\end{scope}
\end{tikzpicture}

23.2.2 \texttt{\textbackslash tzfnarealine(\textbackslash ')}

\texttt{\textbackslash tzfnarealine} draws one or two boundary lines of \texttt{\textbackslash tzfnarea} using \texttt{\textbackslash tzto}, on the \texttt{\textbackslash behind} layer by default. The layer can be changed by \texttt{\textbackslash settzfnarealayer}. It takes two mandatory arguments: \{\texttt{<path>}\} and \{\texttt{<x1>}\).

- \texttt{\textbackslash tzfnarealine\{<path>\}\{<x1>\}} draws a vertical line to the \texttt{x} axis at \texttt{<x1>}. 

163
• \texttt{\textbackslash{}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 \texttt{very thin} by default, which can be changed by \texttt{\settzfnarealinewith}.

You can change the length of lines by specifying \texttt{(<coor1>)} and \texttt{(<coor2>)}.

The \texttt{swap version} \texttt{\textbackslash{}tzfnarealine'} draws one or two horizontal lines from the point \texttt{\((x_i,f(x_i))\)} to the \texttt{y} axis. Everything else is the same as in \texttt{\textbackslash{}tzfnarealine}.
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}.

**Remark:** If you want sharp corners at kinked points, you may need to select an appropriate number of sample points, which is \texttt{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 \texttt{<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.
23.3.2 \texttt{\texttt{tzfnmin}}: Lower envelope curves

\texttt{\texttt{tzfnmin}} plots the minimum of a list of functions. \texttt{\texttt{tzfnmin}} works just like \texttt{\texttt{tzfnmax}}, but it draws lower a envelope curve.

\begin{verbatim}
% syntax: minimum
\tzfnmin{<fn list>}[<domain>]
% syntax: full
\tzfnmin[<opt>]<shift coor>"<path name>"
  {<fn list>}{<domain>}{<text>}{<node opt>}
% remark:
- [<domain>] should be comma separated
- [<from num:to num>] should be of the form [<from num:to num>]
% defaults
[thick,samples=201]<><>
\end{verbatim}

\begin{verbatim}
% syntax: minimum
\tzfnmin{<fn list>}[<domain>]
% syntax: full
\tzfnmin[<opt>]<shift coor>"<path name>"
  {<fn list>}{<domain>}{<text>}{<node opt>}
% remark:
- [<domain>] should be comma separated
- [<from num:to num>] 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 appropriate number of sample points. Try odd number.

To extend the path of \texttt{\texttt{tzfnmin}}, you can use the option \texttt{\texttt{\texttt{code.append}}}, or the macros \texttt{\texttt{\texttt{tzfnminAtBegin}}} and \texttt{\texttt{\texttt{\texttt{tzfnminAtEnd}}}}.
24 Intersections

24.1 \tzXpoint(*): Intersection points

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

\begin{tikzpicture}[scale=.5]
\zhelplines(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 (<dot size>) or the Three Ways (on page 45).
24.2 Vertical intersection points

24.2.1 \tZvXpointat(*)

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

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

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

24.2.2 \tzvXpoint(*)

\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, \tzvXpoint(mypath){3,<y>}, ignoring <y>, is equivalent to \tzvXpointat{mypath}{3}.

*Everything else is the same as in \tzvXpointat.*

\begin{tikzpicture}
% \tzvXpoint(*)
\def\Fx{.5*(x-2)^2}
\def\Fx[0:4] % name path=Fx
\tzfn[Fx][1]
\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}

\subsection*{24.2.2 \tzvXpoint(*)}

\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, \tzvXpoint(mypath){3,<y>}, ignoring <y>, is equivalent to \tzvXpointat{mypath}{3}.

*Everything else is the same as in \tzvXpointat.*

The starred version \tzvXpoint* just adds a node dot to \tzvXpoint.

\begin{tikzpicture}
% \tzvXpoint(*)
\def\Fx{.5*(x-2)^2}
\def\Fx[0:4] % name path=Fx
\tzfn[Fx][1]
\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}

\subsection*{24.3 Horizontal intersection points}

\subsection*{24.3.1 \tzXpointat(*)}

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

\begin{tikzpicture}
\def\Fx{.5*(x-2)^2}
\draw\Fx[0:4] \% name path=Fx
\path[Fx](1)(A)
\draw[dashed](1.5)
\path[Fx](1.5)(A)[45]
\path*(X-2)(Y)[red,draw=al] \% abb
\end{tikzpicture}

24.3.2 `\tzhXpoint(*)`

`\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, `\tzhXpoint{mypath}{{x},3}`, ignoring `<x>`, is equivalent to `\tzhXpointat{mypath}{3}`.

Everything else is the same as in `\tzhXpointat`.

The starred version `\tzhXpoint*` just adds a node dot to `\tzhXpoint`.
\tzhXpoint* prints, at the (first) intersection point, a node dot of the size 2.4pt, by default.

% \tzhXpoint(*)
\begin{tikzpicture}
\helplines*(4,3)
\def\Fx{.5*(x-2)^2}
\xFx[0:4] \% name path=Fx
\coors(0,1)(A)(0,1.5)(B);
\xFx(A)(A)
\dot(A){A}[0]
\xFxdashed(0,1.5)
\xFx*(Fx)(X)[45]
\dot+(X-2)(Y)[red,draw al ] \% abb
\end{tikzpicture}

24.4 \tzLFnXpoint(*): Intersection point of linear functions

Sometimes you may want to quickly find intersection point of two linear functions without specifying path names.
\tzLFnXpoint finds the solution of two linear functions without printing anything by default. You can name it and use it.

% syntax: minimum
\zLFnXpoint(<fn>{<fn}>)
% syntax: medium
\zLFnXpoint(<fn>{<fn>}{<coor name}>{<label>}{<label opt>}{angle}]
% syntax: full
\zLFnXpoint<opt>{<fn>}{<fn>}{<coor name>}
{<label>}{<label opt>}{angle}(<dot size>)
% defaults
[tzdot=2.4pt,draw=none,minumum size=0pt]<m>{m}[0](){0}(2.4pt)

\begin{tikzpicture}[scale=.8]
\helplines(4,3)
\xFyx{y}{x}[0:3]
\xFy{3-x}{x}[0:3]
\zLFnXpoint{x}{3-x}(A) \% invisible
\zdot+->[bend left](A)(2,-1)
\end{tikzpicture}

\zLFnXpoint* additionally print a filled node dot at the solution of two linear equations.

% syntax: minimum
\zLFnXpoint(<fn>{<fn>})
% syntax: medium
\zLFnXpoint(<fn>{<fn>}{<coor name>{<label>}{<label opt>}{angle}]
% syntax: full
\zLFnXpoint<opt>{<fn>}{<fn>}{<coor name>}
{<label>}{<label opt>}{angle}(<dot size>)
% defaults
[tzdot=2.4pt,fill]<m>{m}[0](){0}(2.4pt)
25 Secant and Tangent Lines

25.1 \texttt{\textbackslash tzsecant}

\texttt{\textbackslash tzsecant} draws a line segment or a 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 $x$: \{\texttt{<path>}, \texttt{<from-x>}, and \texttt{<to-x>}\}. With \texttt{\textbackslash settzsecantlayer}, you can change the layer, like \texttt{\textbackslash settzsecantlayer{main}}.

% syntax: minimum
\texttt{\textbackslash tzsecant}{<path>}{<from-x>}{<to-x>}
% syntax: medium
\texttt{\textbackslash tzsecant}{<path>}{<from-x>}{<to-x>}[<domain>][<text>][<node opt>]
% syntax: full
\texttt{\textbackslash tzsecant} [<opt>]<shift coor>"path name"
\texttt{\textbackslash tzsecant}{<path>}{<from-x>}{<to-x>}[<domain>][<text>][<node opt>]<code.append>
% defaults
[]<>=""{<m1>}{<m2>}{<m3>}[]{}[]>

\textbf{Domain} The domain of the form \{\texttt{<from num:to num>}\} is optional. Without specifying the optional domain, \texttt{\textbackslash tzsecant} 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 tzsecant}.
**Shift**

You can move \texttt{\textbackslash tzsecantat} by specifying the option \texttt{<shift coor>}.  

\begin{quote}% \texttt{\textbackslash tzsecant: domain} 
\begin{tikzpicture}[scale=.5] 
\zhelplines(8,6) \tzaxes(-1,-1)(8,6)(\$x\$)(\$y\$) \tbeziers+"curve"(.5,1)(1,6)(-1,-4)(7,5){\texttt{[ar]}} \tzsecant[\blue,\{curve\}{\{1\}}{\texttt{\{shift\}}}{0:5}{\texttt{[a]}}] \tzsecant[\blue,\{curve\}{\{1\}}{\{shift\}}{0:5}{\texttt{[r]}}] \end{tikzpicture} 
\end{quote}

**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{quote}% \texttt{\textbackslash tzsecant: shift, name path (intersection)} 
\begin{tikzpicture}[scale=.5] 
\zhelplines(8,6) \tzaxes(-1,-1)(8,6)(\$x\$)(\$y\$) \tbeziers+"curve"(.5,1)(1,6)(-1,-4)(7,5){\texttt{[ar]}} \tzsecant[\blue,\{curve\}{\{1\}}{\{shift\}}{0:5}{\texttt{[a]}}] \tzsecant[\blue,\{curve\}{\{1\}}{\{shift\}}{0:5}{\texttt{[r]}}] \tzvXpointat*{\texttt{\textbackslash setztzsecantlayer}}{3}{X}{-45} \end{tikzpicture} 
\end{quote}

\texttt{<code.append>} You can extend the path of \texttt{\textbackslash tzsecantat} by writing Ti\textit{K}Z code in the last optional argument \texttt{<code.append>}.  

\begin{quote}% \texttt{\textbackslash tzsecant: <code.append>} 
\begin{tikzpicture}[scale=.5] 
\zhelplines(8,6) \tzaxes(-1,-1)(8,6)(\$x\$)(\$y\$) \tbeziers+"curve"(.5,1)(1,6)(-1,-4)(7,5){\texttt{[ar]}} \tzsecant[\blue,\{curve\}{\{1\}}{\{shift\}}{0:5}{\texttt{[a]}}] \tzsecant[\blue,\{curve\}{\{1\}}{\{shift\}}{0:5}{\texttt{[r]}}] \tzvXpointat*{\texttt{\textbackslash setztzsecantlayer}}{3}{X}{-45} \end{tikzpicture} 
\end{quote}

\subsection{\texttt{\textbackslash tzsecant}}

\texttt{\textbackslash tzsecant} uses two \textit{coordinates} instead of two values of \textit{x} to draw a line segment or a \textit{secant} line of a curve, on the \texttt{behind} layer by default. You need to specify a path name and two coordinates, then \texttt{\textbackslash tzsecant} uses the \textit{x} values of the two coordinates.

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

You can change the layer with \texttt{\setztzsecantlayer}.

\begin{quote}  
\begin{itemize}  
\item % syntax: minimal  
\texttt{\textbackslash tzsecant(<path>)(<coor>)(<coor>)}  
\item % syntax: medium  
\end{itemize}  
\end{quote}
The domain should be of the form \([\text{from num:to num}]\). Without specifying the optional domain, \texttt{tzsecant} draws a line segment connecting two points on the (curved) path.

With the last option \texttt{<code.append>} you can extend the path of \texttt{tzsecant}.

### 25.2 Tangent lines

#### 25.2.1 \texttt{tzttangentat}

\texttt{tzttangentat} draws a tangent line to a curve at a specified value of \(x\). Three mandatory arguments are a curve name, a value of \(x\), and a domain: \{<path>, \{x-val\}, and \{domain\}. By default, the tangent line is drawn on the behind layer, which can be changed by \texttt{setztztangentlayer}, like \texttt{setztztangentlayer(main)}.

**Remark:** To calculate the slope at \(x\), \(x\) varies over the interval \((x-\varepsilon_1, x+\varepsilon_2)\) and \(\varepsilon_1 = \varepsilon_2 = 0.01\), by default. So the slope of tangent line is only approximate.
The domain should be of the form \([<from:to>]\). 

### Domain

The mandatory argument \([<domain>]\) should be of the form \([<from\ num:to\ num>]\).

```latex
% \ttangentat
\begin{tikzpicture}[scale=.5]
\tzhelplines(9,8)
\tzaxes(9,8)
\tzplotcurve"AA"(1,7)(3,3)(8,1);
\ttangentat{AA}{2}[.5:4]
\ttangentat[blue]{AA}{4}[1:7]{tangent}[red,b]
\tzvXpointat*{AA}{4}
\tztickx(1,7)
\end{tikzpicture}
```

### Shift

You can move the tangent line by specifying the option \(<shift\ coor>\).

```latex
% \ttangentat: shift
\begin{tikzpicture}[scale=.5]
\tzhelplines(9,8)
\tzaxes(9,8)
\tzplotcurve"AA"(1,7)(3,3)(8,1);
\ttangentat{AA}{2}[.5:4]
\ttangentat[blue]{AA}{4}[1:7]{tangent}[red,b]
\ttangentat[blue]<2,1>{AA}{4}[1:7]{tangent'}[r]
\tzvXpointat*{AA}{4}
\tztickx(1,7)
\end{tikzpicture}
```

### Naming paths

By specifying the option \("<path\ name>\"\), you can name the path of \ttangentat and use it to find intersection points.

```latex
% \ttangentat: name path (intersection)
\begin{tikzpicture}[scale=.5,font=\footnotesize]
\tzhelplines(9,8)
\tzaxes(9,8)
\tzplotcurve"AA"(1,7)(3,3)(8,1);
\ttangentat{AA}{2}[.5:4]
\ttangentat[blue]{AA}{4}[1:7]{tangent}[red,b]
\tzvXpointat*{AA}{4}
\ttangentat[blue]<2,1>"BB"{AA}{4}[1:7]{tangent'}[r]
\tzvXpointat*{"BB"}{4}[X]
\tzproj(X)(x)\(x\_1\)(x)\(x\_2\)
\end{tikzpicture}
```

\textit{<code.append>} You can extend the path of \ttangentat by writing TikZ code in the last optional argument \texttt{<code.append>}. 

175
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 $(\varepsilon_{1,2})$ immediately after the mandatory argument $(x\text{-val})$. 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: {<path>}, (coor), and [domain].

The value of $y$ in (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.
You can shift the tangent line and extend its path.

You can control the interval of variations of $x$ by the option ($\epsilon_1, \epsilon_2$) or the macro \settztangentepsilon.

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 \settztzlopelayer{main}.

Remark: The slope is approximate and you can manipulate the slope 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 \tzslopeepsilon{<epsilon1>}{<epsilon2>} immediately after the option \{<x-val>\}, which is valid until the end of the \tikzpicture environment, unless changed again.
% syntax: minimum
\tzslopeat\{<path>\}{<x-val>\}{<length>}
% syntax: minimum (for normal line)
\tzslopeat\{<path>\}{<x-val>\}{<length>}[90] % or [-90]
% syntax: full
\tzslopeat\[<opt>\]{<path>\}{<x-val>\}{<length>\}{<rotate>\}{<text>\}{<node opt>\}<code.append
% defaults
[]\{<m>\}{<m>\}{.01,.01\}{<m>\}{0\}{<>}

% \tzslopeat
\begin{tikzpicture}
\tzhelplines\*\(5,4\)
\tzcoors*\(0,0\)(A)(2,2)(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]
\end{tikzpicture}

% \tzslopeat: repeated
\begin{tikzpicture}
\tzhelplines\(1,1\)(5,4)
\tzcoors*(0,0)(A\{2,2\}\{b\}(3.5,1)(C)(5,3\{D\);
\tzparabola"AA"(A)(B)(C)
\foreach \x in \(1,1.5,...,3.5\)
\{ \tzslopeat[thick,red]\{AA\\}{\x\}{2cm} \}
\end{tikzpicture}

Normal lines  With the option \[<rotate>\] immediately after the last mandatory option \{<length>\}, you can draw a normal line to a curve by rotating the slope lines 90° or -90°.
25.3.2 \tzslope

\tzslope is the same as \tzslopeat, except for one thing. \tzslope uses a coordinate instead of a value of $x$ to draw slope lines. So the mandatory arguments of \tzslope are {<path>}, {{<coor>}}, and {<length>}. The $y$ value of $<x,y>$ is ignored.

% syntax
\tzslope[<opt>]{<path>}(<x,y>)({<epsilon1>},{<epsilon2>}){<length>}[<angle>]
{<text>}{<pos,opt>}{<code.append>}

% defaults
[]{(m)}{(0.01,0.01)}{(m)}[0][0]

You can add Ti\hbox{\kern-0.2em\textsc{k}Z} code with the last option <code.append>.
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}
\zhelplines(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}
\zhelplines(4,3)
\settzmidarrow<.25>{Stealth[reversed]}[red,scale=2]
\tzline[-->--](0,3)(4,3)
\settzmidarrow{Circle[open]}[blue,scale=2]
\tzline[-->--](0,2.5)(4,2.5)
\settzmidarrow{Rays}[blue,scale=2]
\tzline[-->--->,out=45](0,0)(4,1)
\end{tikzpicture}

You can use various styles of arrow tips. (See TikZ 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:

```latex
\draw [\{<opt>\}] (0,0) circle (1.2pt) ;
```

\settzmidarrow controls the circle with \{<opt>\}.

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:

```latex
\draw [\{<opt>\}] (2pt,2pt) to (-2pt,-2pt) ;
\draw [\{<opt>\}] (-2pt,2pt) to (2pt,-2pt) ;
```

\settzmidarrow controls the circle with \{<opt>\}. 

---

181
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:

\begin{tikzpicture}
\tzhelplines(4,3)
\tzline[--/--,thick](0,3)(4,3)
\tzline[--/--,.5,blue,thick](0,2.5)(4,2.5)
\tzline[--/--,blue,thick](0,2)(4,2)
\settzmidarrow[thick,blue,tzextend={8pt}{8pt}]
\tzto[--/--,.75,->,out=45](0,0)(4,1)
\end{tikzpicture}

\begin{tikzpicture}
\tzhelplines(4,3)
\tzline[--/--,thick](0,3)(4,3)
\tzline[--/--,.25,blue,thick](0,2.5)(4,2.5)
\settzmidarrow[rotate=90,red,tzextend={3pt}{3pt}]
\tzline[--/--,blue,thick](0,2)(4,2)
\tzto[--/--,.75,->,out=45](0,0)(4,1)
\end{tikzpicture}

\begin{tikzpicture}
\tzhelplines(4,3)
\settzmidarrow[rotate=45,bend left=60]
\tzline[--/--,thick](0,3)(4,3)
\tzline[--/--,.25,blue,thick](0,2.5)(4,2.5)
\settzmidarrow[rotate=90,red,tzextend={3pt}{3pt}]
\tzline[--/--,blue,thick](0,2)(4,2)
\tzto[--/--,.75,->,out=45](0,0)(4,1)
\end{tikzpicture}

$26.2 \ \textbackslash tzbrace(')$

\texttt{\textbackslash tzbrace} takes two coordinates as mandatory arguments to draw a \textit{calligraphic brace} connecting them.
The raise value of a brace is 5pt by default and the value can be changed by the first curly brace optional argument {<raise>}).

The amplitude of a brace is 5pt by default. You can control the amplitude by writing the option amplitude=<dim> in the second bracket option [<decoration opt>].

\tzbrace[thick](0,0)(3,1) % works like:
\draw [thick,decorate,decoration={calligraphic brace, amplitude=5pt, raise=5pt}]
(0,0) to (3,1);

The swap version \tzbrace' swaps the coordinates. So it prints a mirror image of \tzbrace. For example, \tzbrace'(0,0)(3,1) is equivalent to \tzbrace(3,1)(0,0).

You can change the style of the decorating brace by the second bracket optional argument [<decoration opt>].

The color of the calligraphic brace can be changed by the option pen colour in the list of [<draw opt>].

You can also move a brace by specifying the option <shift coor> immediately before the the first mandatory coordinate. The empty shift option <> is not allowed.
\begin{tikzpicture}
\helplines
\brace[very thick,pen colour=blue][amplitude=10pt][<.5,.5>(0,0)(3,1)\{AAA\}[a=15pt] \brace[red,very thick][10pt][brace,amplitude=10pt][0,0)(3,1)\{BBB\}[b=20pt]
\end{tikzpicture}

\section{\texttt{\textbackslash tzsnake}(): Snake lines (Experimental)}
\texttt{\textbackslash tzsnake} connects two points with a snaked line, with many default values, using TikZ’s \texttt{to} operation.

\begin{tikzpicture}
\helplines
\snake(0,1)(3,2)
\snake[->,blue,bend right](1,0)(4,3)
\end{tikzpicture}

% syntax: minimum
\tzsnake[opt]{<segment length>}[<decoration opt>][shift coor>
\draw[<coor>]<text>\{<node opt>\}(<coor>)(<text>\{<node opt>\}code.append
% syntax: full
\tzsnake[opt]{<segment length>}[<decoration opt>][shift coor>
\draw[<coor>]<text>\{<node opt>\}(<coor>)(<text>\{<node opt>\}code.append
% defaults
[]{5pt}[many defaults above]<><(\langle>){\{\langle>}{\}{\langle>}{\}{\langle>}{\}

The key \texttt{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{\tzsnake[]} \texttt{[<decoration op>]}
The plus version `\tzsnake+` uses the second mandatory coordinate as a relative coordinate to the first. Everything else is the same as in `\tzsnake`.

```latex
% \tzsnake: \{segment length\} option
\begin{tikzpicture}
\helplines(4,3)
\tzsnake[blue]{3pt}[coil]{0,1}{spring}[a,sloped]{3,2}
\tzsnake[->,bend right]{3pt}
[coil,amplitude=5pt,post length=20pt]
(1,0){coil line}[b,sloped]{4,3}{Up!}[draw,r]
\end{tikzpicture}

% \tzsnake(+): shift
\begin{tikzpicture}
\helplines(4,3)
\tzsnake[->]{post length=20pt}{0,0}{3,0}
\tzsnake[->,red]{post length=20pt}{<1,1>}{0,0}{3,0} %
\tzsnake[->,bend left,blue]{post length=10pt}
(0,1){snake line}[a,sloped]{4,2}{what?}[r]
\end{tikzpicture}
```
Version history

- v2.0 (2022/02/28)
  - Uploaded to CTAN
  - document done
- v1.98 (2022/02/27)
  - added \tzpointangle
  - added \tznarealine(', \settzfnarealinestyle
- v1.97 (2022/02/26)
  - corrected \tpdfN: bug fix
  - redesigned \tznarea (renamed from \tznArea)
  - added \settzfnareaslayer
- v1.96 (2022/02/25)
  - modified \tzanglemark, \tzeightmark
  - added \settzAAlinestyle and \settzRAlinestyle
  - added \settzAAradius, \settzRAsize
  - predefined coordinates (tzAAmid), (tzRAvertex)
  - corrected \tzdefLFn, \tzdefLFnofy, \tzLfn, \tzLFnofy
  - added error message: infinite slope error (for LFn... things)
- v1.95 (2022/02/24)
  - documented
  - redefined \tzedge(+) and \tzedges(+) to make them consistent with Ti\kZ way
  - added aliases: \tzbox, \tzboxring, \tzcirclering
  - renamed \tzangleONE, \tzangleTWO, \tzanglrstory
- 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\tzoval=\tzellipse
- v1.0 (2021/02/28) uploaded to CTAN
- v0.999a (2021/02/27)
  - writing document
  - some \tz<...>AtBegin and \tz<...>AtEnd not documented
- v0.999 (2021/02/24)
  - fixed the title. “Plot Graphs with Ti\kZ Abbreviations”
  - changed the default \tzpathstyle and \tzlinkstyle from ‘--‘ to ‘to‘ (no harm)
  - added \tzpathlayer and \setttzpathlayer, for later use. (currently not used)
  - removed explanation of the option <<<or+++ in the document (considering to remove later)
  - changed the delimiter <\path style> to [\path style] in \tzpath and \tzlink(s) (critical change)
  - added text opacity=1 everywhere fill opacity=.3 is used

Acknowledgement

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 Te\X expansion issues using expl3.
References
