The coordsys and logsys Packages*

© Mogens Lemvig Hansen†
mlhansen@uniserve.com

2004/11/15

Abstract

The coordsys package provides commands for typesetting number lines (coordinate axes), coordinate systems, and grids in the picture environment. The logsys package extends the coordsys package by providing logarithmic, semi-logarithmic, and double-logarithmic coordinate systems and grids.

Contents

1 Regular Coordinate Systems **2**
1.1 One Dimension 2
1.2 Two Dimensions 3
1.3 Bells and Whistles 7
1.3.1 Manual Labels 7
1.3.2 Thick Tick Density 7
1.3.3 Different Scales on the Two Axes 8
1.3.4 Different Styles of Tick Marks 8

2 Logarithmic Coordinate Systems **9**
2.1 One Dimension 9
2.2 Two Dimensions 10

3 Filling the Coordinate Systems **13**
3.1 Intervals 13
3.2 Curves 14

4 Installation **19**

5 License **19**

6 Acknowledgements **19**

*This file describes version 1.3, 2004/11/15.

†The coordsys and logsys packages are distributed under the LaTeX Project Public License; please see Section 5.

1 Regular Coordinate Systems

Load the coordsys package with the `\usepackage` command.

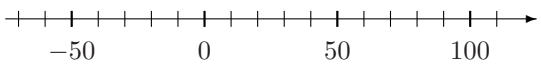
```
\usepackage[⟨options⟩]{coordsys}
```

The coordsys package has three ⟨*options*⟩, centred (the default), outside, and inside, that control the appearance of tick marks; see Section 1.3.4 on page 8.

1.1 One Dimension

`\numbline` The `\numbline` command typesets a number line. For example,

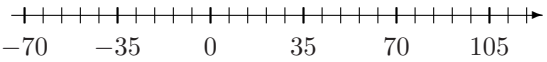
```
\begin{picture}(200,20)(-75,-15)
  \numbline{-75}{125}
\end{picture}
```



You should always use the `\numbline` command inside a picture environment. Do not forget to leave room for the labels on the number line. In the following examples I omit the `\begin{picture}` and `\end{picture}` commands to reduce clutter.

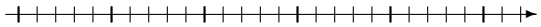
The tick marks are by default separated by 10 units; give `\numbline` an optional first argument to change that. For example

```
\numbline[7]{-75}{125}
```



`\numbline*` The starred version of `\numbline` omits the labels.

```
\numbline*[7]{-75}{125}
```



Thus,

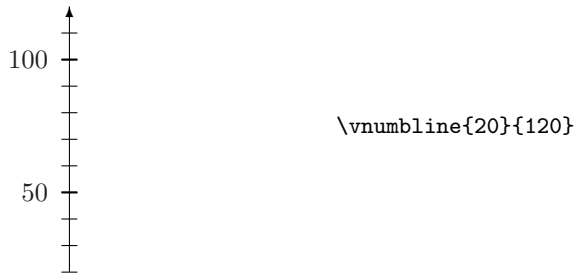
```
\numbline[⟨spacing⟩]{⟨from⟩}{⟨to⟩}
\numbline*[⟨spacing⟩]{⟨from⟩}{⟨to⟩}
```

typeset (horizontal) number lines from ⟨*from*⟩ to ⟨*to*⟩ with ⟨*spacing*⟩ units between the tick marks (the default is 10); use the starred command to suppresses the labels. All the arguments must be integers.

`\vnumbline` The `\vnumbline` and `\vnumbline*` commands similarly typeset vertical number lines.
`\vnumbline*`

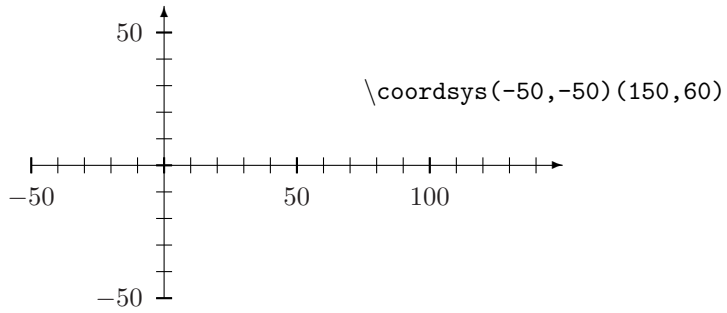
```
\vnumbline[⟨spacing⟩]{⟨from⟩}{⟨to⟩}
\vnumbline*[⟨spacing⟩]{⟨from⟩}{⟨to⟩}
```

The syntax is similar to that of the `\numbline` command on the preceding page.



1.2 Two Dimensions

`\coordsys` Use the `\coordsys` command to typeset coordinate systems.



`\coordsys*` Thus,

`\coordsys[`⟨*h-spacing*⟩`][`⟨*v-spacing*⟩`](`⟨*ll*⟩`)(`⟨*ur*⟩`)`
`\coordsys*[`⟨*h-spacing*⟩`][`⟨*v-spacing*⟩`](`⟨*ll*⟩`)(`⟨*ur*⟩`)`

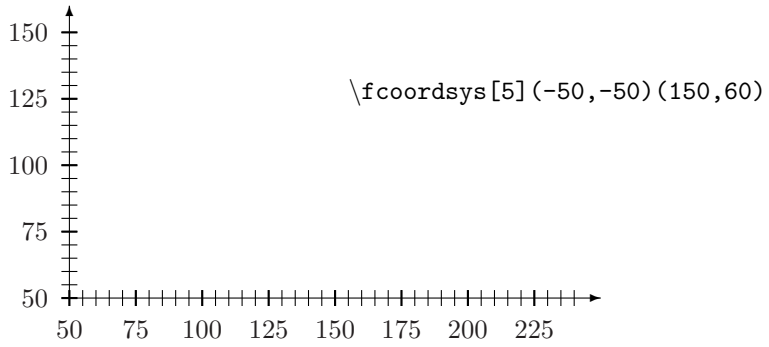
typeset coordinate systems where `(`⟨*ll*⟩`)` is the lower-left corner and `(`⟨*ur*⟩`)` is the upper-right corner. The optional argument ⟨*h-spacing*⟩ gives the spacing between the tick marks on the horizontal axis; the default is 10 units. The optional argument ⟨*v-spacing*⟩ gives the spacing between the tick marks on the vertical axis; the default is to use the same spacing as on the horizontal axis. The starred version omits the labels. All the arguments must be integers.

Regular coordinate systems with intersecting axes are not always appropriate, for example when the point $(0, 0)$ is not in the range of the coordinate system. The coordsys package therefore provides some alternative styles. All the coordinate-system-drawing commands have similar syntax.

`\fcoordsys` The `\fcoordsys` command typesets framed coordinate systems; that is, the
`\fcoordsys*` axes are at the left and bottom edges of the system.

`\fcoordsys[`⟨*h-spacing*⟩`][`⟨*v-spacing*⟩`](`⟨*ll*⟩`)(`⟨*ur*⟩`)`
`\fcoordsys*[`⟨*h-spacing*⟩`][`⟨*v-spacing*⟩`](`⟨*ll*⟩`)(`⟨*ur*⟩`)`

The syntax is similar to that of the `\coordsys` command above.

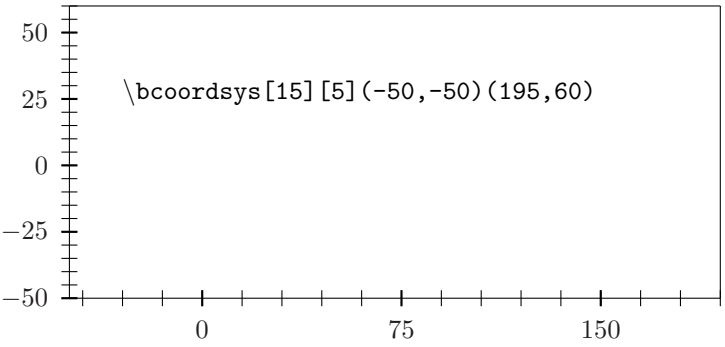


`\bcoordsys`
`\bcoordsys*`

The `\bcoordsys` command typesets boxed coordinate systems.

`\bcoordsys[`⟨*h-spacing*⟩`][`⟨*v-spacing*⟩`](`⟨*ll*⟩`)(`⟨*ur*⟩`)`
`\bcoordsys*[`⟨*h-spacing*⟩`][`⟨*v-spacing*⟩`](`⟨*ll*⟩`)(`⟨*ur*⟩`)`

The syntax is similar to that of the `\coordsys` command on the preceding page.

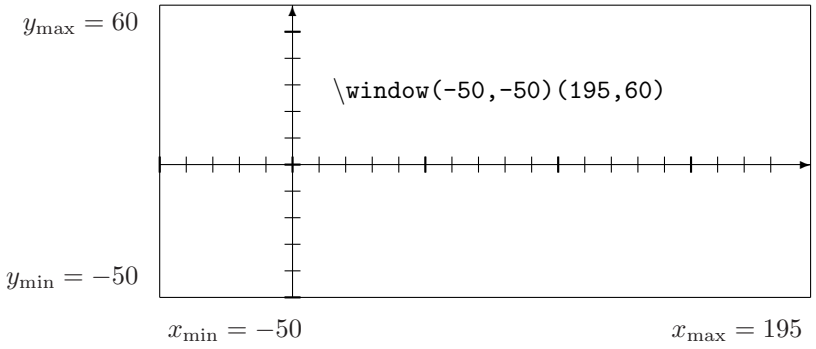


`\window`
`\window*`

The `\window` command typesets a plotting window as on a graphing calculator.

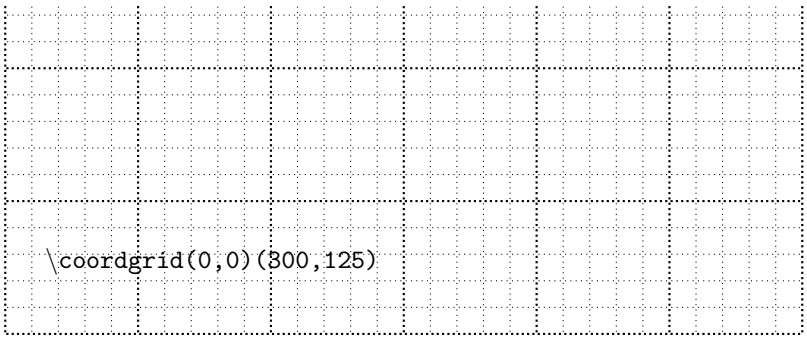
`\window[`⟨*h-spacing*⟩`][`⟨*v-spacing*⟩`](`⟨*ll*⟩`)(`⟨*ur*⟩`)`
`\window*[`⟨*h-spacing*⟩`][`⟨*v-spacing*⟩`](`⟨*ll*⟩`)(`⟨*ur*⟩`)`

The syntax is similar to that of the `\coordsys` command on the page before.



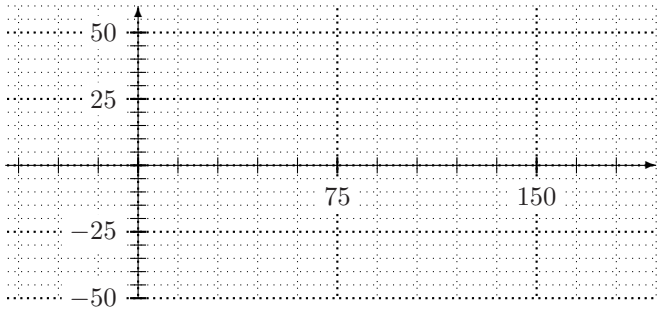
The labels will look nicer if you use the amstext package (which is part of the amsmath package).

`\coordgrid` The `\coordgrid` command typesets a coordinate grid.



The `\coordgrid` command does not print any labels but you can superimpose a coordinate system.

```
\coordgrid[15][5](-50,-50)(195,60)
\coordsys[15][5](-50,-50)(195,60)
```



If you load the color package before the coordsys package, the labels will be printed on a white background.

Typesetting pretty grids with dotted lines is not easy. If a grid does come out ugly, try to increase `\unitlength` to allow for more dots between the lines. You may also reduce the size of the dots; see the `\gridstyle` command on the following

`\coordgrid*` page. Alternatively you can use the `\coordgrid*` command which typesets a coordinate grid with solid lines. You could then use the color package to colour the grid, say, gray; again, see the `\gridstyle` command on the next page.

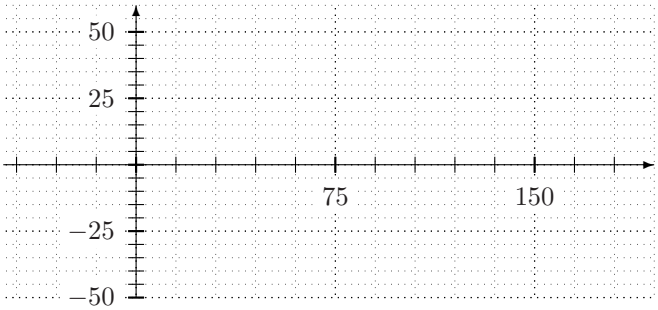
Thus,

`\coordgrid[`⟨*h-spacing*⟩`][`⟨*v-spacing*⟩`](`⟨*ll*⟩`)(`⟨*ur*⟩`)`
`\coordgrid*[`⟨*h-spacing*⟩`][`⟨*v-spacing*⟩`](`⟨*ll*⟩`)(`⟨*ur*⟩`)`

typeset coordinate grids with a syntax similar to that of the `\coordsys` command on page 3. The starred version uses solid lines; the un-starred version uses dotted lines.

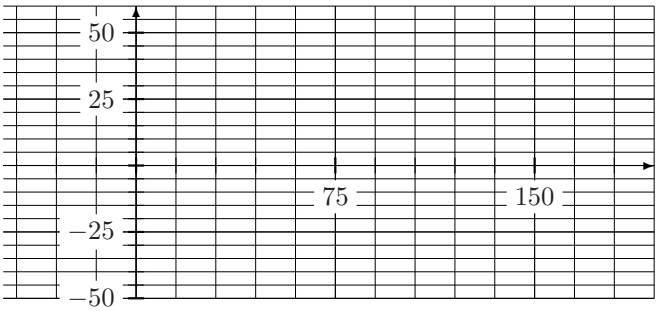
`\gridstyle` The default size of the dots is the with of `\thinlines` and `\thicklines`. If you intend to print your grids at a sufficiently high resolution, you should reduce the size of the dots by issuing a `\gridstyle` command. On my 600 dpi printer, these sizes look nice for points:

```
\gridstyle{\linethickness{0.24pt}}{\linethickness{0.48pt}}
```



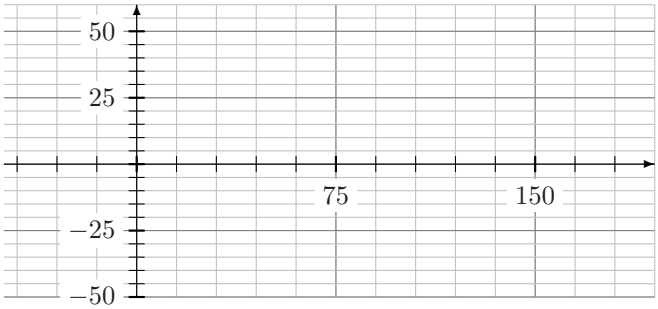
When I use solid lines, I can get away with half that,

```
\gridstyle{\linethickness{0.12pt}}{\linethickness{0.24pt}}
```



If you make grids for the screen, I recommend using colour. For example,

```
\definecolor{gray}{gray}{0.5}
\definecolor{lightgray}{gray}{0.75}
\gridstyle{\thinlines\color{lightgray}}{\thinlines\color{gray}}
\coordgrid*(-109,-65)(200,100)
\coordsys(-109,-65)(200,100)
```



In general,

`\gridstyle{`⟨*thin declaration*⟩`}{`⟨*thick declaration*⟩`}`

Note that these are *declarations*: the ⟨*thin declarations*⟩ apply also to the thick grid lines unless explicitly overruled by the ⟨*thick declaration*⟩. The current `\gridstyle` applies to all types of grids: dotted, solid, regular, or logarithmic. The default style is `\gridstyle{\thinlines}{\thicklines}`.

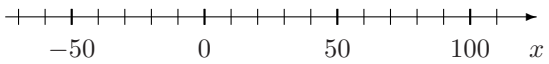
1.3 Bells and Whistles

1.3.1 Manual Labels

`\sethlabel` `\setvlabel`

All the number-line and coordinate-system commands format the labels using the commands `\sethlabel` (for labels on horizontal axes) and `\setvlabel` (for labels on vertical axes). If you set labels manually, you should use the same commands for a uniform appearance.

```
\numbline{-75}{125}
\put(125,0){\sethlabel{x}}
```



The syntax is

`\sethlabel[`⟨*alignment*⟩`]{`⟨*label*⟩`}`
`\setvlabel[`⟨*alignment*⟩`]{`⟨*label*⟩`}`

The ⟨*label*⟩ is set in math mode. The default ⟨*alignment*⟩ is `[t]` for `\sethlabel` and `[r]` for `\setvlabel`. You may want to add to these defaults (as in `\sethlabel[tl]`); you probably do not want to replace the defaults.

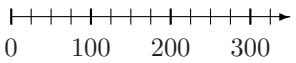
If you want your labels set in some other style, you must re-define `\sethlabel` and `\setvlabel`.

1.3.2 Thick Tick Density

`\hthickratio` `\vthickratio`

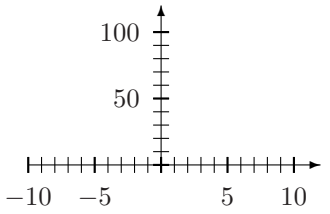
By default every fifth tick mark is thick. You can change that by redefining the commands `\hthickratio` and `\vthickratio`.

```
\renewcommand{\hthickratio}{4}
\numbline[25]{0}{350}
```



1.3.3 Different Scales on the Two Axes

\rescaleby LaTeX (the picture environment) does not support different scales on the two axes; \unitlength is used for both the horizontal and the vertical direction. However, you may create the *appearance* of different scales by scaling the labels on the axes. Here is a coordinate system that has been rigged to contain the graph of $y = x^2$ for x between -10 and 10.




```
\rescaleby{10}{1}{\vlabel}
\coordsys[1](-10,0)(12,12)
```

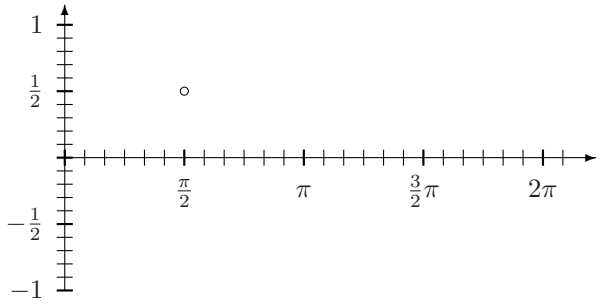
That is,

\rescaleby[⟨*opt*⟩]{⟨*num*⟩}{⟨*den*⟩}{⟨*cmd*⟩}

multiplies the labels by $\frac{\langle num \rangle}{\langle den \rangle}$; both ⟨*num*⟩ and ⟨*den*⟩ should be positive integers. If the optional argument is present, each label will also be multiplied by ⟨*opt*⟩ (which may be a non-integer). The last argument, ⟨*cmd*⟩, must be one of \hlabel or \vlabel.

The following coordinate system has been rigged for graphing sin or cos. Note that only that labels have been scaled; the real coordinates of the small circle are still $(90, 50)$.

```
\renewcommand{\hthickratio}{6}
\rescaleby[\pi]{1}{180}{\hlabel}
\rescaleby{1}{100}{\vlabel}
\coordsys[15][10](0,-100)(400,115)
\put(90,50){\circle{6}}
```



1.3.4 Different Styles of Tick Marks

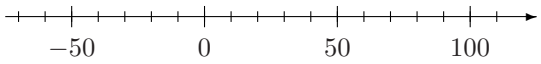
\tickstyle As you can affect the width of the dots or lines of a \coordgrid with a \gridstyle command, so you can affect the width of the tick marks with a \tickstyle command:

`\tickstyle{`⟨*thin declaration*⟩`}{`⟨*thick declaration*⟩`}`

Note that these are *declarations*: the ⟨*thin declarations*⟩ apply also to the thick tick marks unless explicitly overruled by the ⟨*thick declaration*⟩.[1] The default style is `\tickstyle{\thinlines}{\thicklines}`.[2]

`\ticklength` The `\ticklength` command controls the length of the tick marks.

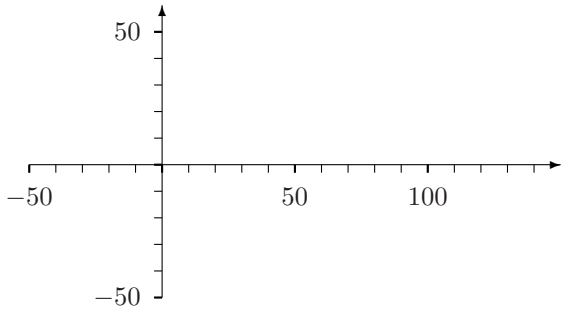
```
\tickstyle{\thinlines\renewcommand{\ticklength}{4pt}}
          {\thinlines\renewcommand{\ticklength}{8pt}}
```



Note that the *command* `\ticklength` must be changed with `\renewcommand`.

If you prefer tick marks just on the outside of the coordinate axes, load coordsys with the outside option.

```
\usepackage[outside]{coordsys}
```



Similarly, the inside option puts the tick marks on the inside of the axes. The default option is centred.

2 Logarithmic Coordinate Systems

Load the logsys package with the `\usepackage` command.

`\usepackage[`⟨*options*⟩`]{coordsys,logsys}`

The logsys package supports the same options, centred, outside, and inside, as the coordsys package.

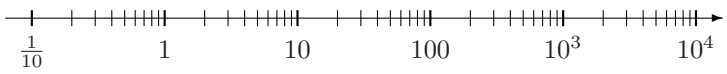
2.1 One Dimension

`\logline` The `\logline` command typesets a horizontal, logarithmic number line.

[1] And the declarations that apply to the thick tick marks spill over to the labels which maybe they shouldn't, so let's not talk about that.

[2] When I wrote on page 7 that the default grid style is `\gridstyle{\thinlines}{\thicklines}` I lied. The default grid style is the follow the current `\tickstyle`.

```
\logline{-60}{210}
```



The `\logline` command typesets only whole blocks of tick marks (whole powers of 10). Making the axis long enough that the tick marks do not collide with the arrow and yet not so long that it looks ridiculous is the users responsibility. The default distance between the thick tick marks (the powers of 10) is 50 units. Therefore I made the axis above extend from -60 (a bit below -50 which appears as $10^{-1} = \frac{1}{10}$) to 210 (a bit above 200 which appears as 10^4).

`\logline*`

Thus,

```
\logline[⟨spacing⟩]{⟨from⟩}{⟨to⟩}
\logline*[⟨spacing⟩]{⟨from⟩}{⟨to⟩}
```

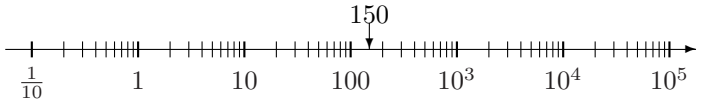
typeset horizontal, logarithmic number lines from ⟨*from*⟩ to ⟨*to*⟩ with ⟨*spacing*⟩ units between the powers of 10; the default is 50. The starred version omits the labels. All the arguments must be integers.

The `\logline` command only typesets an axis; it does not change the way LaTeX and the picture environment interprets coordinates. The equation below gives the relationship between the coordinate to `\put`, x_p, and the apparent coordinate, x_a.

$$x_p = \langle spacing \rangle \log_{10} x_a.$$

For example:

```
\logline[40]{-50}{210}
\put(87.04,10){\makebox(0,0)[b]{150}} % 40*log(150) = 87.04
\put(87.04,10){\vector(0,-1){10}}
```



`\vlogline`
`\vlogline*`

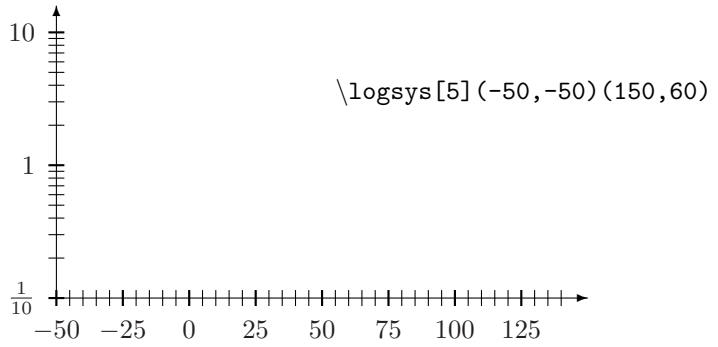
The vertical equivalent is `\vlogline`:

```
\vlogline[⟨spacing⟩]{⟨from⟩}{⟨to⟩}
\vlogline*[⟨spacing⟩]{⟨from⟩}{⟨to⟩}
```

The syntax is similar to that of the `\logline` command above.

2.2 Two Dimensions

`\logsys` Use the `\logsys` command to typeset logarithmic coordinate systems.

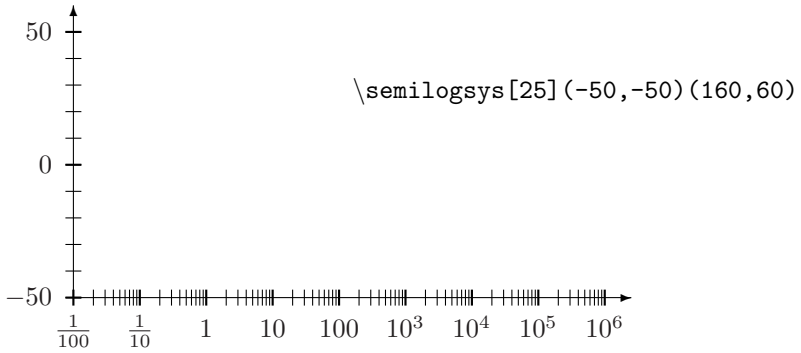


`\logsys*` Thus,

`\logsys[`⟨*h-spacing*⟩`][`⟨*v-spacing*⟩`](`⟨*ll*⟩`)(`⟨*ur*⟩`)`
`\logsys*[`⟨*h-spacing*⟩`][`⟨*v-spacing*⟩`](`⟨*ll*⟩`)(`⟨*ur*⟩`)`

typeset coordinate systems with logarithmic vertical axis, where `(`⟨*ll*⟩`)` is the lower-left corner and `(`⟨*ur*⟩`)` is the upper-right corner. The optional argument ⟨*h-spacing*⟩ gives the spacing between the tick marks on the horizontal axis; the default is 10 units. The optional argument ⟨*v-spacing*⟩ gives the spacing between the thick tick marks (powers of 10) on the vertical axis; the default is 50 units. The starred version omits the labels. All the arguments must be integers.

`\semilogsys` Use the `\semilogsys` command to typeset semi-logarithmic coordinate systems.

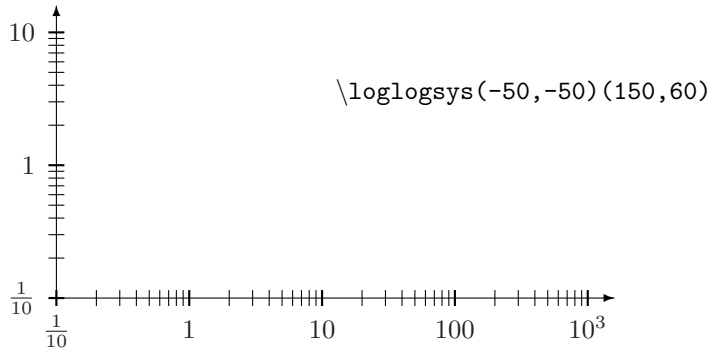


`\semilogsys*` Thus,

`\semilogsys[`⟨*h-spacing*⟩`][`⟨*v-spacing*⟩`](`⟨*ll*⟩`)(`⟨*ur*⟩`)`
`\semilogsys*[`⟨*h-spacing*⟩`][`⟨*v-spacing*⟩`](`⟨*ll*⟩`)(`⟨*ur*⟩`)`

typeset coordinate systems with logarithmic horizontal axis, where `(`⟨*ll*⟩`)` is the lower-left corner and `(`⟨*ur*⟩`)` is the upper-right corner. The optional argument ⟨*h-spacing*⟩ gives the spacing between the thick tick marks (powers of 10) on the horizontal axis; the default is 50 units. The optional argument ⟨*v-spacing*⟩ gives the spacing between the tick marks on the vertical axis; the default is 10 units. The starred version omits the labels. All the arguments must be integers.

`\loglogsys` Use the `\loglogsys` command to typeset double-logarithmic coordinate systems.



\loglogsys* Thus,

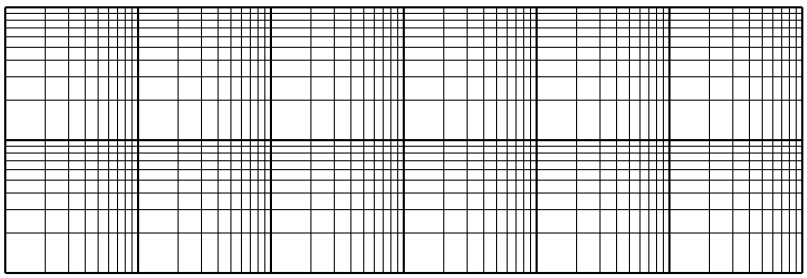
\loglogsys[⟨*h-spacing*⟩][⟨*v-spacing*⟩](⟨*ll*⟩)(⟨*ur*⟩)
\loglogsys*[⟨*h-spacing*⟩][⟨*v-spacing*⟩](⟨*ll*⟩)(⟨*ur*⟩)

typeset coordinate systems with two logarithmic axes, where (⟨*ll*⟩) is the lower-left corner and (⟨*ur*⟩) is the upper-right corner. The optional argument ⟨*h-spacing*⟩ gives the spacing between the thick tick marks (powers of 10) on the horizontal axis; the default is 50 units. The optional argument ⟨*v-spacing*⟩ gives the spacing between the thick tick marks (powers of 10) on the vertical axis; the default is 50 units. The starred version omits the labels. All the arguments must be integers.

\loggrid
\loggrid*
\semiloggrid
\semiloggrid*
\logloggrid
\logloggrid*

The \loggrid, \loggrid*, \semiloggrid, \semiloggrid*, \logloggrid, and \logloggrid* commands typeset logarithmic grids.

\logloggrid*(0,0)(300,100)



Thus,

\loggrid[⟨*h-spacing*⟩][⟨*v-spacing*⟩](⟨*ll*⟩)(⟨*ur*⟩)
\loggrid*[⟨*h-spacing*⟩][⟨*v-spacing*⟩](⟨*ll*⟩)(⟨*ur*⟩)
\semiloggrid[⟨*h-spacing*⟩][⟨*v-spacing*⟩](⟨*ll*⟩)(⟨*ur*⟩)
\semiloggrid*[⟨*h-spacing*⟩][⟨*v-spacing*⟩](⟨*ll*⟩)(⟨*ur*⟩)
\logloggrid[⟨*h-spacing*⟩][⟨*v-spacing*⟩](⟨*ll*⟩)(⟨*ur*⟩)
\logloggrid*[⟨*h-spacing*⟩][⟨*v-spacing*⟩](⟨*ll*⟩)(⟨*ur*⟩)

typeset logarithmic, semi-logarithmic, and double-logarithmic coordinate grids where (⟨*ll*⟩) is the lower-left corner and (⟨*ur*⟩) is the upper-right corner. The

optional argument ⟨*h-spacing*⟩ gives the spacing between the vertical grid lines and the optional argument ⟨*v-spacing*⟩ gives the spacing between the horizontal grid lines; The default spacing is 10 units on linear axes and 50 units between the thick tick marks (the powers of 10) on logarithmic axes. The starred versions use solid lines; the un-starred versions use dotted lines. All the arguments must be integers.

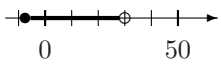
3 Filling the Coordinate Systems

The purpose of the coordsys and logsys packages is to typeset coordinate systems, so maybe the package should end here. However, I have had to write a bit of code to help me fill my coordinate systems, so I'll share that.

3.1 Intervals

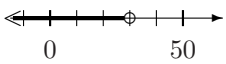
`\interval` The `\interval` command takes as argument an interval in standard notation and draws it on the horizontal coordinate axis or number line.

```
\numbline{-15}{65}
\interval[-7.5,30)
```



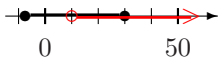
The interval can be open, half-open, or closed as in (a, b), $(a, b]$, $[a, b)$, and $[a, b]$[3] You can specify infinite intervals with `<` or `>`.

```
\numbline{-15}{65}
\interval<-14,30[
```



`\intervalthickness` The `\intervalthickness` command sets the thickness of the fat line that marks the interval. An optional argument allows you to draw the fat line off centre.

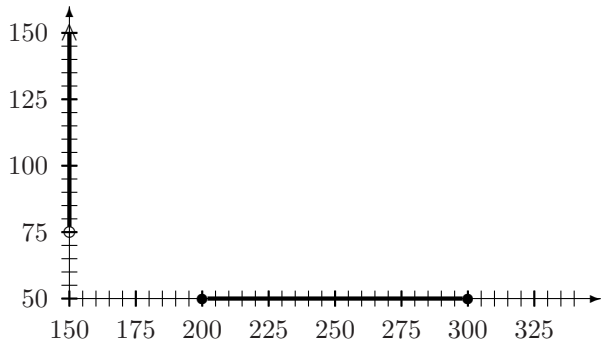
```
\numbline{-15}{65}
\intervalthickness[0.5pt]{1pt}
\interval[-7.5,30]
\color{red}
\intervalthickness[-0.5pt]{1pt}
\interval(10,55>
```



`\vinterval` The `\vinterval` command similarly draws intervals on the vertical axis.

[3] The notation $]a, b[$ for open (or half-open) intervals is also supported.

```
\fcoordsys[5](150,50)(350,160)
\interval[200,300]
\vinterval(75,150>
```



To sum up:

\interval⟨*L-delim*⟩⟨*from*⟩,⟨*to*⟩⟨*R-delim*⟩
\vinterval⟨*L-delim*⟩⟨*from*⟩,⟨*to*⟩⟨*R-delim*⟩
\intervalthickness[⟨*offset*⟩]{⟨*width*⟩}

Here ⟨*L-delim*⟩ must be one of `[`, `(`, `<`, or `]`, and ⟨*R-delim*⟩ must be one of `]`, `)`, `>`, or `[`. Both ⟨*from*⟩ and ⟨*to*⟩ are numbers, integers or decimals[4] Both ⟨*offset*⟩ and ⟨*width*⟩ must be lengths. The centre of the fat interval line will drawn ⟨*offset*⟩ above the horizontal axis or ⟨*offset*⟩ to the right of the vertical axis.

3.2 Curves

For plotting curves I recommend the epic and eepic packages.[5] I especially like (e)epic's `\putfile` command which allows you to calculate your plot with some external program, export the points to plot to a text file, and import that file to have LaTeX draw the plot. For external program I use Maple. The coordsys package includes a piece of Maple code that exports a Maple plot in the form `\putfile` expects. You can therefore take advantage of Maple adaptive plotting algorithm[6] in your LaTeX plots.

Here is how it works: In Maple, read in the file `putfile` which is part of the coordsys package.

```
> read "/where/you/placed/putfile";
```

[4] Due to implementation details, a decimal must have at least one digit on either side of the decimal point. Thus `0.5` is ok but `.5` is not.

[5] Both are old LaTeX209 packages that work fine in LaTeX 2ε. The eepic package re-implements the epic package making use of PostScript. The eepic package works well with dvips and dvipdfm but not with pdfLaTeX.

[6] Maple cleverly calculates more points where the curve wiggles.

Make your plot, here $y = \frac{x^2}{10}$.

```
> plot( x^2/10, x=-10..10, title="x^2/10");
```

When you are happy with your plot, assign it to a variable.

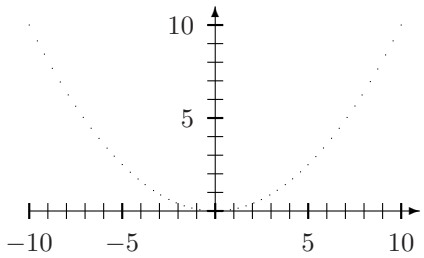
```
> P := %:
```

Then use the `putfile` command to export the curve to the file `parabola.put`.

```
> putfile( "parabola.put", P );
```

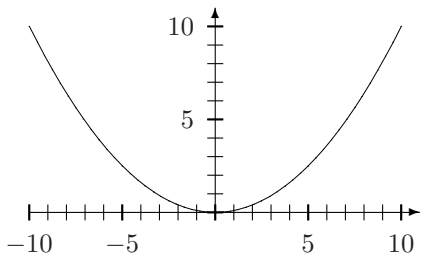
In LaTeX, use the `\putfile` command to put `\picsquare`s at each point listed in `parabola.put`.

```
\coordsys[1](-10,0)(11,11)
\putfile{parabola.put}{\picsquare}
```



The (e)epic package provides an environment, drawjoin, that joins all points set with the special `\put` command `\jput`. Thus, to draw a curve with the `\putfile` command you enter a drawjoin environment, let `\put` be a synonym for `\jput`, and call `\putfile`.

```
\coordsys[1](-10,0)(11,11)
\begin{drawjoin}
   \let\put=\jput
   \putfile{parabola.put}{\picsquare}
\end{drawjoin}
```

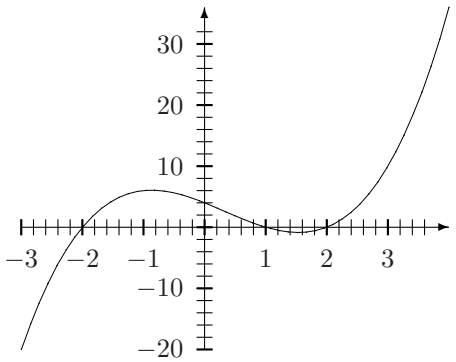


To draw a curve that requires (or looks best with) different scales on the two axes, you can use the `scale` option to `putfile` in Maple:

```
> plot( (x+2)*(x-1)*(x-2), x=-3..4 );
> putfile( "cubic.put", %, scale=[10, 1] );
```

Now all the x-coordinates in `cubic.put` are 10 times larger than they should be. Thus re-scale the horizontal labels by $\frac{1}{10}$.

```
\rescaleby{1}{10}{\hlabel}
\coordsys[2](-30,-20)(40,36)
\begin{drawjoin}
   \let\put=\jput
   \putfile{cubic.put}{\picsquare}
\end{drawjoin}
```

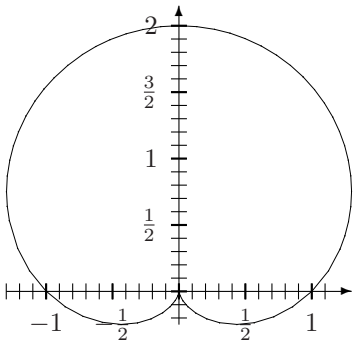


Scaling can be necessary even when the two axes do use the same scale. Consider the graph of $r = 1 + \sin\theta$ in polar coordinates. The graph is 2.6 units wide and 2.25 units tall. All the arguments to the `\coordsys` command must be integers, so the smallest distance between tick marks is 1 unit; the coordinate system will have very few tick marks. Instead scale up by, say, a factor 100. In Maple:

```
> polarplot( 1+sin(theta), theta=0..2*Pi );
> putfile( "cardioid.put", %, scale=[100,100] );
```

In LaTeX:

```
\rescaleby{1}{100}{\hlabel}
\rescaleby{1}{100}{\vlabel}
\coordsys(-130,-25)(130,215)
\begin{drawjoin}
   \let\put=\jput
   \putfile{cardioid.put}{\picsquare}
\end{drawjoin}
```

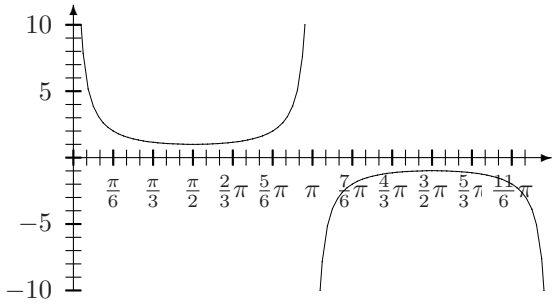


If the plot contains more than one curve, `putfile` generates one file for each. You can of course specify multiple curves directly, but they can also be the result of the `discont` option to `plot`.

```
> plot( csc(x), x=0..2*Pi, y=-10..10, discont=true );
> putfile( "csc.put", %, scale=[180/Pi, 10] );
```

Here `putfile` wrote two files calling them `csc1.put` and `csc2.put`. These two curves must be drawn by different drawjoin environments.

```
\renewcommand{\hthickratio}{3}
\rescaleby[\pi]{1}{180}{\hlabel}
\rescaleby{1}{10}{\vlabel}
\coordsys(0,-100)(360,115)
\begin{drawjoin}
   \let\put=\jput
   \putfile{csc1.put}{\picsquare}
\end{drawjoin}
\begin{drawjoin}
   \let\put=\jput
   \putfile{csc2.put}{\picsquare}
\end{drawjoin}
```

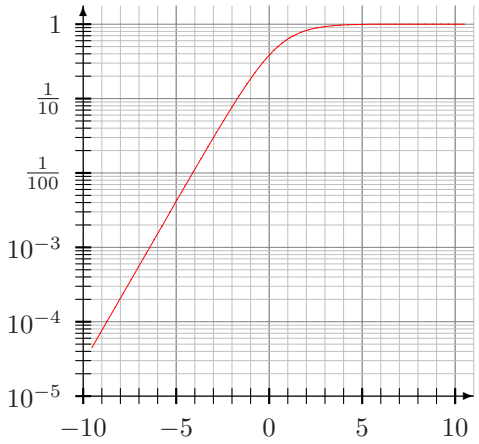


For logarithmic plots scaling is almost always necessary. Consider the graph of $y = \frac{1}{1+e^{-x}}$ for x running from -10 to 10. The y-coordinates extend over five powers of 10, so scale by 4 to make it comparable to the range of x-coordinates.

```
> logplot(1/(1+exp(-x)), x=-10..10);
> putfile("logistic.put", %, scale=[1,4]);
```

The `\rescaleby` command is not suitable for logarithmic axes. Instead set the optional ⟨*spacing*⟩ argument to the correct distance between the powers of 10, in this case 4.

```
\logsys[1][4](-10,-20)(11,1)
\begin{drawjoin}
   \let\put=\jput
   \putfile{logistic.put}{\picsquare}
\end{drawjoin}
```



Use the `putfile` command as follows:

`putfile(` ⟨*filename*⟩`,` ⟨*PLOT structure*⟩ [`,` ⟨*options*⟩] `);`

If ⟨*filename*⟩ is a string and the ⟨*PLOT structure*⟩ contains more than one curve, `putfile` will append $1, 2, 3, \ldots$ as needed; `putfile` also appends an extension of `.tex` if the ⟨*filename*⟩ has no extension. If the ⟨*filename*⟩ is a list or set of filenames, they are used as is. (So if you do not want `putfile` to meddle with your filename, give it in a set.)

The only possible option is

`scale = [` ⟨*number*⟩`,` ⟨*number*⟩ `]`

The `putfile` command will multiply all x-coordinates by the first number and all y-coordinates by the second number. Both numbers must be `evalf`able.

If you scale the coordinates of a curve, you must of course compensate for that to get an honest representation of your data in LaTeX.

- If you scale a regular (linear) axis by, say, `scale=[3/5,1]`, you should use `\rescaleby{5}{3}{\hlabels}` to re-scale the labels.

- If you scale a logarithmic axis by, say, `scale=[1,30]` you should use the optional ⟨*spacing*⟩ argument to the command that draws the coordinate system, say `\logsys[30](...`

4 Installation

Extract the `.sty` files from the `.dtx` file by running `coordsys.ins` through LaTeX. Then generate the documentation for the coordsys and logsys packages by running the file `coordsys.dtx` through LaTeX—thrice to resolve cross references.[7]

You now have to decide what to do with several files.

- Move the files `coordsys.sty` and `logsys.sty` to some directory where LaTeX can find it; `(local)texmf/tex/latex/misc` would be the natural choice.
- Move the documentation, `coordsys.dvi` or `coordsys.pdf`, to `(local)texmf/doc/latex/misc`.
- You may discard the source files, `coordsys.dtx` and `coordsys.ins`, or store them in `(local)texmf/source/latex/misc`.
- If you ran `coordsys.dtx` through LaTeX, several `.put` were created. You can discard those.

5 License

This program consists of the files `coordsys.dtx` and `coordsys.ins`.

6 Acknowledgements

Thank you to Scott Pakin (`pakin@uiuc.edu`) for his input on the implementation of Euclid's algorithm.

Thank you to Svend Daugaard Pedersen (`Svend@DaugaardPedersen.dk`) for catching an error in the logarithmic systems and for suggestions that led to `\gridstyle`.

Thank you to Staffan Lundberg who found an error in my understanding of Maple's structured types and thus prompted me to polish the Maple code for `pufile`.

[7] If you want an index, you must run MakeIndex (`makeindex -s gind.ist coordsys`) between the second and third LaTeX run.

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

B

\bcoordsys *4*
\bcoordsys* *4*

C

\coordgrid *5*
\coordgrid* *5*
\coordsys *3*
\coordsys* *3*

F

\fcoordsys *3*
\fcoordsys* *3*

G

\gridstyle *6*

H

\hthickratio *7*

I

\interval *13*
\intervalthickness . *13*

L

\loggrid *12*
\loggrid* *12*
\logline *9*
\logline* *10*
\logloggrid *12*
\logloggrid* *12*
\loglogsys *11*
\loglogsys* *12*
\logsys *10*
\logsys* *11*

N

\numbline *2*
\numbline* *2*

R

\rescaleby *8*

S

\semiloggrid *12*
\semiloggrid* *12*
\semilogsys *11*
\semilogsys* *11*
\sethlabel *7*
\setvlabel *7*

T

\ticklength *9*
\tickstyle *8*

V

\vinterval *13*
\vlogline *10*
\vlogline* *10*
\vnumbline *2*
\vnumbline* *2*
\vthickratio *7*

W

\window *4*
\window* *4*