pstricks-add
additionals Macros for pstricks*

v.1.50

Herbert Voß

June 24, 2004

Abstract

This version of pstricks-add needs pstricks.tex - patch 15 from May 2004, otherwise the additional macros may not work in the espected way. The whole ellipsis stuff, the dashed lines and the option asolid (renamed to eofill) are now part of the new pstricks.tex package, available at CTAN or at http://perce.de/LaTeX/pstricks.

Contents

*This document was written with Kile: 1.61 (Qt: 3.1.1; KDE: 3.2.1; http://sourceforge.net/projects/kile/) and the PDF output was build with VTeX/Free (http://www.micropress-inc.com/linux)

Part I
pstricks

1 Numeric functions

pstricks itself has an own divide macro, called \pst@divide which can divide two lengthes and saves the quotient as a floating number:

```
\pst@divide{<dividend>}{<divisor>}{<result as a macro>}

\makeatletter
\pst@divide{34pt}{6pt}\quotient
\quotient
\makeatother
```

this gives the output 5.66666. The result is not a length!

pstricks-add defines an additional numeric function for the modulo:

```
\pst@mod{<integer>}{<integer>}{<result as a macro>}

\makeatletter
\pst@mod{34}{6}\modulo
\quotient
\makeatother
```

this gives the output 4. Using this internal numeric functions in documents require a setting inside the makeatletter and makeatother environment. It makes some sense to define a new macroname in the preamble to use it without, e.g. \let\modulo\pst@mod.

2 \pslineII Colored lines

The dashed lines are by default black and white lines. The new macro \pslineII offers two-color lines and has the same syntax as \psline.

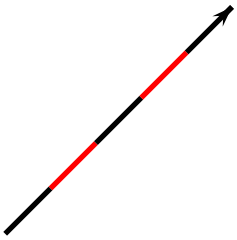
```
1 \pslineII[linewidth=5pt,arrowscale=2]{o-o}(0,0)(12,0)
```

2.1 The options

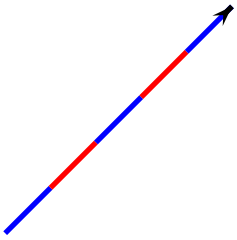
name	meaning
`dashColorI`	first color, default is `black`
`dashColorII`	second color, default is `red`
`dashNo`	the difference in per cent of the colored lines, default ist 0.2
`linecap`	how two lines are connected. 0: no modification 1: rounded edges 2: an additional half square at both ends

`dashNo` can have values greater than 1. In this case the value will be taken as an absolute width in the pt unit. Only this unit is possible!

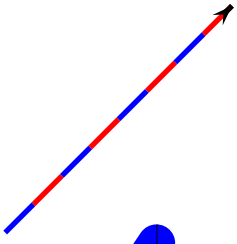
2.2 Examples



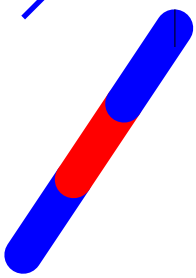
```
\pslineII{->}(0,0)(3,3)
```



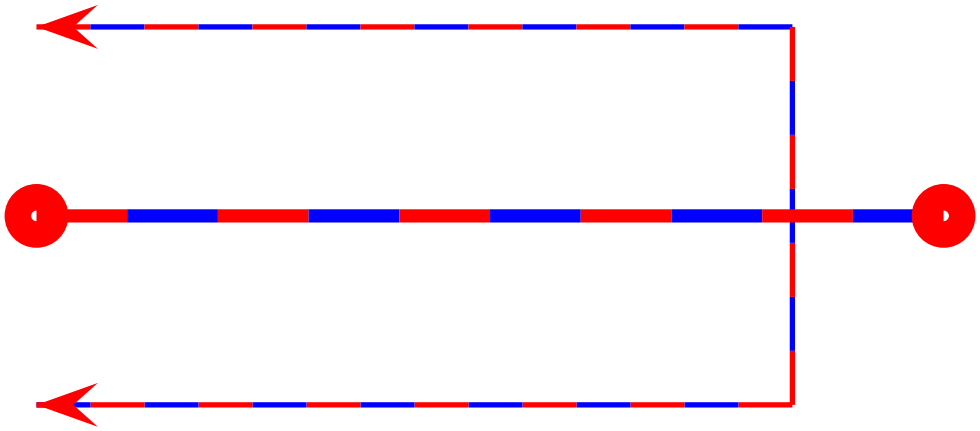
```
\pslineII[dashColorI=blue]{->}(0,0)(3,3)
```



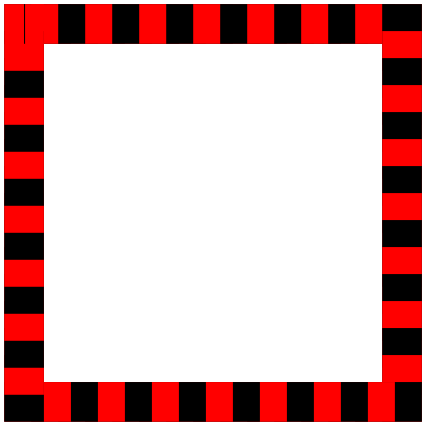
```
\pslineII[dashColorI=blue,dashNo=15]{->}(0,0)(3,3)
```

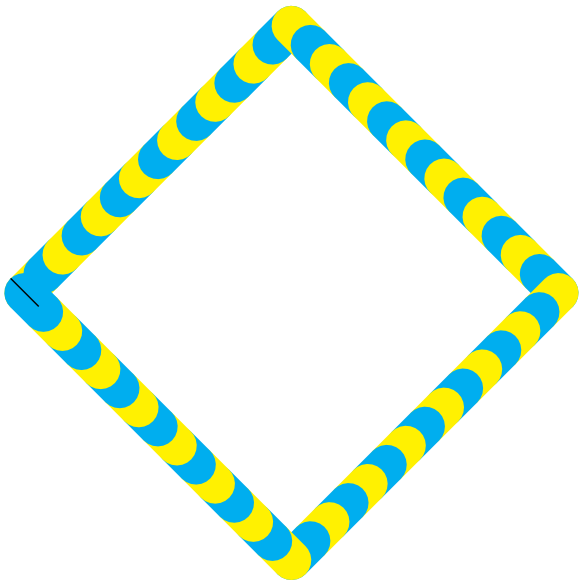


```
\pslineII[dashColorI=blue,%
    linecap=1,%
    dashNo=0.3,linewidth=0.5](0,0)(2,3)
```



```
\psset{linecolor=red,arrowscale=3}
\psset{dashColorI=red,dashColorII=blue,dashNo=20,linewidth=2pt}
\begin{pspicture}(0,0)(12,-5)
\pslineII{<->}(0,0)(10,0)(10,-5)(0,-5)
\pslineII[linewidth=5pt,%
 dashNo=7,arrowscale=2]{o-o}(0,-2.5)(12,-2.5)
\end{pspicture}
```





```
\psset{linewidth=15pt,dashNo=10}
\begin{pspicture}(0,1)(10,-6)
 \pslineII[linecap=2](0,0)(5,0)(5,-5)(0,-5)(0,0)
 \rput{45}(7,-2.5){%
  \pslineII[%
   linecap=1,%
   dashColorI=yellow,%
   dashColorII=cyan](0,0)(5,0)(5,-5)(0,-5)(0,0)%
 }
\end{pspicture}
```

3 \pslineIII Variable linewidth

By default all lines have a fixed width. \pslineIII allows to define the start and the end width of a line. It has the same syntax as \psline.

```
\pslineIII[wBegin=1cm,wEnd=0.3cm,linecolor=cyan](0,0)(12,0)
```

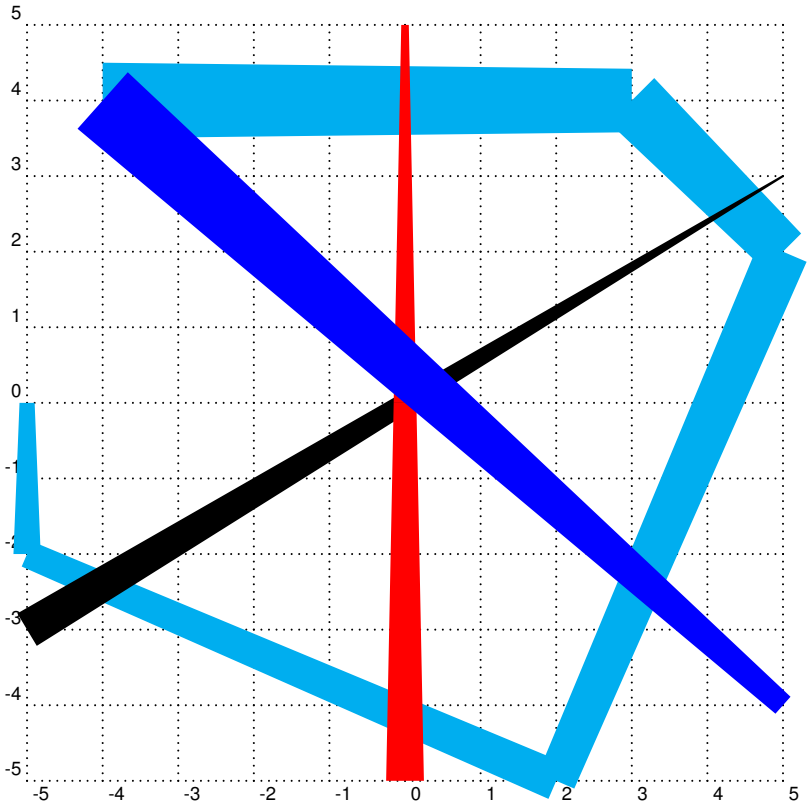
3.1 The options

name	meaning
wBegin	first width, default is \pslinewidth
wEnd	last width, default is \pslinewidth

It is also possible to use pslineIII with more than two coordinates, like

```
\pslineIII[wBegin=1cm,wEnd=0.1cm,linecolor=cyan](0,0)(0,1.5)(12,1.5)(12,0)
```

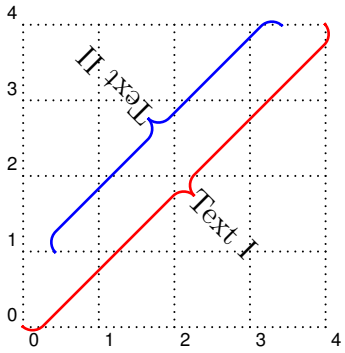
3.2 Examples


5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
-4
-3
-2
-1
0
1
2
3
4
5

4 \psbrace

4.1 Syntax

```
\psbrace[<options>](<A>)(<B>){<text>}
```




```
\begin{pspicture}(4,4)
\psgrid[subgriddiv=0,griddots=10]
\pnode(0,0){A}
\pnode(4,4){B}
\psbrace[linecolor=red,ref=lC](A)(B){Text I}
\psbrace[linecolor=blue,ref=lC](3,4)(0,1){Text II}
\end{pspicture}
```

The option `\specialCoor` is enabled, so that all types of coordinates are possible, (nodename), (x, y), $(nodeA|nodeB)$, ...

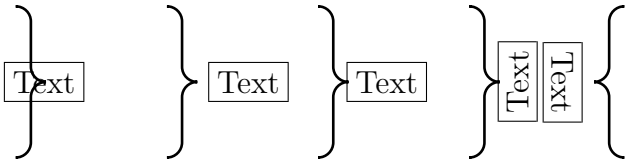
4.2 Options

Additional to all other available options from `pstricks` or the other related packages, there are two new option, named `braceWidth` and `bracePos`. All important ones are shown in the following table.

name	meaning
`braceWidth`	default is 0.35
`bracePos`	relative position (default is 0.5)
`linearc`	absolute value for the arcs (default is 2mm)
`nodesepA`	x-separation (default is 0*pt*)
`nodesepB`	y-separation (default is 0*pt*)
`rot`	additional rotating for the text (default is 0)
`ref`	reference point for the text (default is c)

By default the text is written perpedicular to the brace line and can be changed with the `pstricks` option `rot=...`. The text parameter can take any object and may also be empty. The reference point can be any value of the combination of `l` (left) or `r` (right) and `b` (bottom) or `B` (Baseline) or `C` (center) or `t` (top), where the default is `c`, the center of the object.

4.3 Examples



```
\psbrace(0,0)(0,2){\fbox{Text}}%
\psbrace[nodesepA=20pt](2,0)(2,2){\fbox{Text}}
\psbrace[ref=lC](4,0)(4,2){\fbox{Text}}
\psbrace[ref=lt,rot=90,nodesepB=-15pt](6,0)(6,2){\fbox{Text}}
\psbrace[ref=lt,rot=90,nodesepA=-5pt,nodesepB=15pt](8,2)(8,0){\fbox{Text}}
```

$\int_1^\infty \frac{1}{x}\,dx = 1$ $\int_1^\infty \frac{1}{x}\,dx = 1$ $\int_1^\infty \frac{1}{x}\,dx = 1$ $\int_1^\infty \frac{1}{x}\,dx = 1$ $\int_1^\infty \frac{1}{x}\,dx = 1$

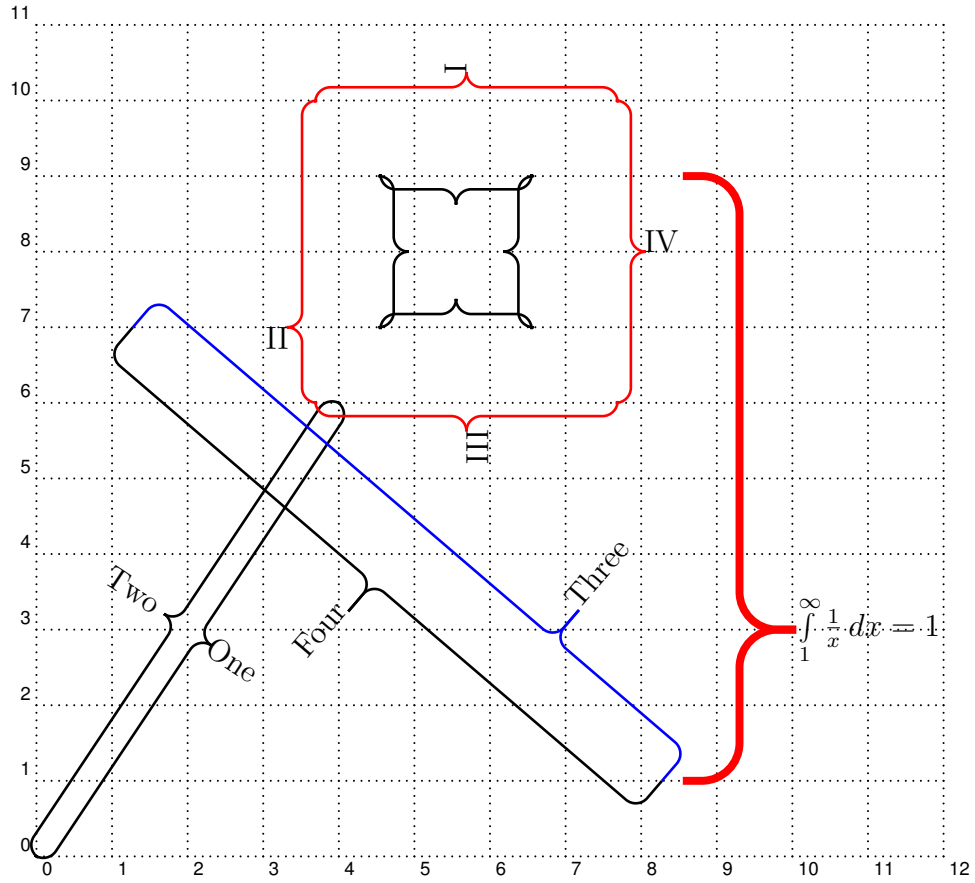
```
\def\someMath{$\int\limits_1^{\infty}\frac{1}{x}\,dx=1$}
\psbrace(0,0)(0,2){\someMath}%
\psbrace[nodesepA=30pt](2,0)(2,2){\someMath}
\psbrace[ref=lC](4,0)(4,2){\someMath}
\psbrace[ref=lt,rot=90,nodesepB=-30pt](6,0)(6,2){\someMath}
\psbrace[ref=lt,rot=90,nodesepB=30pt](8,2)(8,0){\someMath}
```

some very, very long wonderful Text

Text

Text

```
\begin{pspicture}(\linewidth,5)
\psbrace(0,0.5)(\linewidth,0.5){\fbox{Text}}%
\psbrace[bracePos=0.25,nodesepB=-10pt,rot=90](0,2)(\linewidth,2){\fbox{Text
  }}
\psbrace[ref=lC,nodesepA=-3.5cm,nodesepB=-15pt,rot=90](0,4)(\linewidth,4){\
  fbox{some very, very long wonderful Text}}
\end{pspicture}
```




```
\begin{pspicture}(12,11)
\psgrid[subgriddiv=0,griddots=10]
\pnode(0,0){A}
\pnode(4,6){B}
\psbrace[ref=lC](A)(B){One}
\psbrace[rot=180,nodesepA=-5pt,ref=rb](B)(A){Two}
\psbrace[linecolor=blue,bracePos=0.25,%
 braceWidth=1,ref=lB](8,1)(1,7){Three}
\psbrace[braceWidth=-1,rot=180,ref=rB](8,1)(1,7){Four}
\psbrace[linearc=0.5,linecolor=red,linewidth=3pt,%
 braceWidth=1.5,bracePos=0.25,ref=lC](8,1)(8,9){\someMath}
\psbrace(4,9)(6,9){}
\psbrace(6,9)(6,7){}
\psbrace(6,7)(4,7){}
\psbrace(4,7)(4,9){}
\psset{linecolor=red}
\psbrace[ref=lb](7,10)(3,10){I}
\psbrace[ref=lb,bracePos=0.75](3,10)(3,6){II}
\psbrace[ref=lb](3,6)(7,6){III}
\psbrace[ref=lb](7,6)(7,10){IV}
\end{pspicture}
```

It is also possible to put a vertical brace around a default paragraph. This works with setting two invisible nodes at the beginning and the end of the paragraph. Inentation is possible with a minipage.

Some nonsense text, which is nothing more than nonsense. Some nonsense text, which is nothing more than nonsense.

Some nonsense text, which is nothing more than nonsense. Some nonsense text, which is nothing more than nonsense. Some nonsense text, which is nothing more than nonsense. Some nonsense text, which is nothing more than nonsense. Some nonsense text, which is nothing more than nonsense. Some nonsense text, which is nothing more than nonsense. Some nonsense text, which is nothing more than nonsense. Some nonsense text, which is nothing more than nonsense. Some nonsense text, which is nothing more than nonsense.

Some nonsense text, which is nothing more than nonsense. Some nonsense text, which is nothing more than nonsense.

Some nonsense text, which is nothing more than nonsense. Some nonsense text, which is nothing more than nonsense. Some nonsense text, which is nothing more than nonsense. Some nonsense text, which is nothing more than nonsense. Some nonsense text, which is nothing more than nonsense. Some nonsense text, which is nothing more than nonsense. Some nonsense text, which is nothing more than nonsense. Some nonsense text, which is nothing more than nonsense.

```
Some nonsense text, which is nothing more than nonsense.
Some nonsense text, which is nothing more than nonsense.

\noindent\rnode{A}{}

\vspace*{-1ex}
Some nonsense text, which is nothing more than nonsense.
Some nonsense text, which is nothing more than nonsense.
Some nonsense text, which is nothing more than nonsense.
Some nonsense text, which is nothing more than nonsense.
Some nonsense text, which is nothing more than nonsense.
Some nonsense text, which is nothing more than nonsense.
Some nonsense text, which is nothing more than nonsense.
Some nonsense text, which is nothing more than nonsense.

\vspace*{-2ex}
\noindent\rnode{B}{}\psbrace[linecolor=red](A)(B){}

Some nonsense text, which is nothing more than nonsense.
Some nonsense text, which is nothing more than nonsense.

\medskip
\hfill\begin{minipage}{0.95\linewidth}
\noindent\rnode{A}{}

\vspace*{-1ex}
Some nonsense text, which is nothing more than nonsense.
Some nonsense text, which is nothing more than nonsense.
Some nonsense text, which is nothing more than nonsense.
Some nonsense text, which is nothing more than nonsense.
```

```
Some nonsense text, which is nothing more than nonsense.
Some nonsense text, which is nothing more than nonsense.
Some nonsense text, which is nothing more than nonsense.
Some nonsense text, which is nothing more than nonsense.

\vspace*{-2ex}
\noindent\rnode{B}{}\psbrace[linecolor=red](A)(B){}
\end{minipage}
```

5 Arrows

5.1 Definition

`pstricks-add` defines the following "arrows":

Value	Example	Name
`-`		None
`<->`		Arrowheads.
`>-<`		Reverse arrowheads.
`<<->>`		Double arrowheads.
`>>-<<`		Double reverse arrowheads.
`|-|`		T-bars, flush to endpoints.
`|*-|*`		T-bars, centered on endpoints.
`[-]`		Square brackets.
`]-[`		Reversed square brackets.
`(-)`		Rounded brackets.
`)-(`		Reversed rounded brackets.
`o-o`		Circles, centered on endpoints.
`*-*`		Disks, centered on endpoints.
`oo-oo`		Circles, flush to endpoints.
`**-**`		Disks, flush to endpoints.
`|<->|`		T-bars and arrows.
`|>-<|`		T-bars and reverse arrows.
`H-H|`		left/right hook arrows.

You can also mix and match, e.g., `->`, `*-)` and `[->` are all valid values of the `arrows` parameter. The parameter can be set with

```
\psset{arrows=<type>}
```

or for some macros with a special option, like

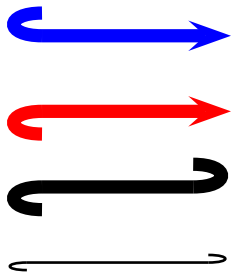
```
\psline[<general options>]{<arrow type>}(A)(B)
\psline[linecolor=red,linewidth=2pt]{|->}(0,0)(0,2)
```

5.2 hookrightarrow and hookleftarrow

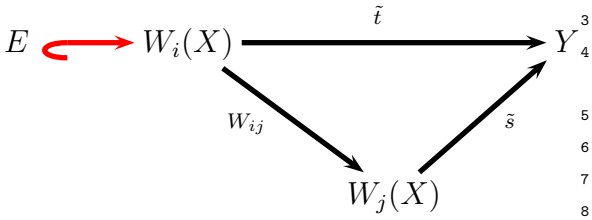
This is another type of an arrow and abbreviated with H. The length and width of the hook is set by the new options hooklength and hookwidth, which are by default set to

```
\psset{hooklength=3mm,hookwidth=1mm}
```

If the line begins with a right hook then the line ends with a left hook and vice versa:



```
\begin{pspicture}(3,4)
\psline[linewidth=5pt,linecolor=blue,hooklength=5mm,
  hookwidth=-3mm]{H->}(0,3.5)(3,3.5)
\psline[linewidth=5pt,linecolor=red,hooklength=5mm,hookwidth
  =3mm]{H->}(0,2.5)(3,2.5)
\psline[linewidth=5pt,hooklength=5mm,hookwidth=3mm]{H-H
  }(0,1.5)(3,1.5)
\psline[linewidth=1pt]{H-H}(0,0.5)(3,0.5)
\end{pspicture}
```




```
$\begin{psmatrix}
E&W_i(X)&&Y\\
&&W_j(X)
\psset{arrows=->,nodesep=3pt,linewidth=2pt
  }
\everypsbox{\scriptstyle}
\ncline[linecolor=red,arrows=H->,%
  hooklength=4mm,hookwidth=2mm]{1,1}{1,2}
\ncline{1,2}{1,4}^{\tilde{t}}
\ncline{1,2}{2,3}<{W_{ij}}
\ncline{2,3}{1,4}>{\tilde{s}}
\end{psmatrix}$
```

5.3 ArrowInside Option

It is now possible to have arrows inside the lines and not only at the beginning or the end. The new defined options

Name	Example	Output
ArrowInside	\psline[ArrowInside=->](0,0)(2,0)	
ArrowInsidePos	\psline[ArrowInside=->,% ArrowInsidePos=0.25](0,0)(2,0)	
ArrowInsidePos	\psline[ArrowInside=->,% ArrowInsidePos=10](0,0)(2,0)	
ArrowInsideNo	\psline[ArrowInside=->,% ArrowInsideNo=2](0,0)(2,0)	
ArrowInsideOffset	\psline[ArrowInside=->,% ArrowInsideNo=2,% ArrowInsideOffset=0.1](0,0)(2,0)	
ArrowInside	\psline[ArrowInside=->]{->}(0,0)(2,0)	
ArrowInsidePos	\psline[ArrowInside=->,% ArrowInsidePos=0.25]{->}(0,0)(2,0)	
ArrowInsidePos	\psline[ArrowInside=->,% ArrowInsidePos=10]{->}(0,0)(2,0)	
ArrowInsideNo	\psline[ArrowInside=->,% ArrowInsideNo=2]{->}(0,0)(2,0)	
ArrowInsideOffset	\psline[ArrowInside=->,% ArrowInsideNo=2,% ArrowInsideOffset=0.1]{->}(0,0)(2,0)	
ArrowFill	\psline[ArrowFill=false,% arrowinset=0]{->}(0,0)(2,0)	
ArrowFill	\psline[ArrowFill=false,% arrowinset=0]{<<->>}(0,0)(2,0)	
ArrowFill	\psline[ArrowInside=->,% arrowinset=0,% ArrowFill=false,% ArrowInsideNo=2,% ArrowInsideOffset=0.1]{->}(0,0)(2,0)	

Without the default arrow definition there is only the one inside the line, defined by the type and the position. The position is relative to the length of the whole line. 0.25 means at 25% of the line length. The peak of the arrow gets the coordinates which are calculated by the macro. If you want arrows with an abolute position difference, then choose a value greater than `1`, e.g. `10`, which gives every 10 pt an arrow. The default unit `pt` cannot be changed.

5.4 `ArrowFill` Option

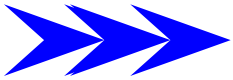
By default all arrows are filled polygons. With the option `ArrowFill=false` there are "white" arrows. Only for the begiining/end arrows they are empty, the inside arrows are overpainted with the line.

```
\psline[%
   linecolor=red,%
   arrowinset=0]{<->}(0,0)(3,0)
```

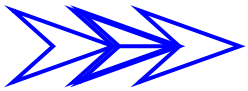
```
\psline[%
   linecolor=red,%
   ArrowFill=false,%
   arrowinset=0]{<->}(0,0)(3,0)
```

```
\psline[%
    linecolor=red,%
    arrowsize=0.2,%
    ArrowFill=false,%
    arrowinset=0]{<->}(0,0)(3,0)
```

```
\psline[%
    linecolor=blue,%
    arrowscale=6,%
    ArrowFill=true]{<->}(0,0)(3,0)
```

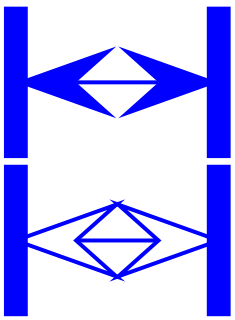


```
\psline[%
    linecolor=blue,%
    arrowscale=6,%
    ArrowFill=false]{<->}(0,0)(3,0)
```



```
\psline[%
    linecolor=blue,%
    arrowscale=6,%
    ArrowFill=true]{>|->|}(0,0)(3,0)
```

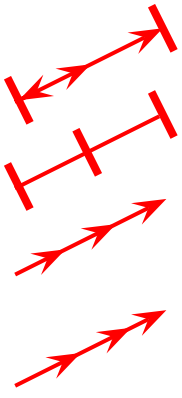
```
\psline[%
    linecolor=blue,%
    arrowscale=6,%
    ArrowFill=false]{>|->|}(0,0)(3,0)
```



5.5 Examples

All examples are printed with `\psset{arrowscale=2,linecolor=red}`.

5.5.1 \psline

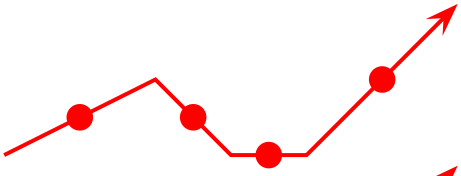


```
\psline[ArrowInside=->]{|<->|}(2,1)
```

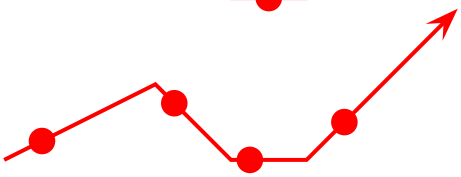
```
\psline[ArrowInside=-|]{|-|}(0,0)(2,1)
```

```
\psline[ArrowInside=->,ArrowInsideNo=2]{->}(0,0)(2,1)
```

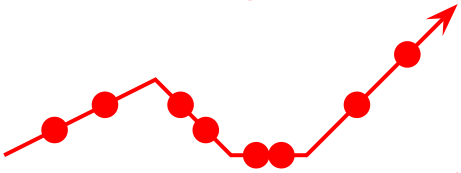
```
\psline[ArrowInside=->,%
   ArrowInsideNo=2,%
   ArrowInsideOffset=0.1]{->}(0,0)(2,1)
```



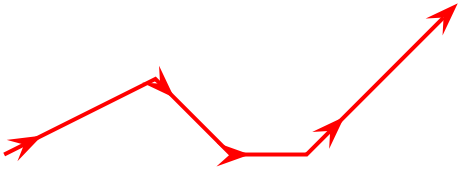
```
\psline[ArrowInside=-*]{->}%
   (0,0)(2,1)(3,0)(4,0)(6,2)
```



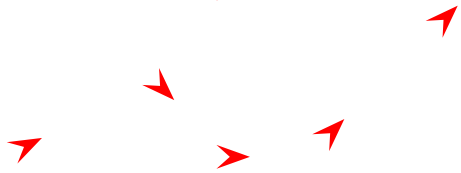
```
\psline[ArrowInside=-*,%
   ArrowInsidePos=0.25]{->}%
      (0,0)(2,1)(3,0)(4,0)(6,2)
```



```
\psline[ArrowInside=-*,%
   ArrowInsidePos=0.25,%
   ArrowInsideNo=2]{->}%
      (0,0)(2,1)(3,0)(4,0)(6,2)
```

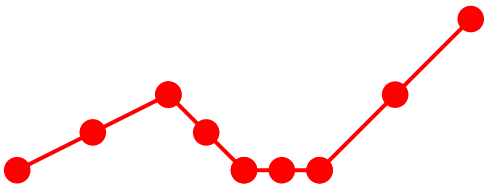


```
\psline[ArrowInside=->,%
   ArrowInsidePos=0.25]{->}%
      (0,0)(2,1)(3,0)(4,0)(6,2)
```

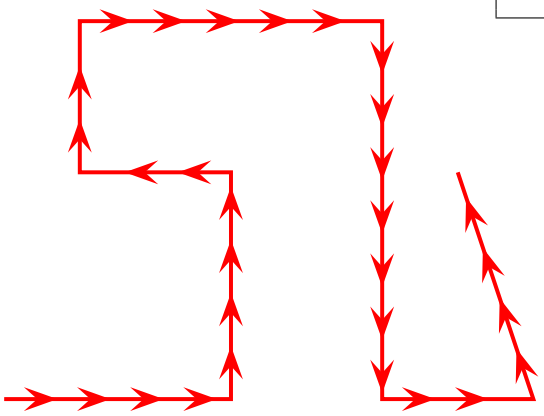


```
\psline[linestyle=none,%
   ArrowInside=->,%
   ArrowInsidePos=0.25]{->}%
      (0,0)(2,1)(3,0)(4,0)(6,2)
```

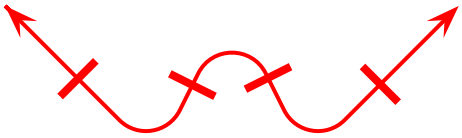
```
\psline[ArrowInside=-<,%
   ArrowInsidePos=0.75]{->}%
      (0,0)(2,1)(3,0)(4,0)(6,2)
```



```
\psset{ArrowInside=-*}
\psline(0,0)(2,1)(3,0)(4,0)(6,2)
\psset{linestyle=none}
\psline[ArrowInsidePos=0]%
   (0,0)(2,1)(3,0)(4,0)(6,2)
\psline[ArrowInsidePos=1]%
   (0,0)(2,1)(3,0)(4,0)(6,2)
```

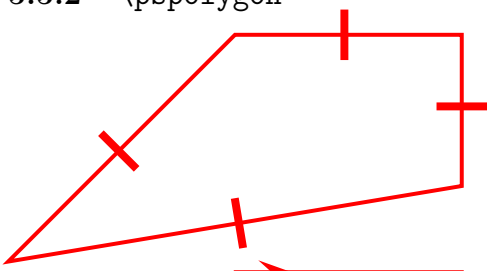


```
\psline[ArrowInside=->,%
   ArrowInsidePos=20](0,0)(3,0)%
     (3,3)(1,3)(1,5)(5,5)%
     (5,0)(7,0)(6,3)
```

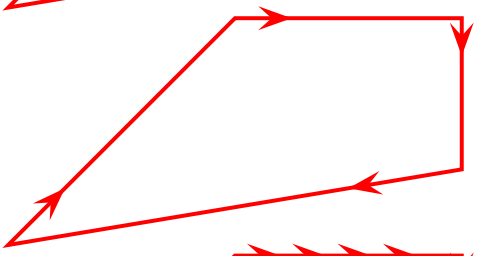


```
\psline[linearc=0.5,%
    ArrowInside=-|]{<->}%
    (0,2)(2,0)(3,2)(4,0)(6,2)
```

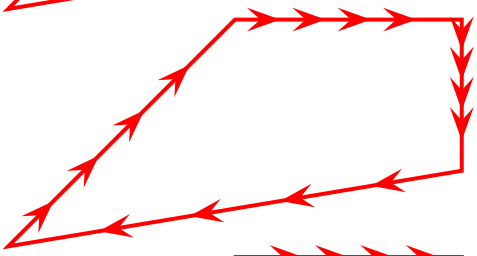
5.5.2 \pspolygon



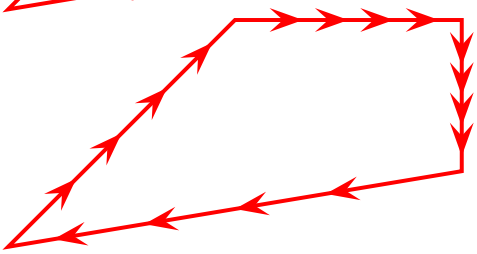
```
\pspolygon[ArrowInside=-|]%
    (0,0)(3,3)(6,3)(6,1)
```



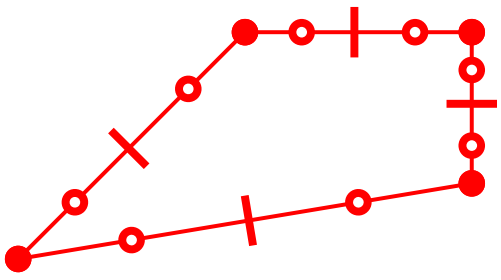
```
\pspolygon[ArrowInside=->,%
   ArrowInsidePos=0.25]%
     (0,0)(3,3)(6,3)(6,1)
```



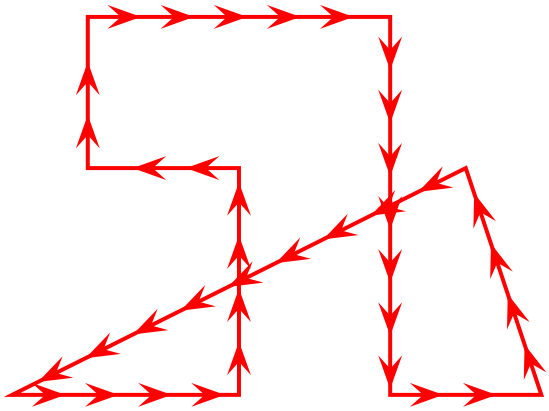
```
\pspolygon[ArrowInside=->,%
   ArrowInsideNo=4]%
     (0,0)(3,3)(6,3)(6,1)
```



```
\pspolygon[ArrowInside=->,%
   ArrowInsideNo=4,%
   ArrowInsideOffset=0.1]%
     (0,0)(3,3)(6,3)(6,1)
```

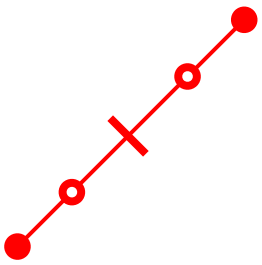


```
\pspolygon[ArrowInside=-|]%
   (0,0)(3,3)(6,3)(6,1)
\psset{linestyle=none,ArrowInside=-*}
\pspolygon[ArrowInsidePos=0]%
   (0,0)(3,3)(6,3)(6,1)
\pspolygon[ArrowInsidePos=1]%
   (0,0)(3,3)(6,3)(6,1)
\psset{ArrowInside=-o}
\pspolygon[ArrowInsidePos=0.25]%
   (0,0)(3,3)(6,3)(6,1)
\pspolygon[ArrowInsidePos=0.75]%
   (0,0)(3,3)(6,3)(6,1)
```

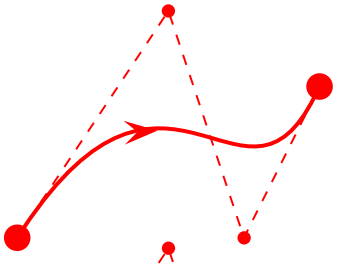


```
\pspolygon[ArrowInside=->,%
    ArrowInsidePos=20](0,0)(3,0)%
      (3,3)(1,3)(1,5)(5,5)%
      (5,0)(7,0)(6,3)
```

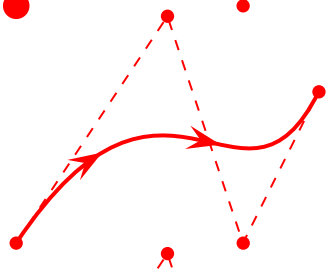
5.5.3 \psbezier



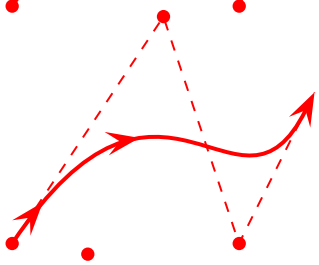
```
\psbezier[ArrowInside=-|](1,1)(2,2)(3,3)
\psset{linestyle=none,ArrowInside=-o}
\psbezier[ArrowInsidePos=0.25](1,1)(2,2)(3,3)
\psbezier[ArrowInsidePos=0.75](1,1)(2,2)(3,3)
\psset{linestyle=none,ArrowInside=-*}
\psbezier[ArrowInsidePos=0](1,1)(2,2)(3,3)
\psbezier[ArrowInsidePos=1](1,1)(2,2)(3,3)
```



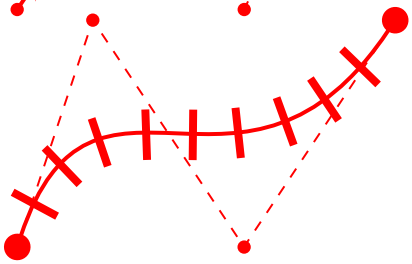
```
\psbezier[ArrowInside=->,%
    showpoints=true]%
       {*-*}(2,3)(3,0)(4,2)
```



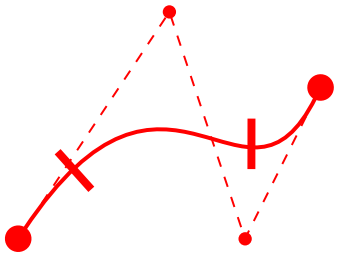
```
\psbezier[ArrowInside=->,%
    showpoints=true,%
    ArrowInsideNo=2](2,3)(3,0)(4,2)
```



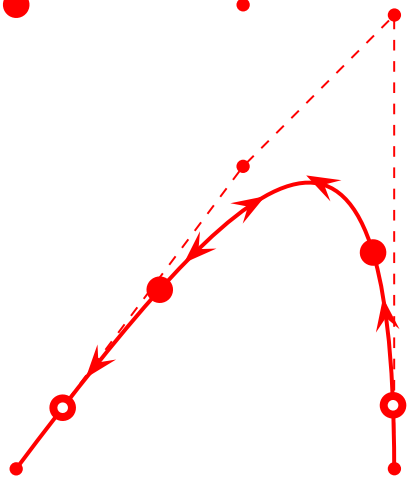
```
\psbezier[ArrowInside=->,%
    showpoints=true,%
    ArrowInsideNo=2,%
    ArrowInsideOffset=-0.2]{->}(2,3)(3,0)(4,2)
```



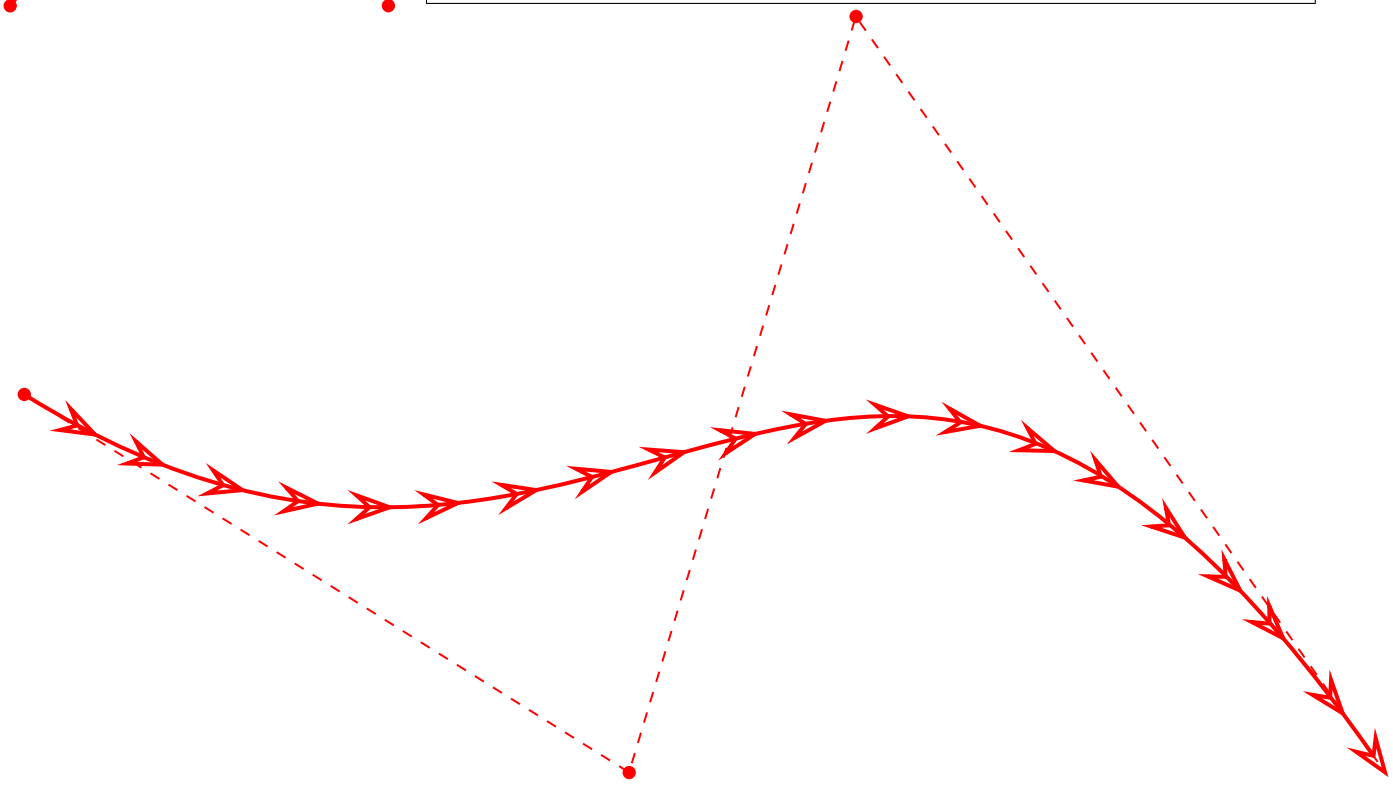
```
\psbezier[ArrowInsideNo=9,%
    ArrowInside=-|, showpoints=true]%
       {*-*}(1,3)(3,0)(5,3)
```



```
\psset{ArrowInside=-|}
\psbezier[ArrowInsidePos=0.25,%
     showpoints=true]{*-*}(2,3)(3,0)(4,2)
\psset{linestyle=none}
\psbezier[ArrowInsidePos=0.75](2,3)(3,0)(4,2)
```



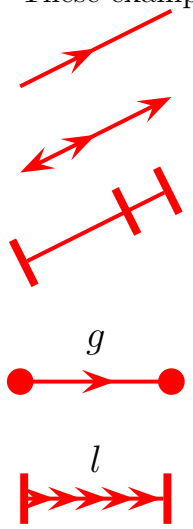
```
\pnode(3,4){A}\pnode(5,6){B}\pnode(5,0){C}
\psbezier[ArrowInside=->,%
   showpoints=true](A)(B)(C)
\psset{linestyle=none,ArrowInside=-<}
\psbezier[ArrowInsideNo=4](A)(B)(C)
\psset{ArrowInside=-o}
\psbezier[ArrowInsidePos=0.1](A)(B)(C)
\psbezier[ArrowInsidePos=0.9](A)(B)(C)
\psset{ArrowInside=-*}
\psbezier[ArrowInsidePos=0.3](A)(B)(C)
\psbezier[ArrowInsidePos=0.7](A)(B)(C)
```



```
\psbezier[ArrowInsideNo=19,%
    ArrowInside=->,%
    showpoints=true]{->}(-3,0)(5,-5)(8,5)(15,-5)
```

5.5.4 \pcline

These examples need the package pst-node.



```
\pcline[ArrowInside=->](0,0)(2,1)
```

```
\pcline[ArrowInside=->]{<->}(0,0)(2,1)
```

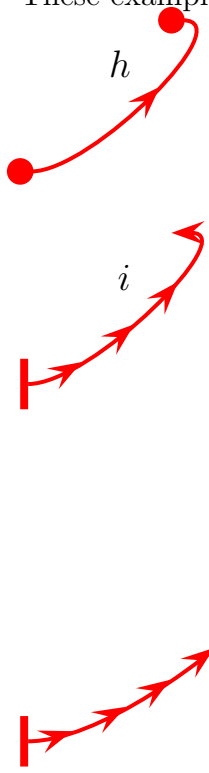
```
\pcline[ArrowInside=-|,%
   ArrowInsidePos=0.75]{|-|}(0,0)(2,1)
```

```
\pcline[ArrowInside=->,%
   ArrowInsidePos=0.65]{*-*}(0,0)(2,0)
\naput[labelsep=0.3]{\large$g$}
```

```
\pcline[ArrowInside=->,ArrowInsidePos=10]{|-|}(0,0)(2,0)
\naput[labelsep=0.3]{\large$l$}
```

5.5.5 \pccurve

These examples also need the package pst-node.



```
\pccurve[ArrowInside=->,%
   ArrowInsidePos=0.65,%
   showpoints=true]{*-*}(0,0)(2,2)
\naput[labelsep=0.3]{\large$h$}
```

```
\pccurve[ArrowInside=->,%
   ArrowInsideNo=3,%
   showpoints=true]{|->}(0,0)(2,2)
\naput[labelsep=0.3]{\large$i$}
```

```
\pccurve[ArrowInside=->,%
   ArrowInsidePos=20]{|-|}(0,0)(4,4)
\naput[labelsep=0.3]{\large$k$}
```

6 \psFormatInt

There exist some packages and a lot of code to format an integer like 1 000 000 or 1, 234, 567 (in Europe 1.234.567). But all packages expect a real number as argument and cannot handle

macros as an argument. For this case `pstricks-add` has a macro `psFormatInt` which can handle both:

1,234,567
1,234,567
1.234.567
1·234·567
965,432

```
\psFormatInt{1234567}\\
\psFormatInt[intSeparator={,}]{1234567}\\
\psFormatInt[intSeparator=.]{1234567}\\
\psFormatInt[intSeparator=$\cdot$]{1234567}\\
\def\temp{965432}
\psFormatInt{\temp}
```

With the option `intSeparator` the symbol can be changed to any possible character.

Part II
pst-node

7 \nclineII

The dashed lines are by default black and white lines. The new macro `\nclineII` offers two-color lines and has the same syntax as `\ncline`:

`\ncline[<options>]{<Node A>}{<Node B>}`


```
\circlenode[linecolor=blue,linewidth=2pt]{A}{A}%
\hspace{9cm}\circlenode[linecolor=cyan,linewidth=2pt]{B}{B}
\nclineII[linewidth=5pt]{A}{B}
```

7.1 The options

These options are all defined in the package `pstricks-add`.

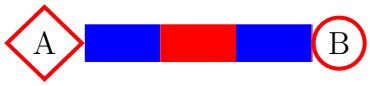
name	meaning
`dashColorI`	first color, default is `black`
`dashColorII`	second color, default is `red`
`dashNo`	the difference in per cent of the colored lines, default ist 0.2

`dashNo` can have values greater than 1. In this case the value will be taken as an absolute width in the pt unit. Only this unit is possible!

7.2 Examples


```
\circlenode{A}{A}%
\hspace{3cm}%
\dianode{B}{B}%
\nclineII[linewidth=8pt,dashColorI=blue]{A}{B}
```


```
\circlenode{A}{A}%
\hspace{3cm}%
\circlenode{B}{B}%
\nclineII[dashColorI=blue,%
   linewidth=3pt,dashNo=15]{->}{A}{B}
```

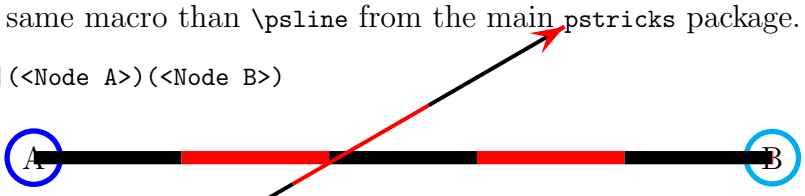



```
\dianode{A}{A}%
\hspace{3cm}%
\circlenode{B}{B}%
\nclineII[dashColorI=blue,%
    linecap=1,%
    dashNo=0.3,linewidth=0.5]{A}{B}
```

8 \pclineII

This is nearly the same macro than `\psline` from the main `pstricks` package.

```
\pcline[<options>](<Node A>)(<Node B>)
```




```
\circlenode[linecolor=blue,linewidth=2pt]{A}{A}%
\hspace{9cm}\circlenode[linecolor=cyan,linewidth=2pt]{B}{B}
\pclineII[linewidth=5pt](A)(B)
```

This macro makes only sense when connecting two "invisible" nodes, like this connection from here to the above word `pstricks`.

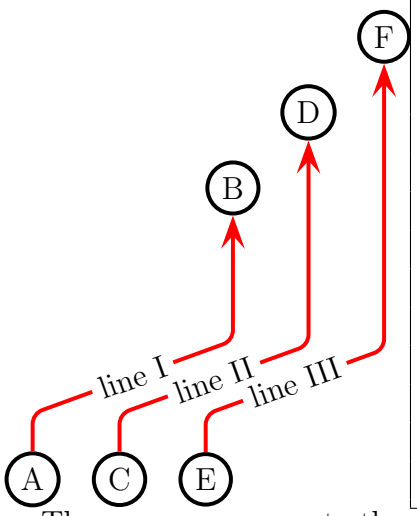
```
This macro makes only sense when connecting two
''invisible`` nodes, like this connection from
here\pnode{D}\pclineII{->}(D)(C){} to the above word \verb|pstricks|.
```

9 \ncdiag and \pcdiag

With the new option `lineAngle` the lines drawn by the `ncdiag` macro can now have a specified gradient. Without this option one has to define the two arms (which maybe zero) and PSTricks draws the connection between them. Now there is only a static `armA`, the second one `armB` is dynamically when an angle `lineAngle` is defined. This angle is the gradient of the intermediate line between the two arms. The syntax of `ncdiag` is

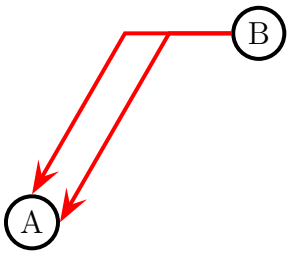
```
\ncdiag[<options>]{<Node A>}{<Node B>}
\pcdiag[<options>](<Node A>)(<Node B>)
```

name	meaning
`lineAngle`	angle of the intermediate line segment. Default is 0, which is the same than using `ncdiag` without the `lineAngle` option.

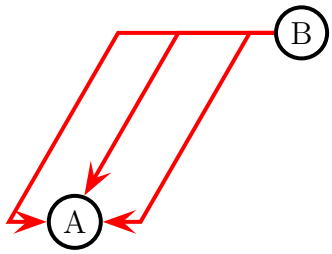



```
\psset{linecolor=black}
\circlenode{A}{A}%
\quad\circlenode{C}{C}%
\quad\circlenode{E}{E}
\rput(0,4){\circlenode{B}{B}}
\rput(1,5){\circlenode{D}{D}}
\rput(2,6){\circlenode{F}{F}}
{\psset{arrowscale=2,linearc=0.2,%
  linecolor=red,armA=0.5, angleA=90,angleB=-90}
\ncdiag[lineAngle=20]{->}{A}{B}
\ncput*[nrot=:U]{line I}
\ncdiag[lineAngle=20]{->}{C}{D}
\ncput*[nrot=:U]{line II}
\ncdiag[lineAngle=20]{->}{E}{F}
\ncput*[nrot=:U]{line III}}
```

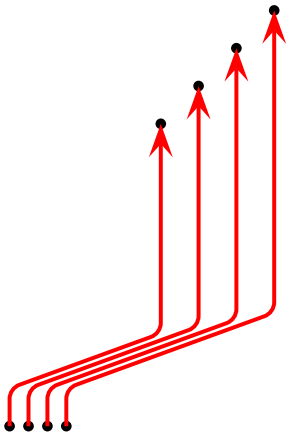
The `ncdiag` macro sets the `armB` dynamical to the calculated value. Any user setting of this `armB` is overwritten by the macro. The `armA` could be set to a zero length:




```
\psset{linecolor=black}
\rput(0.5,0.5){\circlenode{A}{A}}
\rput(3.5,3){\circlenode{B}{B}}
{\psset{linecolor=red,arrows=<-,arrowscale=2}
\ncdiag[lineAngle=60,%
    armA=0,angleA=0,angleB=180]{A}{B}
\ncdiag[lineAngle=60,%
    armA=0,angleA=90,angleB=180]{A}{B}}
```




```
\psset{linecolor=black}
\rput(1,0.5){\circlenode{A}{A}}
\rput(4,3){\circlenode{B}{B}}
{\psset{linecolor=red,arrows=<-,arrowscale=2}
\ncdiag[lineAngle=60,%
    armA=0.5,angleA=0,angleB=180]{A}{B}
\ncdiag[lineAngle=60,%
    armA=0,angleA=70,angleB=180]{A}{B}
\ncdiag[lineAngle=60,%
    armA=0.5,angleA=180,angleB=180]{A}{B}}
```

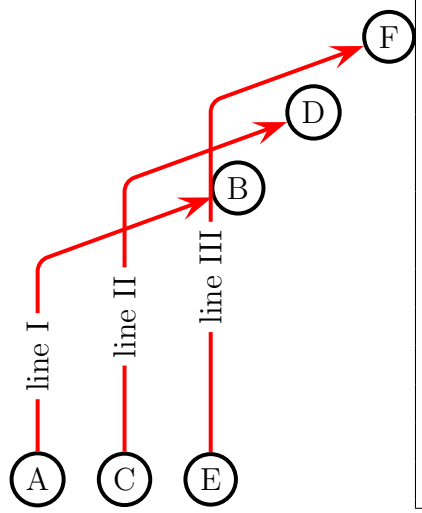


```
\psset{linecolor=black}
\cnode*(0,0){2pt}{A}%
\cnode*(0.25,0){2pt}{C}%
\cnode*(0.5,0){2pt}{E}%
\cnode*(0.75,0){2pt}{G}%
\cnode*(2,4){2pt}{B}%
\cnode*(2.5,4.5){2pt}{D}%
\cnode*(3,5){2pt}{F}%
\cnode*(3.5,5.5){2pt}{H}%
{\psset{arrowscale=2,linearc=0.2,%
  linecolor=red,armA=0.5, angleA=90,angleB=-90}
\pcdiag[lineAngle=20]{->}(A)(B)
\pcdiag[lineAngle=20]{->}(C)(D)
\pcdiag[lineAngle=20]{->}(E)(F)
\pcdiag[lineAngle=20]{->}(G)(H)}
```

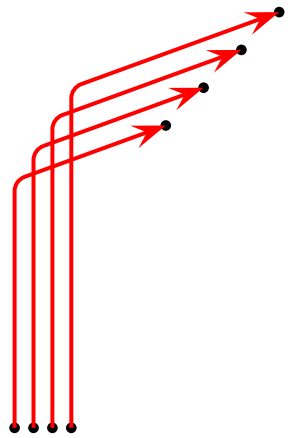
10 \ncdiagg and \pcdiagg

This is nearly the same than `\ncdiag` except that `armB=0` and the `angleB` value is conputed by the macro, so that the line ends at the node with an angle like a `\pcdiagg` line. The syntax of `ncdiagg/pcdiagg` is

```
\ncdiag[<options>]{<Node A>}{<Node B>}
\pcdiag[<options>](<Node A>)(<Node B>)
```

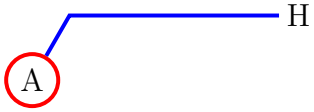



```
\psset{linecolor=black}
\circlenode{A}{A}%
\quad\circlenode{C}{C}%
\quad\circlenode{E}{E}
\rput(0,4){\circlenode{B}{B}}
\rput(1,5){\circlenode{D}{D}}
\rput(2,6){\circlenode{F}{F}}
{\psset{arrowscale=2,linearc=0.2,%
  linecolor=red,armA=0.5, angleA=90}
\ncdiagg[lineAngle=-160]{->}{A}{B}
\ncput*[nrot=:U]{line I}
\ncdiagg[lineAngle=-160]{->}{C}{D}
\ncput*[nrot=:U]{line II}
\ncdiagg[lineAngle=-160]{->}{E}{F}
\ncput*[nrot=:U]{line III}}
```



```
\psset{linecolor=black}
\cnode*(0,0){2pt}{A}%
\cnode*(0.25,0){2pt}{C}%
\cnode*(0.5,0){2pt}{E}%
\cnode*(0.75,0){2pt}{G}%
\cnode*(2,4){2pt}{B}%
\cnode*(2.5,4.5){2pt}{D}%
\cnode*(3,5){2pt}{F}%
\cnode*(3.5,5.5){2pt}{H}%
{\psset{arrowscale=2,linearc=0.2,%
  linecolor=red,armA=0.5, angleA=90}
\pcdiagg[lineAngle=20]{->}(A)(B)
\pcdiagg[lineAngle=20]{->}(C)(D)
\pcdiagg[lineAngle=20]{->}(E)(F)
\pcdiagg[lineAngle=20]{->}(G)(H)}
```

The only problem for `\ncdiagg` is, that you need the right value for `lineAngle`. If the node connection is on the wrong side of the second node, then choose the corresponding angle, e.g.: if 20° is wrong then take −160°, the corresponding to 180°.




```
\circlenode{a}{A}
\rput[l](3,1){\rnode{b}{H}}
\ncdiagg[lineAngle=60,angleA=180,armA=.5,
   nodesepA=3pt,linecolor=blue]{b}{a}
```

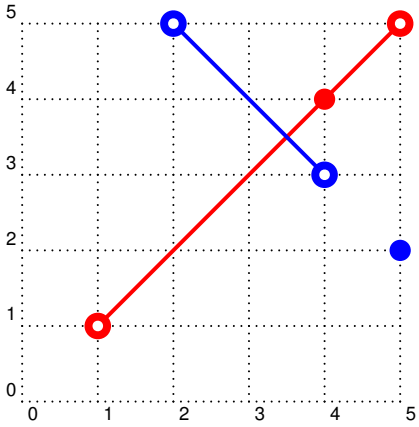

```
\circlenode{a}{A}
\rput[l](3,1){\rnode{b}{H}}
\ncdiagg[lineAngle=60,armA=.5,
   nodesepB=3pt,linecolor=blue]{a}{b}
```


```
\circlenode{a}{A}
\rput[l](3,1){\rnode{b}{H}}
\ncdiagg[lineAngle=-120,armA=.5,
   nodesepB=3pt,linecolor=blue]{a}{b}
```

11 \psLNode and \psLCNode

`\psLNode` interpolates the Line $\overline{AB}$ by the given value and sets a node at this point. The syntax is

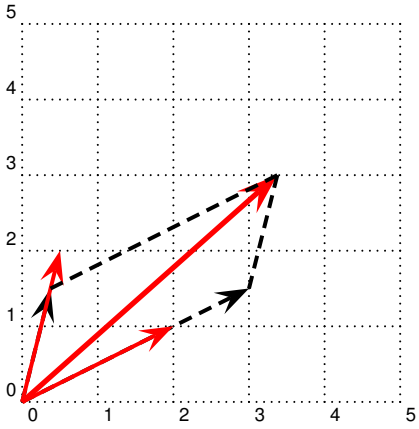
```
\setLNode(P1)(P2){value}{Node name}
```



```
\begin{pspicture}(5,5)
\psgrid[subgriddiv=0,griddots=10]
\psset{linecolor=red}
\psline{o-o}(1,1)(5,5)
\setLNode(1,1)(5,5){0.75}{PI}
\qdisk(PI){4pt}
\psset{linecolor=blue}
\psline{o-o}(4,3)(2,5)
\setLNode(4,3)(2,5){-0.5}{PII}
\qdisk(PII){4pt}
\end{pspicture}
```

The \psLCNode macro builds the linear combination of the two given vectors and stores the end of the new vector as a node. All vectors start at $(0, 0)$, so a \rput maybe appropriate. The syntax is

```
\setLCNode(P1){value 1}(P2){value 2}{Node name}
```

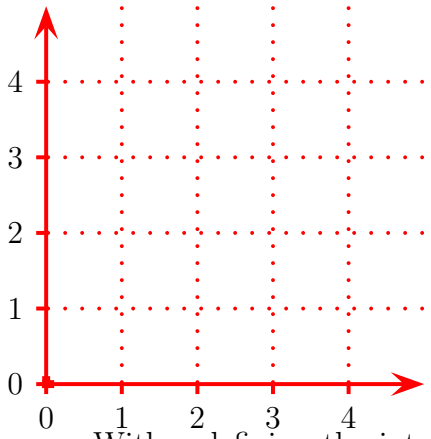


```
\begin{pspicture}(5,5)
\psgrid[subgriddiv=0,griddots=10]
\psset{linecolor=black}
\psline[linestyle=dashed]{->}(3,1.5)
\psline[linestyle=dashed]{->}(0.375,1.5)
\psset{linecolor=red}
\psline{->}(2,1)\psline{->}(0.5,2)
\setLCNode(2,1){1.5}(0.5,2){0.75}{PI}
\psline[linewidth=2pt]{->}(PI)
\psset{linecolor=black}
\psline[linestyle=dashed](3,1.5)(PI)
\psline[linestyle=dashed](0.375,1.5)(PI)
\end{pspicture}
```

Part III
pst-plot

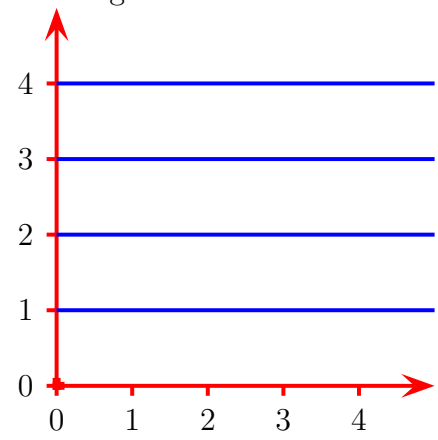
12 New option ticklines

With a grid it is possible to have „ticklines“ but only for both axes. With the option `ticklines` the values `x|y|all` are possible. By default the lines are dotted (`arrows=-,linestyle=dotted,dotsep=5pt`).




```
\begin{pspicture}(5,5)
  \psaxes[tickstyle=bottom,ticklines=all]{->}(5,5)
\end{pspicture}
```

With redefining the internal styles `\ticklinesXStyle` and `\ticklinesYStyle` this style can be changed.




```
\newpsstyle{ticklinesYStyle}{linestyle=solid,%
   linecolor=blue,arrows=-}
\begin{pspicture}(5,5)
  \psaxes[tickstyle=bottom,ticklines=y]{->}(5,5)
\end{pspicture}
```

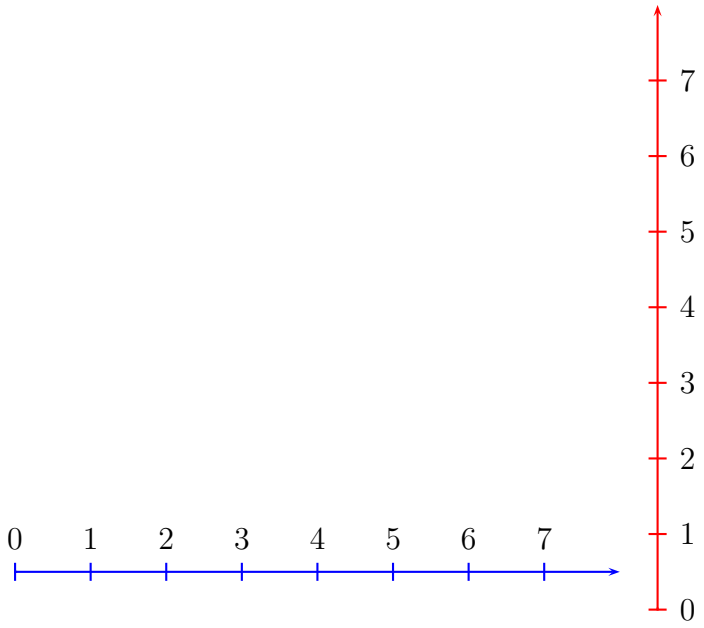
13 New macro \resetOptions

Sometimes it is difficult to know what options which are changed inside a long document are different to the default one. With this macro all options depending to `pst-plot` can be reset. This depends to all options of the packages `pstricks`, `pst-plot` and `pst-node`.

14 New options xyAxes, xAxes and yAxes

Sometimes there is only a need for one axes with ticks. In this case you can set one of the following options to false. The `xyAxes` makes only sense, when you want to set both, x and y to true with only one command again to the default, because with `xyAxes=false` you get nothing with the `psaxes` macro.

Name	Setting
`xyAxes`	default is `true`
`xAxes`	default is `true`
`yAxes`	default is `true`



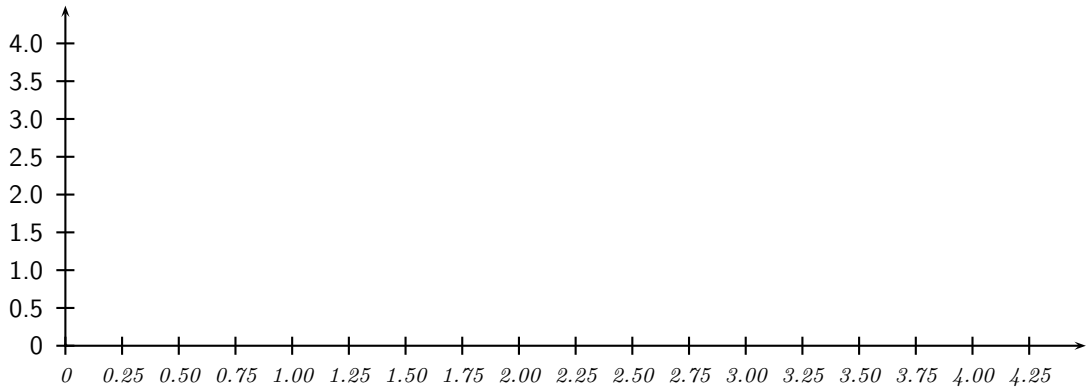

```
\resetOptions
\begin{pspicture}(8,1)
\psaxes[yAxes=false,linecolor=blue]{->}(0,0.5)(8,0.5)
\end{pspicture}%
\begin{pspicture}(1,8)
\psaxes[xAxes=false,linecolor=red]{->}(0.5,0)(0.5,8)
\end{pspicture}
```

15 New options xyLabel, xLabel and yLabel

There are no special keywords to change the labelstyle for the `\psaxes` macro. `pst-plot-add` defines two options:

Name	Setting
`xyLabel`	default is `{}`
`xLabel`	default is `{}`
`yLabel`	default is `{}`

With `xyLabel` it is possible to set both axes with the same command sequence. In difference to the default `pst-plot` package the coordinates are not printed in mathmode. This makes it easier to choose other different textstyles.



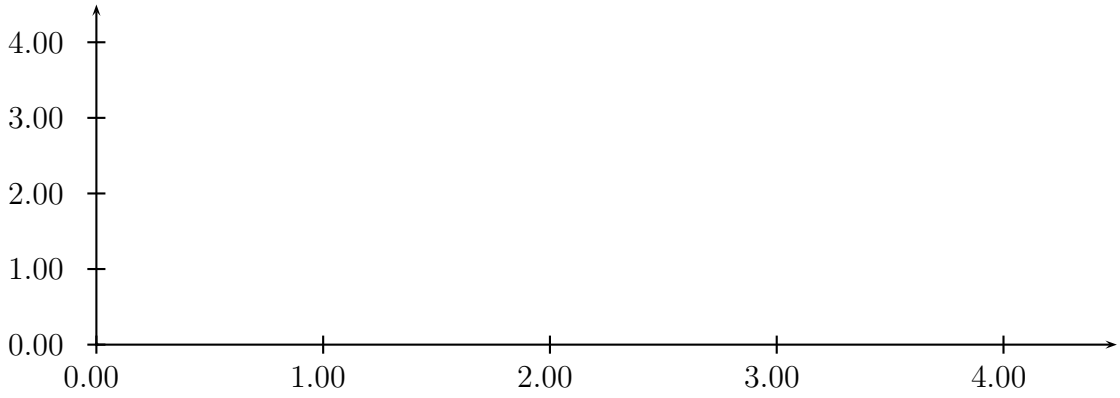

```
{\psset{yunit=1cm,xunit=3cm}
\begin{pspicture}(-0.2,-0.5)(5,4.75)
\psaxes[xLabel={\scriptsize\itshape},%
   yLabel={\sffamily\footnotesize},%
   Dy=0.5, Dx=0.25]{->}(0,0)(4.5,4.5)
\end{pspicture}}
```

16 New options `xyDecimals`, `xDecimals` and `yDecimals`

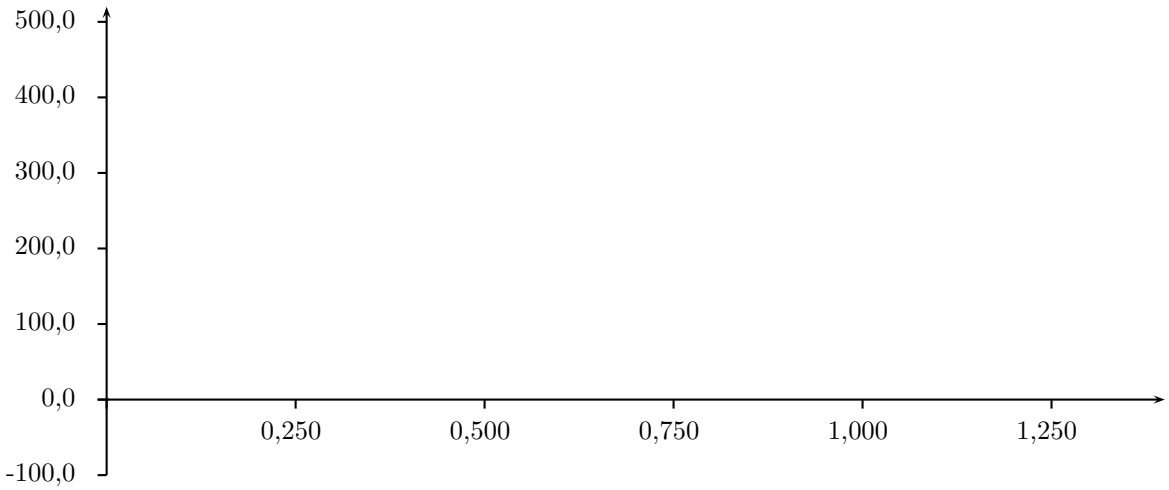
By default the labels of the axes get numbers with or without decimals, just depending to the numbers. With these options `??Decimals` it is possible to determine the decimals, where the option `xyDecimals` sets this identical for both axes.

Name	Setting
`xyDecimals`	default is `{}`
`xDecimals`	default is `{}`
`yDecimals`	default is `{}`

The default setting `{}` means, that you'll get the standard behaviour.



```
\begin{pspicture}(-0.2,-0.5)(5,4.75)
 \psaxes[xyDecimals=2]{->}(0,0)(4.5,4.5)
\end{pspicture}
```

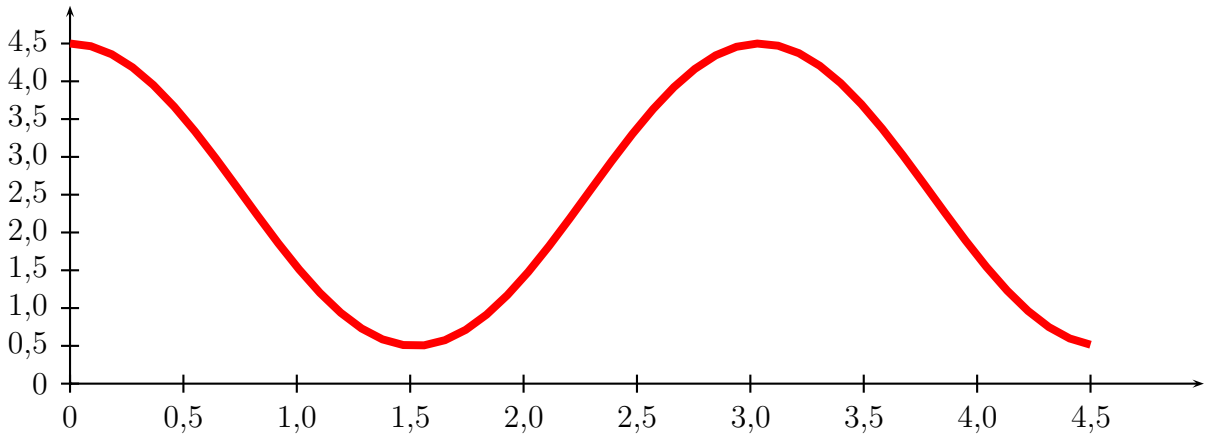



```
\begin{pspicture}(-0.1,-150)(1.5,550.0)
 \psaxes[Dx=0.25,Dy=100, tickstyle=bottom,%
  xyLabel=\footnotesize,comma,xDecimals=3,yDecimals=1]%
  {->}(0,0)(0,-100)(1.4,520)
\end{pspicture}
```

17 New option comma

Setting this option to true gives labels with a comma as a decimal separator instead of the dot. comma and comma=true is the same.

```
\begin{pspicture}(0,-0.5)(5,5.5)
\psaxes[xunit=3cm,Dx=0.5,Dy=0.5,comma]{->}(5,5)
\psplot[xunit=3cm,linecolor=red,linewidth=3pt]%
    {0}{4.5}{x 180 mul 1.52 div cos 2 mul 2.5 add}
\end{pspicture}
```



18 New options `logBase`, `xlogBase` and `ylogBase`

There are additional options `logBase` xlogBase | ylogBase| to get one or both axes with logarithm labels.

Name	Setting
`logBase`	default is `{}`
`xlogBase`	default is `{}`
`ylogBase`	default is `{}`

For an intervall of $[10^{-3}...10^{2}]$ choose a `pstricks` intervall of [-3,2]. `pstricks` takes 0 as the origin of this axes, which is wrong if we want to have a logarithm axes. With the options `Oy` and `Ox` we can set the origin to -3, so that the first label gets 10^{-3}. If this is not done by the user than `pstricks-add` does it by default. An alternative is to set these parameters to empty values `Ox={},Oy={}`, in this case `pstricks-add` does nothing.

18.1 y axis (`ylogBase`)

Figure 1 shows the graph of the function $y = \log x$ with a logarithmic y-axis. The code is:

```
\begin{pspicture}(-0.5,-3.5)(6.5,1.5)
 \psplot[linewidth=2pt,%
 plotpoints=100,linecolor=red]{1.001}{6}{x log log} % log(x)
 \psaxes[ylogBase=10]{<->}(0,-3)(6.5,1.5)
 \uput[-90](6.5,-3){x}
 \uput[180](0,1.5){y}
 \rput(5,1){$y=\log x$}
\end{pspicture}
```

The values for the `psaxes` y-coordinate are now the exponents to the base 10 and for the right function to the base e: $10^{-3} \ldots 10^{1}$ which corresponds to the given y-intervall $-3 \ldots 1.5$, where only integers as exponents are possible. These logarithm labels have no effect to the internal used units. To draw the logarithm function we have to use the math function

$$y = \log\{\log x\}$$

$$y = \ln\{\ln x\}$$

with an drawing intervall of $1.001 \ldots 6$.

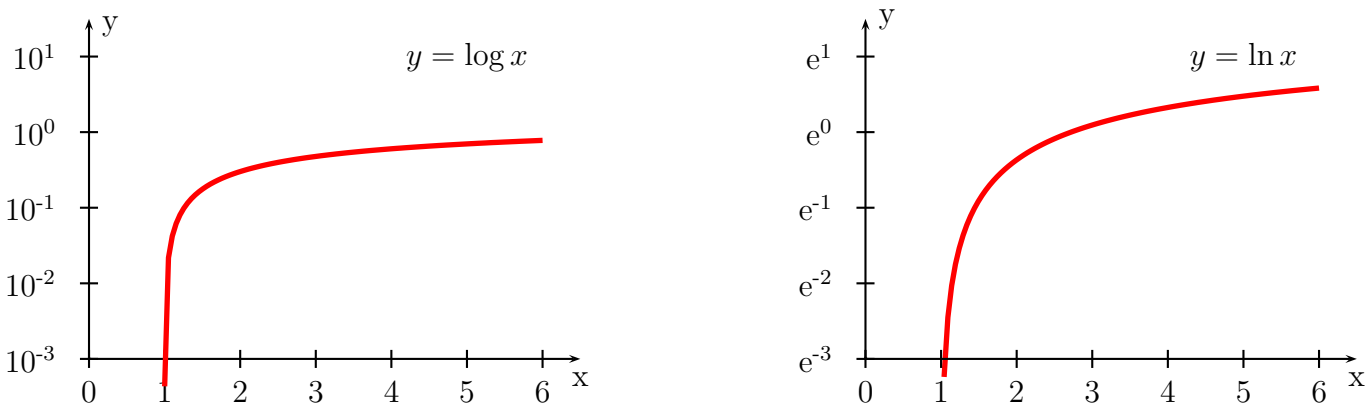


Figure 1: log axes usage: (`ylogBase=10` and ylogBase=e)

18.2 x axis (`xlogBase`)

Figure 2 shows the same for the x axis. Now we have to use the easy math function

$$y = x$$

because the x axis is still $\log x$.

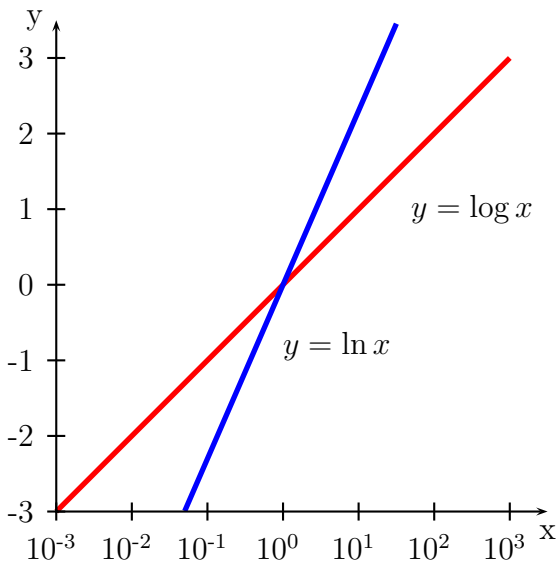
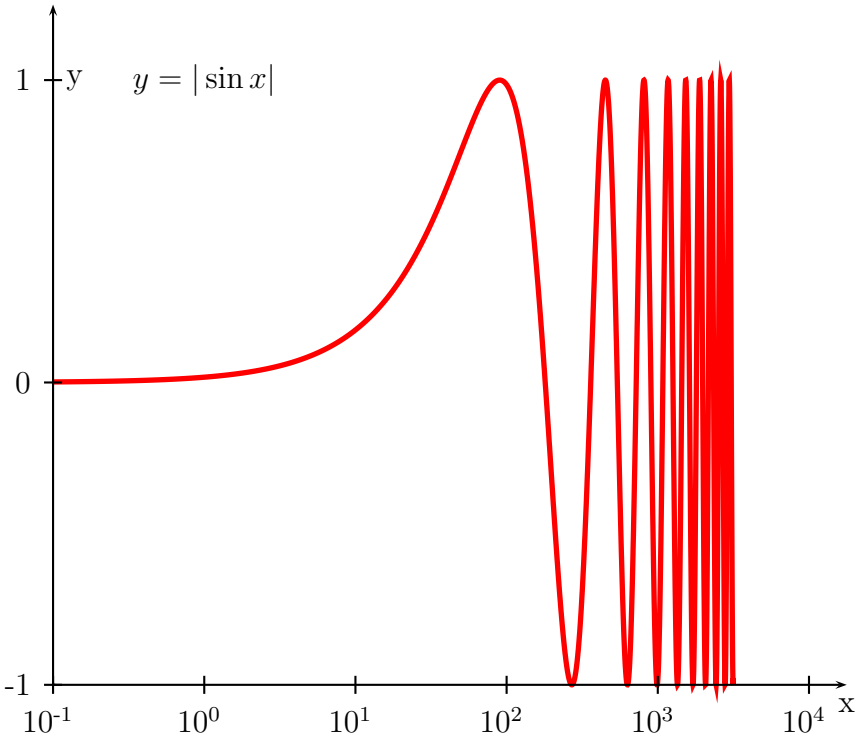


Figure 2: log axes usage: (`xlogBase=10`)

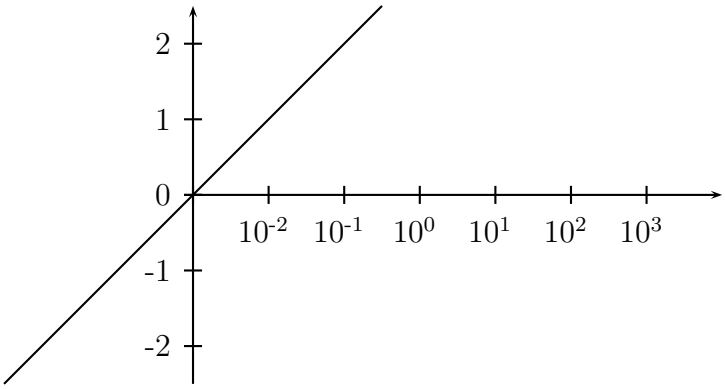
The code for figure 2:

```
\begin{pspicture}(-3.5,-3.5)(3.5,3.5)
  \psplot[linewidth=2pt,linecolor=red]{-3}{3}{x} % log(x)
  \psplot[linewidth=2pt,linecolor=blue]{-1.3}{1.5}{x 0.4343 div} % ln(x)
```

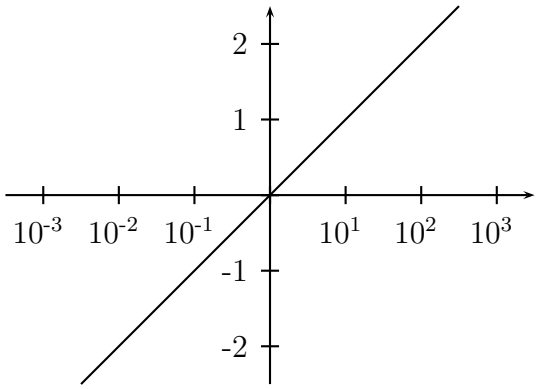
```
    \psaxes[xlogBase=10,Oy=-3]{<->}(-3,-3)(3.5,3.5)
    \uput[-90](3.5,-3){x}
    \uput[180](-3,3.5){y}
    \rput(2.5,1){$y=\log x$}
    \rput[lb](0,-1){$y=\ln x$}
  \end{pspicture}
```




```
{\psset{yunit=4cm,xunit=2cm}
\begin{pspicture}(-1.25,-1.25)(4.25,1.5)
 \uput[-90](4.25,-1){x}
 \uput[0](-1,1){y}
 \rput(0,1){$y=|\sin x|$}
 \psplot[linewidth=2pt,%
  plotpoints=5000,linecolor=red]{-1}{3.5}{10 x exp sin } % y=|sin(x)|
 \psaxes[xlogBase=10,logLines=x,Oy=-1]{->}(-1,-1)(4.25,1.25)
\end{pspicture}}
```




```
\begin{pspicture}(-3.5,-2.5)(3.5,2.5)
  \psaxes[xlogBase=10]{->}(0,0)(-3.5,-2.5)(3.5,2.5)
  \psplot{-2.5}{2.5}{10 x exp log}
\end{pspicture}
```




```
\begin{pspicture}(-3.5,-2.5)(3.5,2.5)
  \psaxes[xlogBase=10,Ox={},Oy={}]{->}(0,0)(-3.5,-2.5)(3.5,2.5)
  \psplot{-2.5}{2.5}{10 x exp log}
\end{pspicture}
```

18.3 All axes (logBase)

This mode is in math also called double logarithm. It is a combination of the two forgoing modes and the function is now

$$y = \log x$$

and is shown in figure 3.

The code for figure 3:

```
\begin{pspicture}(-3.5,-3.5)(3.5,3.5)
    \psplot[linewidth=2pt,linecolor=red]{0.001}{3}{x log} % log(x)
    \psaxes[logBase=10,Oy=-3]{<->}(-3,-3)(3.5,3.5)
    \uput[-90](3.5,-3){x}
    \uput[180](-3,3.5){y}
    \rput(2.5,1){$y=\log x$}
\end{pspicture}
```

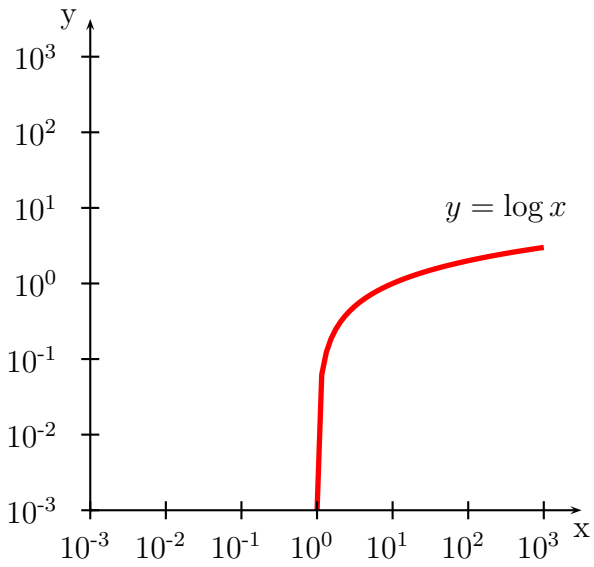


Figure 3: log axes usage: (`xlogBase=10`)

18.4 No logstyle (`logBase={}`)

This is only a demonstration that the default option `logBase={}` still works ... :-)

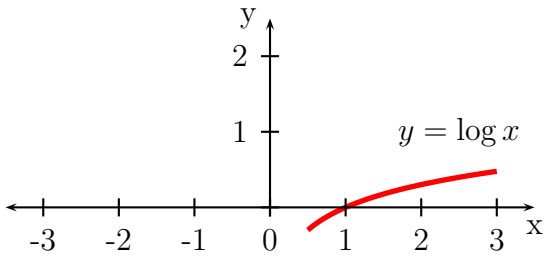


Figure 4: log axes usage: (`logBase={}`)

The code for figure 4:

```
\begin{pspicture}(-3.5,-0.5)(3.5,2.5)
  \psplot[linewidth=2pt,linecolor=red,logBase={}]{0.5}{3}{x log} % log(x)
  \psaxes{<->}(0,0)(-3.5,0)(3.5,2.5)
  \uput[-90](3.5,0){x}
  \uput[180](0,2.5){y}
  \rput(2.5,1){$y=\log x$}
\end{pspicture}
```

18.5 More examples for the `logBase` option

The code for figure 5:

```
\begin{pspicture}(-0.5,1.75)(6.5,4.5)
  \psaxes[ylogBase=10,Oy=2]{<->}(0,2)(0,2)(6.5,4.5)
\end{pspicture}
```

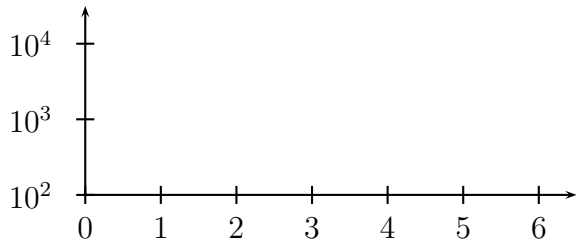


Figure 5: log axes usage: (`ylogBase=10`)

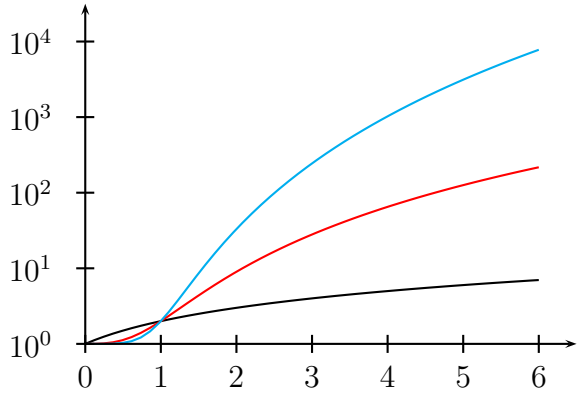


Figure 6: log axes usage: (`ylogBase=10`)

The code for figure 6:

```
\begin{pspicture}(-0.5,-0.25)(6.5,4.5)
  \psplot{0}{6}{x x cos add log}                          % x    + cox(x)
  \psplot[linecolor=red]{0}{6}{x 3 exp x cos add log}  % x^3 + cos(x)
  \psplot[linecolor=cyan]{0}{6}{x 5 exp x cos add log} % x^5 + cos(x)
  \psaxes[ylogBase=10]{<->}(6.5,4.5)
\end{pspicture}
```

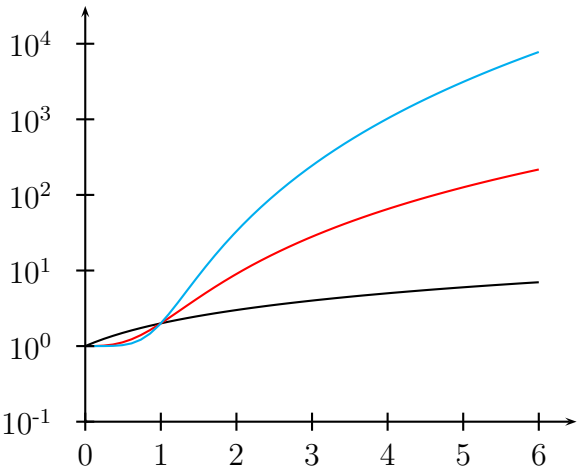


Figure 7: log axes usage: (`ylogBase=10`)

The code for figure 7:

```
\begin{pspicture}(-0.5,-1.25)(6.5,4.5)
  \psplot{0}{6}{x x cos add log}                          % x    + cox(x)
  \psplot[linecolor=red]{0}{6}{x 3 exp x cos add log}  % x^3 + cos(x)
  \psplot[linecolor=cyan]{0}{6}{x 5 exp x cos add log} % x^5 + cos(x)
  \psaxes[ylogBase=10]{<->}(0,-1)(0,-1)(6.5,4.5)
\end{pspicture}
```

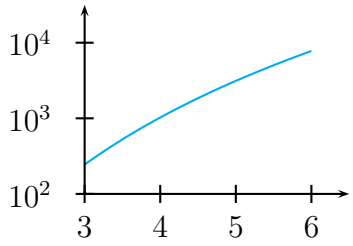


Figure 8: log axes usage: (`ylogBase=10`)

The code for figure 8:

```
\begin{pspicture}(2.5,1.75)(6.5,4.5)
    \psplot[linecolor=cyan]{3}{6}{x 5 exp x cos add log} % x^5 + cos(x)
    \psaxes[ylogBase=10,0x=3,0y=2]{<->}(3,2)(3,2)(6.5,4.5)
\end{pspicture}
```

19 New option `logLines`

The syntax is

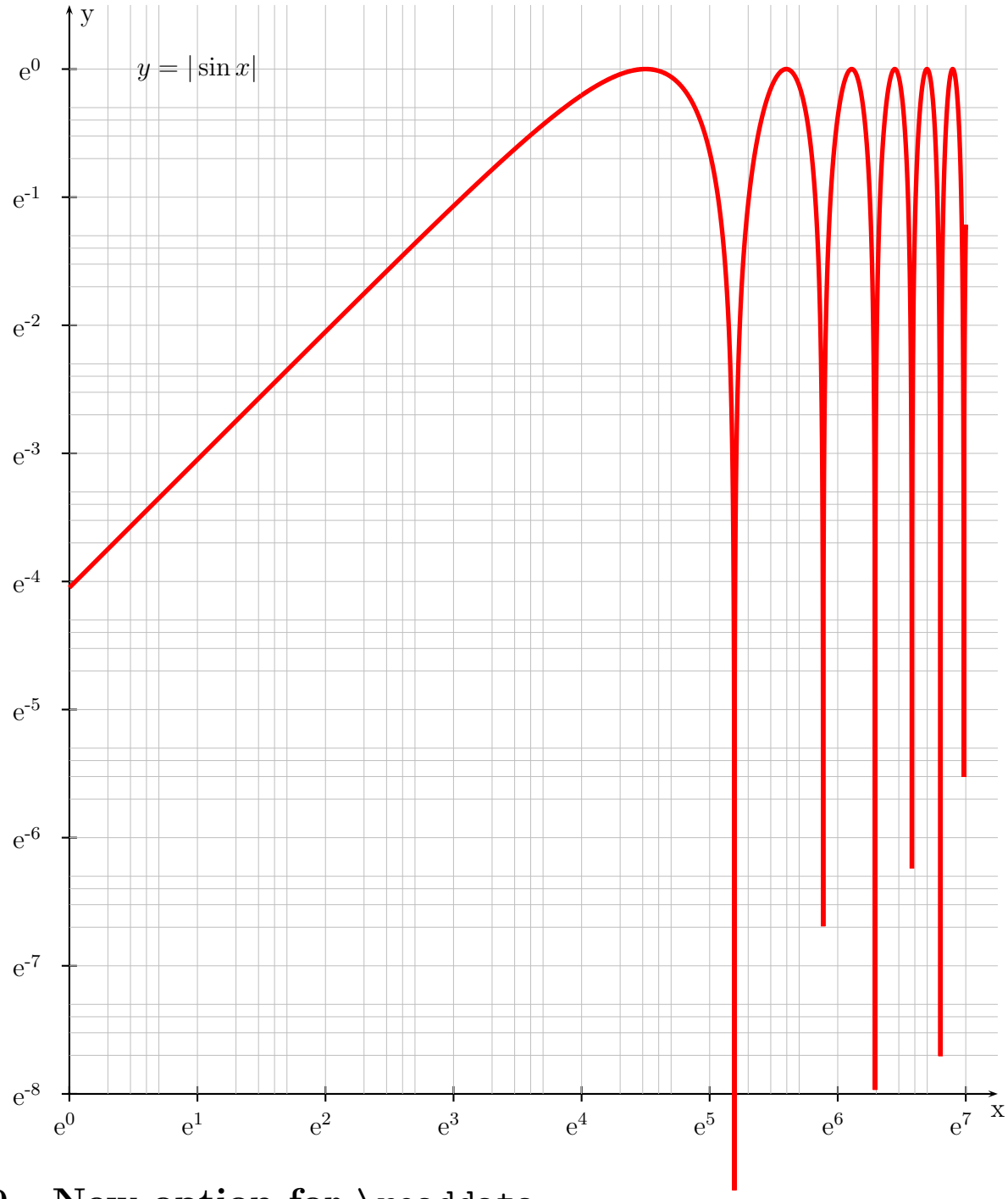
`logLines=all|x|y|none`

with `none` as the default option. With this option there will be 5 lines per unit, at `2,3,4,5`.


```
1 \psaxes[logBase=10,logLines=all]{->}(-1,-4)(3.25,0.5)
```

The default settings for these logarithm lines is

```
\psset{arrows=-,linewidth=0.1pt,linecolor=lightgray}%
```



20 New option for \readdata

By default the macros `\readdata` reads every data record, which could be annoying when there are more than 10000 records to read. The package `pst-plot-add` defines an additional key `nStep`, which allows to read only a selected part of the data records, e.g. `nStep=10`, only every 10^{th} records is saved.

```
1 \readdata[nStep=10]{\dataA}{stressrawdata.dat}
```

The default value for `nStep` is 1.

21 New options for \listplot

By default the plot macros \dataplot, \fileplot and \listplot plot every data record. The package pst-plot-add defines additional keys nStep, nStart, nEnd and xStep, xStart, xEnd, which allows to plot only a selected part of the data records, e.g. nStep=10. These "n" options mark the number of the record to be plot $(0, 1, 2, ...)$ and the "x" ones the x-values of the data records.

Name	Default setting
nStart	1
nEnd	{}
nStep	1
xStart	{}
xEnd	{}
yStart	{}
yEnd	{}
xStep	0
plotNo	1
plotNoMax	1

These new options are only available for the \listplot macro, which is not a real limitation, because all data records can be read from a file with the \readdata macro (see example files or [3]):

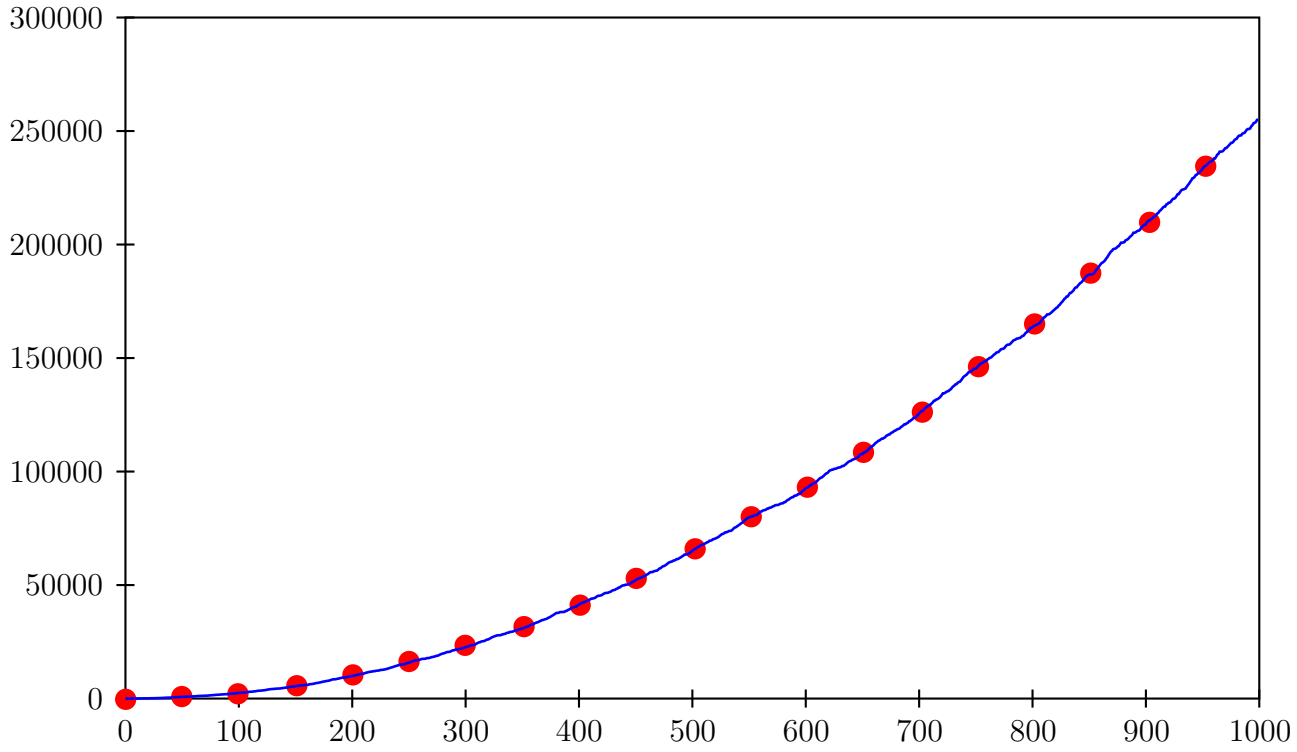
```
\readdata[nStep=10]{\data}{/home/voss/data/data1.dat}
```

The use nStep and xStep options make only real sense when also using the option plotstyle=dots. Otherwise the coordinates are connected by a line as usual. Also the xStep option needs increasing x values. Pay attention that nStep can be used for \readdata and for \listplot. If used in both macros than the effect is multiplied, e.g. \readdata with nStep=5 and \listplot with nStep=10 means, that only every 50^{th} data records is read and plotted.

When both, x/yStart/End are defined then the values are also compared with both values.

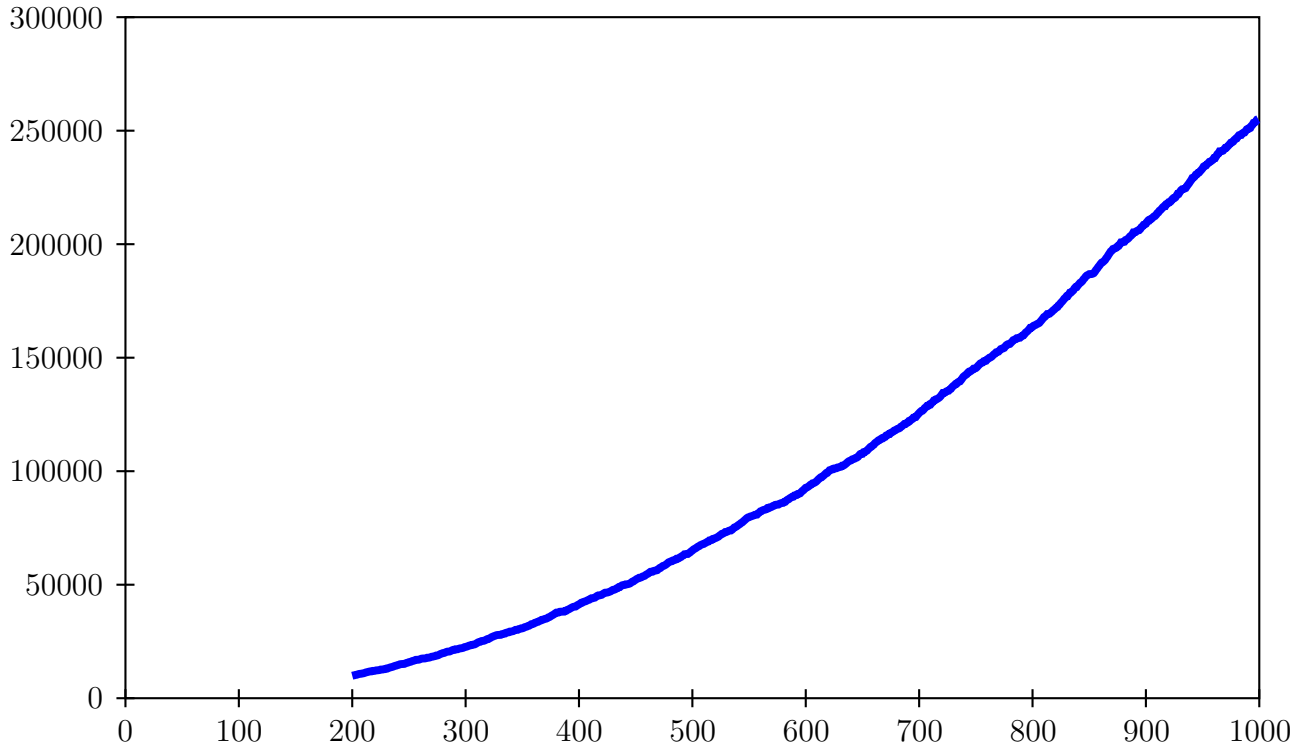
21.1 Example for nStep/xStep

The datafile data.dat contains 1000 data records. The thin blue line is the plot of all records with the plotstyle option curve.



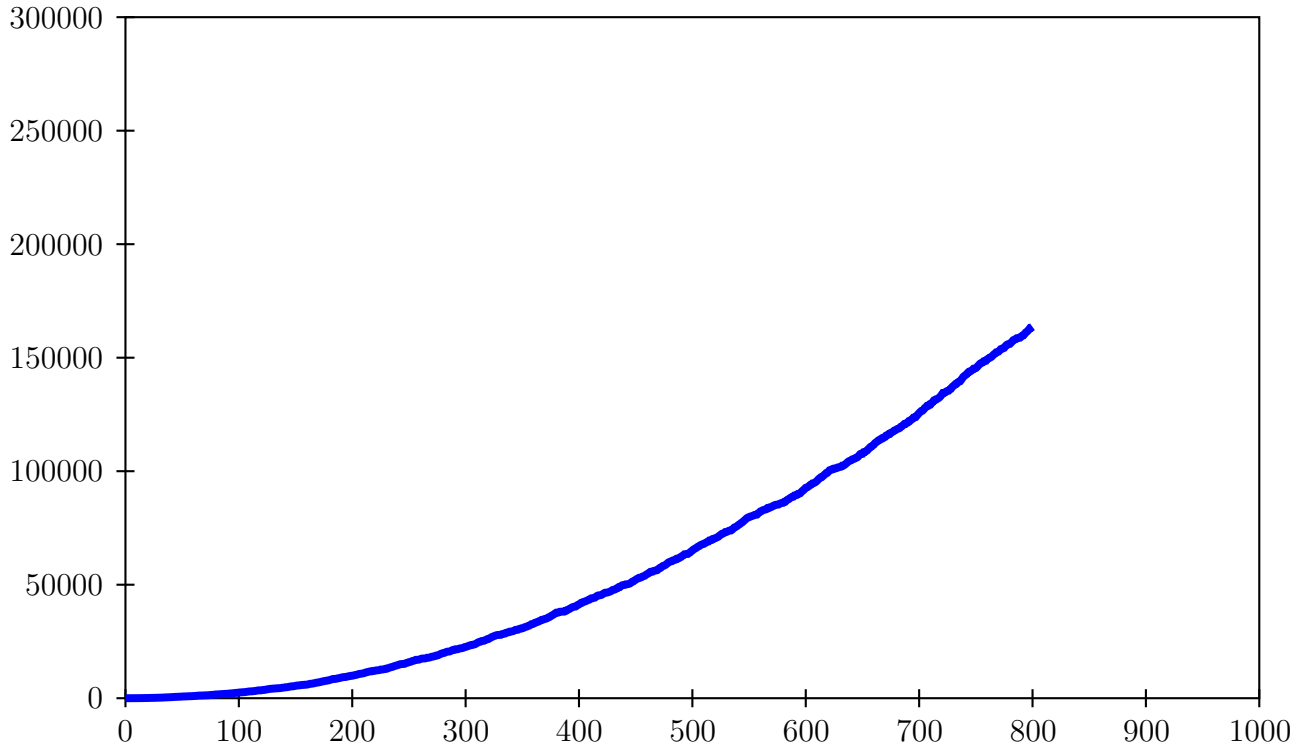

```
\readdata{\data}{examples/data.dat}
\psset{xunit=0.15mm,yunit=0.0003mm}
\begin{pspicture}(-80,-30000)(1000,300000)
\psaxes[axesstyle=frame,Dx=100,dx=100,%
    Dy=50000,dy=50000](1000,300000)
\listplot[nStep=50,%
    linewidth=3pt,%
    linecolor=red,%
    plotstyle=dots]{\data}
\listplot[linewidth=1pt,%
    linecolor=blue]{\data}
\end{pspicture}
```

21.2 Example for nStart/xStart



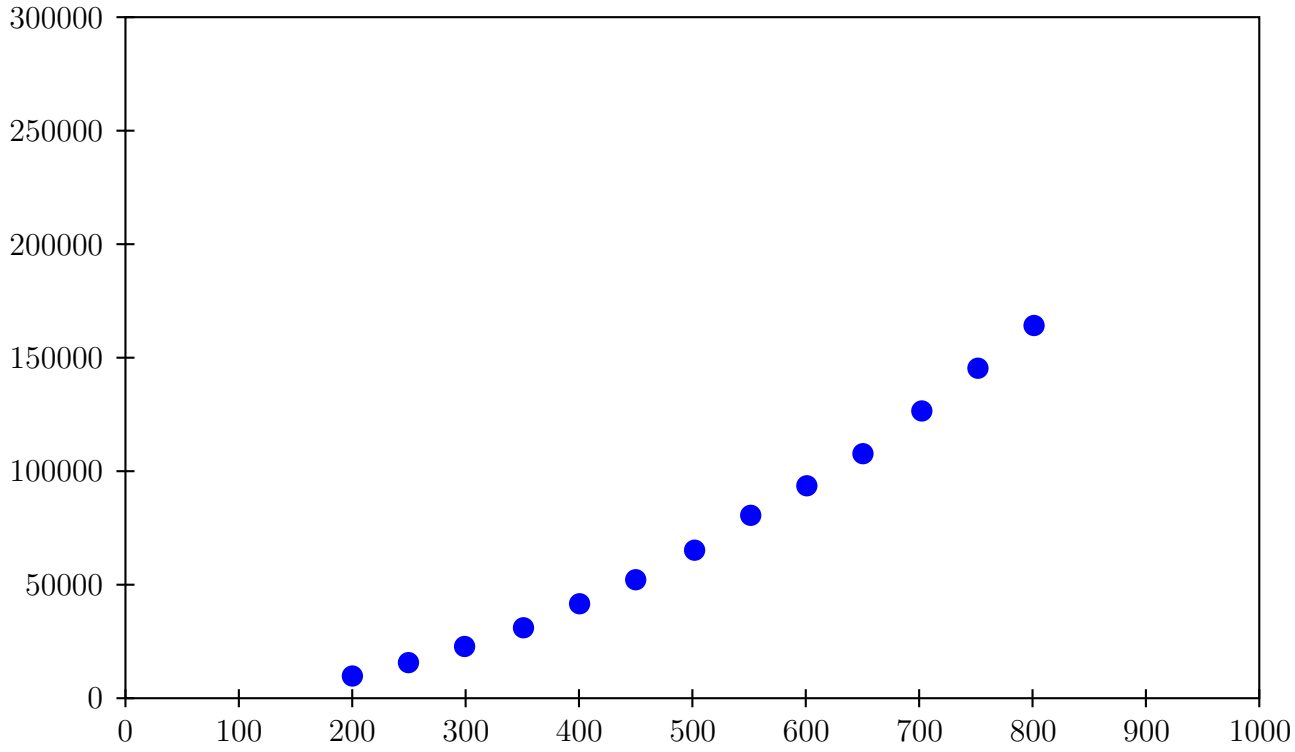

```
\readdata{\data}{examples/data.dat}
\psset{xunit=0.15mm,yunit=0.0003mm}
\begin{pspicture}(-80,-30000)(1000,310000)
\psaxes[axesstyle=frame,Dx=100,dx=100,%
    Dy=50000,dy=50000](1000,300000)
\listplot[nStart=200,%
 linewidth=3pt,%
    linecolor=blue]{\data}
\end{pspicture}
```

21.3 Example for nEnd/xEnd




```
\readdata{\data}{examples/data.dat}
\psset{xunit=0.15mm,yunit=0.0003mm}
\begin{pspicture}(-80,-30000)(1000,310000)
\psaxes[axesstyle=frame,Dx=100,dx=100,%
    Dy=50000,dy=50000](1000,300000)
\listplot[nEnd=800,%
 linewidth=3pt,%
    linecolor=blue]{\data}
\end{pspicture}
```

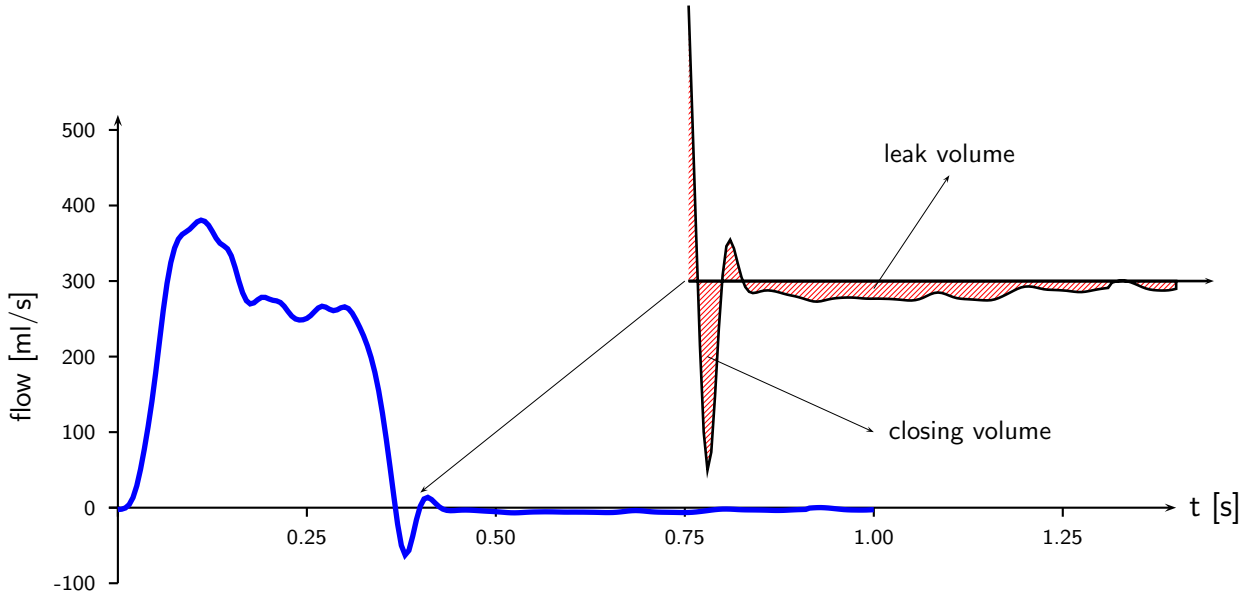
21.4 Example for all new options




```
\readdata{\data}{examples/data.dat}
\psset{xunit=0.15mm,yunit=0.0003mm}
\begin{pspicture}(-80,-30000)(1000,310000)
\psaxes[axesstyle=frame,Dx=100,dx=100,%
    Dy=50000,dy=50000](1000,300000)
\listplot[nStart=200, nEnd=800, nStep=50,%
 linewidth=3pt,%
    linecolor=blue,%
 plotstyle=dots]{\data}
\end{pspicture}
```

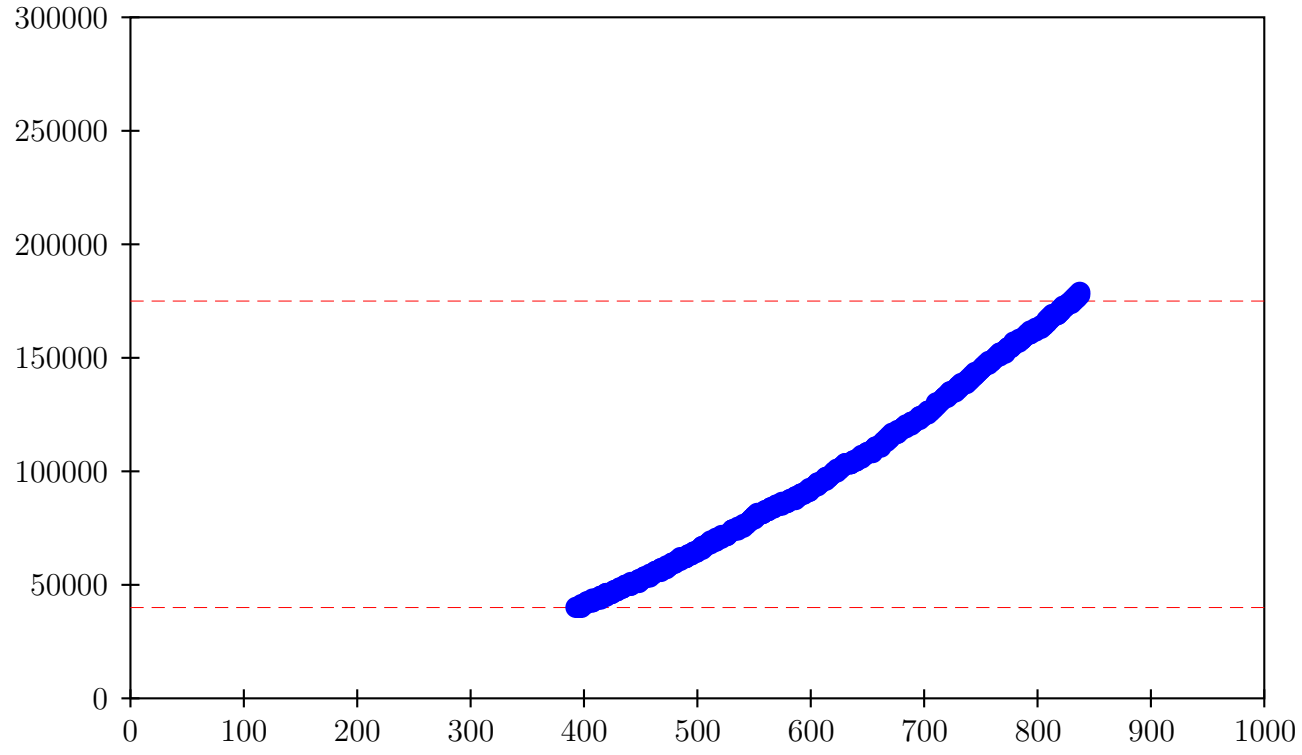
21.5 Example for xStart

This example shows the use of the same plot with different units and different `xStart` value. The blue curve is the original plot of the data records. To show the important part of the curve there is another one plotted with a greater `yunit` and a start value of `xStart=0.35`. This makes it possible to have a kind of a zoom to the original graphic.




```
\psset{xunit=10cm, yunit=0.01cm,%
 xLabel={\scriptsize\sffamily}, yLabel={\scriptsize\sffamily}%
}
\readdata{\data}{examples/data3.dat}
\begin{pspicture}(-0.1,-100)(1.5,700.0)
 \psaxes[Dx=0.25,Dy=100,tickstyle=bottom]{->}(0,0)(0,-100)(1.4,520)
 \uput[0](1.4,0){\textsf{t [s]}}
 \rput(-0.125,200){\rotateleft{\small\sffamily flow [ml/s]}}
 \listplot[linewidth=2pt, linecolor=blue]{\data}
 \rput(0.4,300){
  \pscustom[yunit=0.04cm, linewidth=1pt]{%
   \psline(1,-2.57)(1,0)(0.355,0)
   \listplot[xStart=0.355]{\data}
   \fill[fillstyle=hlines,fillcolor=gray,
    hatchwidth=0.4pt,hatchsep=1.5pt, hatchcolor=red]%
   \psline[linewidth=0.5pt]{->}(0.7,0)(1.05,0)
  }%
 }
 \psline[linewidth=.01]{->}(0.75,300)(0.4,20)
 \psline[linewidth=.01]{->}(1,290)(1.1,440)
 \rput(1.1,470){\footnotesize\sffamily leak volume}
 \psline[linewidth=.01]{->}(0.78,200)(1,100)
 \rput[l](1.02,100){\footnotesize\sffamily closing volume}
\end{pspicture}
```

21.6 Example for yStart/yEnd




```
\readdata{\data}{examples/data.dat}
\psset{xunit=0.15mm,yunit=0.0003mm}
\begin{pspicture}(-80,-30000)(1000,310000)
\psaxes[axesstyle=frame,Dx=100,dx=100,%
    Dy=50000,dy=50000](1000,300000)
\psset{linewidth=0.1pt, linestyle=dashed,linecolor=red}
\psline(0,40000)(1000,40000)
\psline(0,175000)(1000,175000)
\listplot[yStart=40000, yEnd=175000,%
 linewidth=3pt,%
    linecolor=blue,%
 plotstyle=dots]{\data}
\end{pspicture}
```

21.7 Example for plotNo/plotNoMax

By default the plot macros expect `x|y` data records, but when having data files with multiple values for y, like

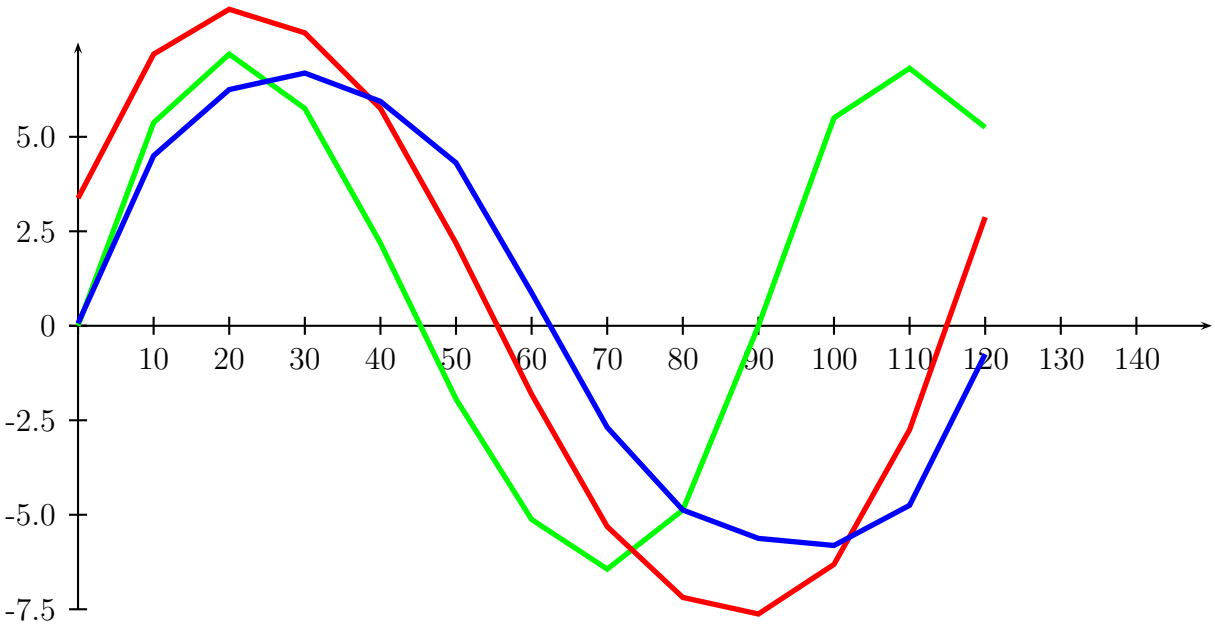
```
x y1 y2 y3 y4 ... yMax
x y1 y2 y3 y4 ... yMax
...
```

you can select the y value which should be plotted. The option `plotNo` marks the plotted value (default 1) and the option `plotNoMax` tells `pst-plot` how many y values are present. There are no real restrictions in the maximum number for `plotNoMax`.

We have the following data file:

```
[% file examples/data.dat
0    0    3.375    0.0625
10    5.375    7.1875    4.5
20    7.1875    8.375    6.25
30    5.75    7.75    6.6875
40    2.1875    5.75    5.9375
50    -1.9375    2.1875    4.3125
60    -5.125    -1.8125    0.875
70    -6.4375    -5.3125    -2.6875
80    -4.875    -7.1875    -4.875
90    0    -7.625    -5.625
100    5.5    -6.3125    -5.8125
110    6.8125    -2.75    -4.75
120    5.25    2.875    -0.75
]%
```

which holds data records for multiple plots (`x y1 y2 y3`). This can be plotted without any modification to the data file:



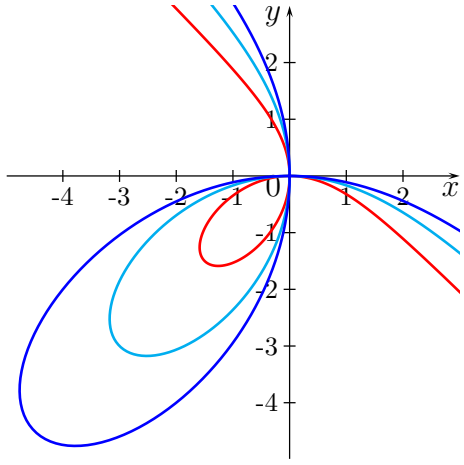

```
\readdata\Data{examples/dataMul.dat}

\psset{xunit=0.1cm, yunit=0.5cm}
\begin{pspicture}(0,-7.5)(150,10)
\psaxes[Dx=10,Dy=2.5]{->}(0,0)(0,-7.5)(150,7.5)
\psset{linewidth=2pt,plotstyle=line}
\listplot[linecolor=green,plotNo=1,plotNoMax=3]{\Data}
\listplot[linecolor=red,plotNo=2,plotNoMax=3]{\Data}
\listplot[linecolor=blue,plotNo=3,plotNoMax=3]{\Data}
\end{pspicture}
```

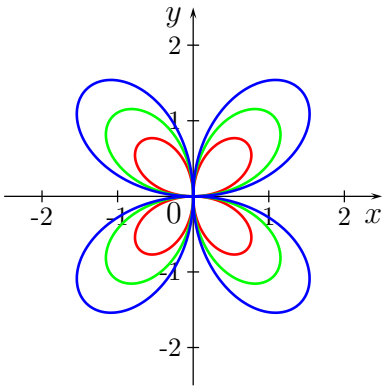
22 Polar plots

With the option `polarplot=false|true` it is possible to use `\psplot` in polar mode:

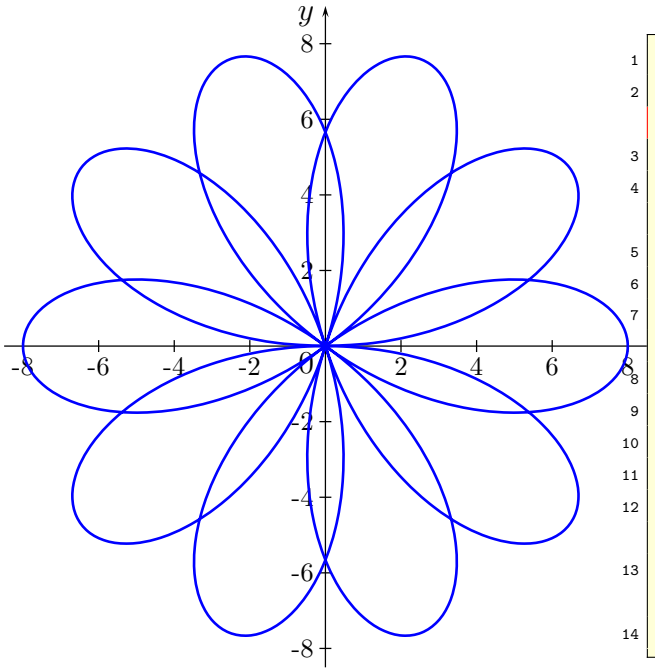
`\psplot[polarplot=true,...]{<start angle>}{<end angle>}{x y}`




```
\psset{plotpoints=200,unit=0.75}
\begin{pspicture}*(-5,-5)(3,3)% Ulrich Dirr
  \psaxes[labelsep=.75mm,xyLabel=\footnotesize,
    arrowlength=1.75,ticksize=2pt,%
    linewidth=0.17mm]{->}(0,0)(-4.99,-4.99)
      (3,3)
  \rput[Br](3,-.35){$x$}
  \rput[tr](-.15,3){$y$}
  \rput[Br](-.15,-.35){$0$}
  %
  \psset{linewidth=.35mm,polarplot=true}
  \psplot[linecolor=red]{140}{310}{3 neg x sin
    mul x cos mul x sin 3 exp x cos 3 exp add
    div}
  \psplot[linecolor=cyan]{140}{310}{6 neg x sin
     mul x cos mul x sin 3 exp x cos 3 exp add
    div}
  \psplot[linecolor=blue]{140}{310}{9 neg x sin
     mul x cos mul x sin 3 exp x cos 3 exp add
    div}
\end{pspicture}
```




```
\psset{plotpoints=200,unit=1}
\begin{pspicture}(-2.5,-2.5)(2.5,2.5)% Ulrich Dirr
 \psaxes[labelsep=.75mm,xyLabel=\footnotesize,%
 arrowlength=1.75,%
 ticksize=2pt,%
 linewidth=0.17mm]{->}(0,0)(-2.5,-2.5)(2.5,2.5)
  \rput[Br](2.5,-.35){$x$}
  \rput[tr](-.15,2.5){$y$}
  \rput[Br](-.15,-.35){$0$}
  %
  \psset{linewidth=.35mm,plotstyle=curve,polarplot=
    true}
  \psplot[linecolor=red]{0}{360}{x cos 2 mul x sin
    mul}
  \psplot[linecolor=green]{0}{360}{x cos 3 mul x
    sin mul}
  \psplot[linecolor=blue]{0}{360}{x cos 4 mul x sin
     mul}
\end{pspicture}
```




```
\psset{plotpoints=200,unit=0.5}
\begin{pspicture}(-8.5,-8.5)(9,9)%
  Ulrich Dirr
\psaxes[Dx=2,dx=2,Dy=2,dy=2,%
 labelsep=.75mm,xyLabel=\footnotesize,
   %
 arrowlength=1.75,%
 ticksize=2pt,%
 linewidth=0.17mm]{->}(0,0)(-8.5,-8.5)
   (9,9)
\rput[Br](9,-.7){$x$}
\rput[tr](-.3,9){$y$}
\rput[Br](-.3,-.7){$0$}
%
\psset{linewidth=.35mm,plotstyle=curve
  ,polarplot=true}
\psplot[linecolor=blue]{0}{720}{8 2.5
  x mul sin mul}
\end{pspicture}
```

23 Change log

Look at the end of the package file for the change log.

24 Credits

Denis Girou | Christophe Jorssen | Manuel Luque | Jens-Uwe Morawski | Timothy Van Zandt

References

[1] Denis Girou. Présentation de PSTricks. *Cahier GUTenberg*, 16:21–70, April 1994.

[2] Michel Goosens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Graphics Companion*. Addison-Wesley Publishing Company, Reading, Mass., 1997.

[3] Laura E. Jackson and Herbert Voß. Die plot-funktionen von `pst-plot`. *Die TeXnische Komödie*, 2/02:27–34, June 2002.

[4] Nikolai G. Kollock. *PostScript richtig eingesetzt: vom Konzept zum praktischen Einsatz*. IWT, Vaterstetten, 1989.

[5] Herbert Voß. *Chaos und Fraktale selbst programmieren: von Mandelbrotmengen über Farbmanipulationen zur perfekten Darstellung*. Franzis Verlag, Poing, 1994.

[6] Herbert Voß. Die mathematischen Funktionen von PostScript. *Die TeXnische Komödie*, 1/02, March 2002.

[7] Timothy van Zandt. *PSTricks - PostScript macros for generic TEX*. http://www.tug.org/application/PSTricks, 1993.

[8] Timothy van Zandt. ***multido.tex*** *- a loop macro, that supports fixed-point addition*. CTAN:/graphics/pstricks/generic/multido.tex, 1997.

[9] Timothy van Zandt. ***pst-plot****: Plotting two dimensional functions and data*. CTAN:graphics/pstricks/generic/pst-plot.tex, 1999.

[10] Timothy van Zandt and Denis Girou. Inside PSTricks. *TUGboat*, 15:239–246, September 1994.