`pstricks-add`
additionals Macros for `pstricks`

v.2.72

Herbert Voß

November 26, 2005

Abstract

This version of `pstricks-add` needs `pstricks.tex` version >1.04 from June 2004, otherwise the additional macros may not work as espected. The ellipsis material 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-add` will for ever be an experimental and dynamical package, try it at your own risk.

- It is important to load `pstricks-add` as **last** PSTricks related package, otherwise a lot of the macros won't work in the expected way.
- `pstricks-add` uses the extended version of the keyval package. So be sure, that you have installed `pst-xkey` which is part of the `xkeyval`-package and that all packages, that uses the old keyval interface are loaded **before** the `xkeyval`.[1]
- the option `tickstyle` from `pst-plot`is no more supported, use `ticksize` instead.
- the option `xyLabel` is no more supported, use the macros `\def\pshlabel#1{...}` and `\def\psvlabel#1{...}` instead.

Contents

Part I

pstricks

1 Numeric functions

All macronames contain a `@` in their name, because they are only for internal use, but it is no problem to use it as the other macros. One can define another name without a `@`:

```
\makeatletter
\let\pstdivide\pst@divide
\makeatother
```

or put the macro inside of the `\makeatletter` – `\makeatother` sequence.

1.1 `\pst@divide`

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

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

```
5.66666
-0.17647
```

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

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

1.2 `\pst@mod`

`pstricks-add` defines an additional numeric function for the modulus:

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

```
4
1
```

```
\makeatletter
\pst@mod{34}{6}\modulo \modulo\\
\pst@mod{25}{-6}\modulo \modulo
\makeatother
```

this gives the output 4. Using this internal numeric functions in documents requires a setting inside the `makeatletter` and `makeatother` environment. It makes some sense to define a new macroname in the preamble to use it throughou, e.g. `\let\modulo\pst@mod`.

1.3 `\pst@max`

`\pst@max{<integer>}{<integer>}{<result as count register>}`

```
-6
11
```

```
\newcount\maxNo
\makeatletter
\pst@max{-34}{-6}\maxNo \the\maxNo\\
\pst@max{0}{11}\maxNo \the\maxNo
\makeatother
```

1.4 `\pst@maxdim`

`\pst@maxdim{<dimension>}{<dimension>}{<result as dimension register>}`

```
1234.0pt
967.39369pt
```

```
\newdimen\maxDim
\makeatletter
\pst@maxdim{34cm}{1234pt}\maxDim \the\maxDim\\
\pst@maxdim{34cm}{123pt}\maxDim \the\maxDim
\makeatother
```

1.5 `\pst@abs`

`\pst@abs{<integer>}{<result as a count register>}`

```
34
4
```

```
\newcount\absNo
\makeatletter
\pst@abs{-34}\absNo \the\absNo\\
\pst@abs{4}\absNo \the\absNo
\makeatother
```

1.6 \pst@absdim

```
\pst@absdim{<dimension>}{<result as a dimension register>}
```

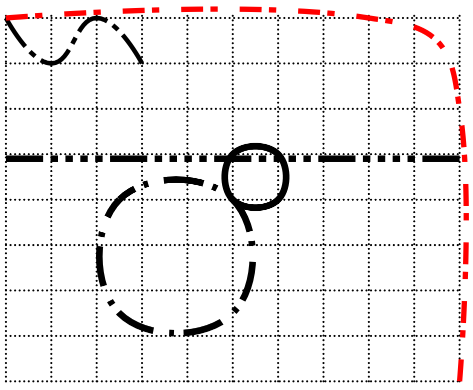
967.39369pt
0.00006pt

```
\newdimen\absDim
\makeatletter
\pst@absdim{-34cm}\absDim \the\absDim\\
\pst@absdim{4sp}\absDim \the\absDim
\makeatother
```

2 Dashed Lines

Tobias Nï¿½ring implemented an enhanced feature for dashed lines. The number of arguments is no more limited.

```
dash=value1[unit] value2[unit] ...
```

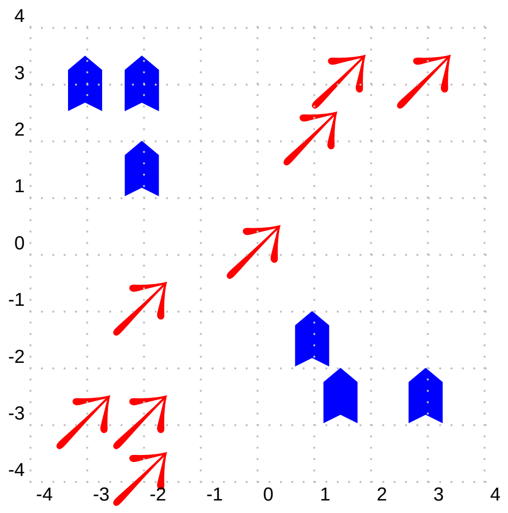


```
\psset{linewidth=2.5pt,unit=0.6}
\begin{pspicture}(-5,-4)(5,4)
 \psgrid[subgriddiv=0,griddots=10,gridlabels=0pt]
  \psset{linestyle=dashed}
  \pscurve[dash=5mm 1mm 1mm 1mm,linewidth
    =0.1](-5,4)(-4,3)(-3,4)(-2,3)
  \psline[dash=5mm 1mm 1mm 1mm 1mm 1mm 1mm 1mm 1
    mm 1mm](-5,0.9)(5,0.9)
  \psccurve[linestyle=solid](0,0)(1,0)(1,1)(0,1)
  \psccurve[linestyle=dashed,dash=5mm 2mm 0.1
    0.2,linetype=0](0,0)(-2.5,0)(-2.5,-2.5)
    (0,-2.5)
  \pscurve[dash=3mm 3mm 1mm 1mm,linecolor=red,
    linewidth=2pt](5,-4)(5,2)(4.5,3.5)(3,4)(-5,4)
\end{pspicture}
```

3 \rmultiput: a multiple \rput

`PSTricks` already knows a `multirput`, which puts a box n times with a difference of *dx* and *dy* relativ to each other. It is not possible to put it with a different distance from one point to the next one. This is possible with `rmultiput`:

```
\rmultiput[<options>]{<any material>}(x1,y1)(x2,y2) ... (xn,yn)
\rmultiput*[<options>]{<any material>}(x1,y1)(x2,y2) ... (xn,yn)
```

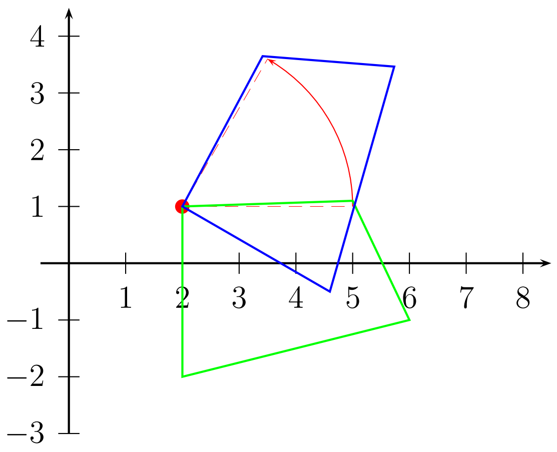


```
\psset{unit=0.75}
\begin{pspicture}(-4,-4)(4,4)
\rmultiput[rot=45]{\red\psscalebox{3}{\ding{250}}}%
    (-2,-4)(-2,-3)(-3,-3)(-2,-1)(0,0)(1,2)(1.5,3)
      (3,3)
\rmultiput[rot=90,ref=lC]{\blue\psscalebox{2}{\ding
  {253}}}%
    (-2,2.5)(-2,2.5)(-3,2.5)(-2,1)(1,-2)(1.5,-3)
      (3,-3)
\psgrid[subgriddiv=0,gridcolor=lightgray]
\end{pspicture}
```

4 \psrotate: Rotating objects

`\rput` also has an optional argument for rotating objects, but always depending to the `\rput` coordinates. With `\psrotate` the rotating center can be placed anywhere. The rotation is done with `\pscustom`, all optional arguments are only valid if they are part of the `\pscustom` macro.

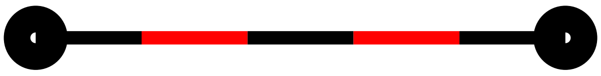
```
\psrotate[options](x,y){rot angle}{<object>}
```



```
\psset{unit=0.75}
\begin{pspicture}(-0.5,-3.5)(8.5,4.5)
  \psaxes{->}(0,0)(-0.5,-3)(8.5,4.5)
  \psdots[linecolor=red,dotscale=1.5](2,1)
  \psarc[linecolor=red,linewidth=0.4pt,showpoints
    =true]
        {->}(2,1){3}{0}{60}
  \pspolygon[linecolor=green](2,1)(5,1.1)(6,-1)
    (2,-2)
  \psrotate[linecolor=blue](2,1){60}{
    \pspolygon(2,1)(5,1.1)(6,-1)(2,-2)}
\end{pspicture}
```

5 \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`.



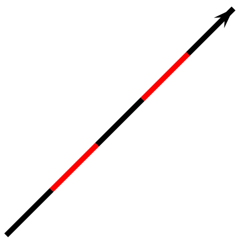
```
\begin{pspicture}(0,-0.5)(7,0.5)
\pslineII[linewidth=5pt,arrowscale=2]{o-o
  }(0,0)(7,0)
\end{pspicture}
```

5.1 The options

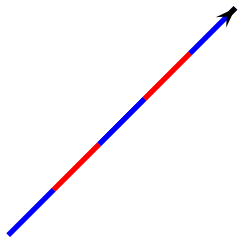
name	meaning
`dashColorI`	first color, default is `black`
`dashColorII`	second color, default is `red`
`dashNo`	the difference in per cent of the colored lines, default is 0.2
`linecap`	how two lines are connected. 0: no modification 1: rounded edges 2: an additional half square at both ends

`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!

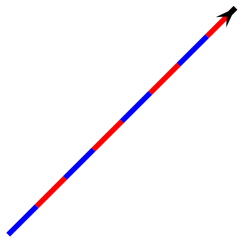
5.2 Examples



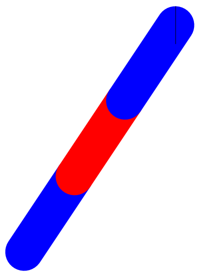
```
\psset{linewidth=2pt}
\begin{pspicture}(3,3)
  \pslineII{->}(0,0)(3,3)
\end{pspicture}
```



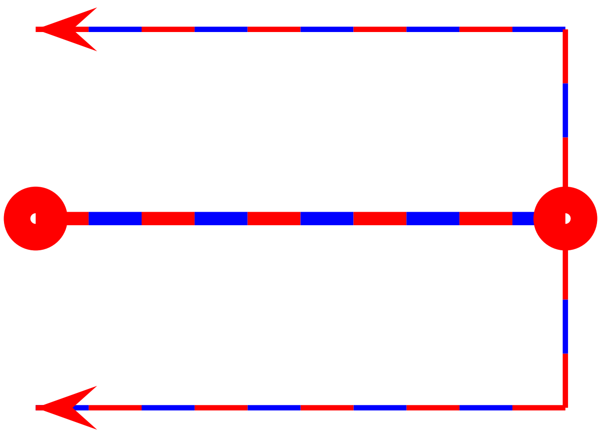
```
\psset{linewidth=2pt}
\begin{pspicture}(3,3)
  \pslineII[dashColorI=blue]{->}(0,0)(3,3)
\end{pspicture}
```



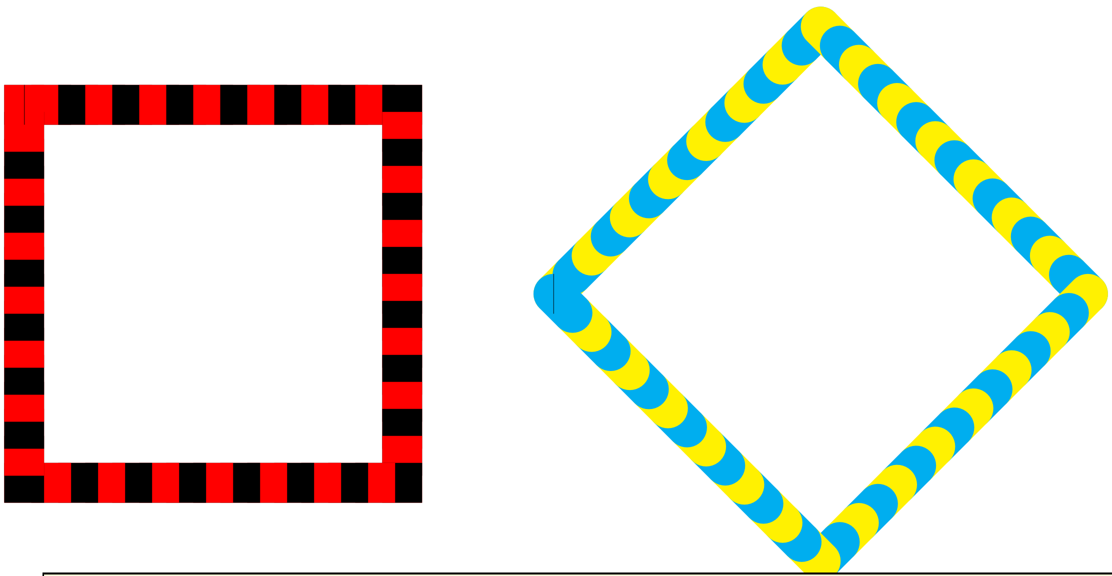
```
\psset{linewidth=2pt}
\begin{pspicture}(3,3)
  \pslineII[dashColorI=blue,dashNo=15]{->}(0,0)(3,3)
\end{pspicture}
```



```
\psset{linewidth=2pt}
\begin{pspicture}(3,3)
  \pslineII[dashColorI=blue,linecap=1,%
      dashNo=0.3,linewidth=0.5](0,0)(2,3)
\end{pspicture}
```



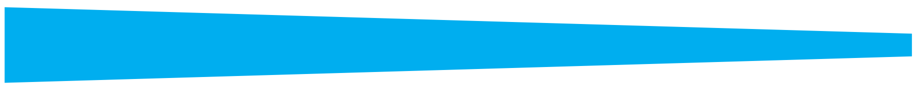
```
\psset{linecolor=red,arrowscale=3}
\psset{dashColorI=red,dashColorII=blue,
  dashNo=20,linewidth=2pt}
\begin{pspicture}(0,0)(7,-5)
\pslineII{<->}(0,0)(7,0)(7,-5)(0,-5)
\pslineII[linewidth=5pt,%
    dashNo=0.1,arrowscale=2]{o-o
      }(0,-2.5)(7,-2.5)
\end{pspicture}
```



```
\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}
```

6 \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`.

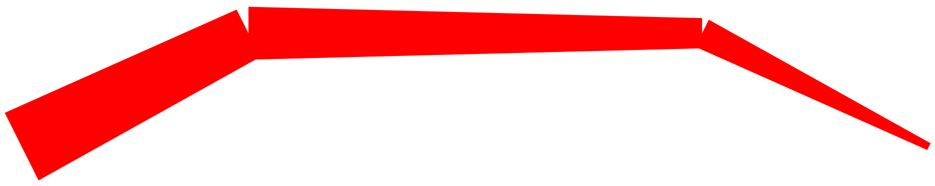


```
\pslineIII[wBegin=1cm,wEnd=0.3cm,linecolor=cyan](0,0)(12,0)
```

6.1 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

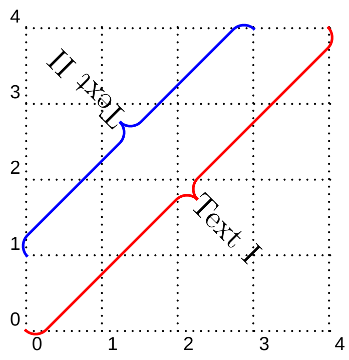


```
\pslineIII[wBegin=1cm,wEnd=0.1cm,linecolor=cyan](0,0)(0,1.5)(12,1.5)
  (12,0)
```

7 `\psbrace`

7.1 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), $(nodeA|nodeB)$, ...

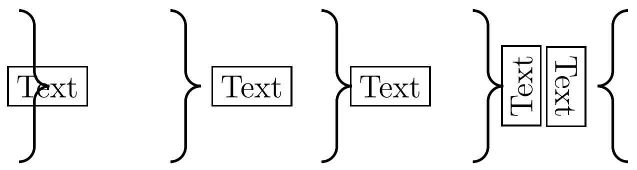
7.2 Options

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.

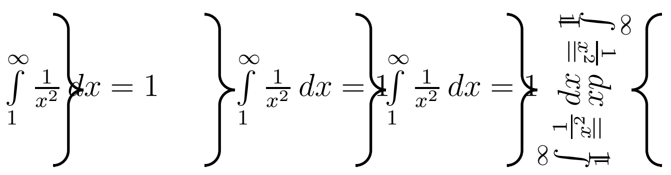
name	meaning
`braceWidth`	default is 0.35
`bracePos`	relative position (default is 0.5)
`linearc`	absolute value for the arcs (default is 2mm)
`nodesepA`	x-separation (default is $0pt$)
`nodesepB`	y-separation (default is $0pt$)
`rot`	additional rotating for the text (default is 0)
`ref`	reference point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

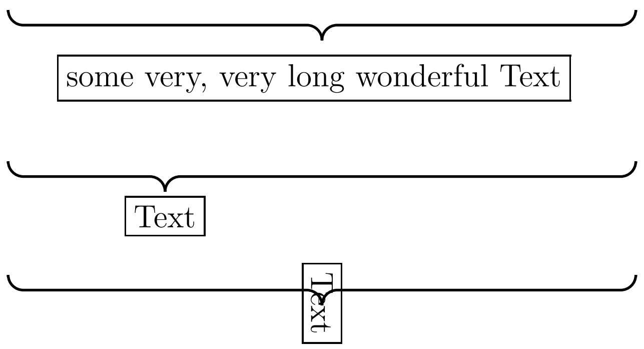
7.3 Examples



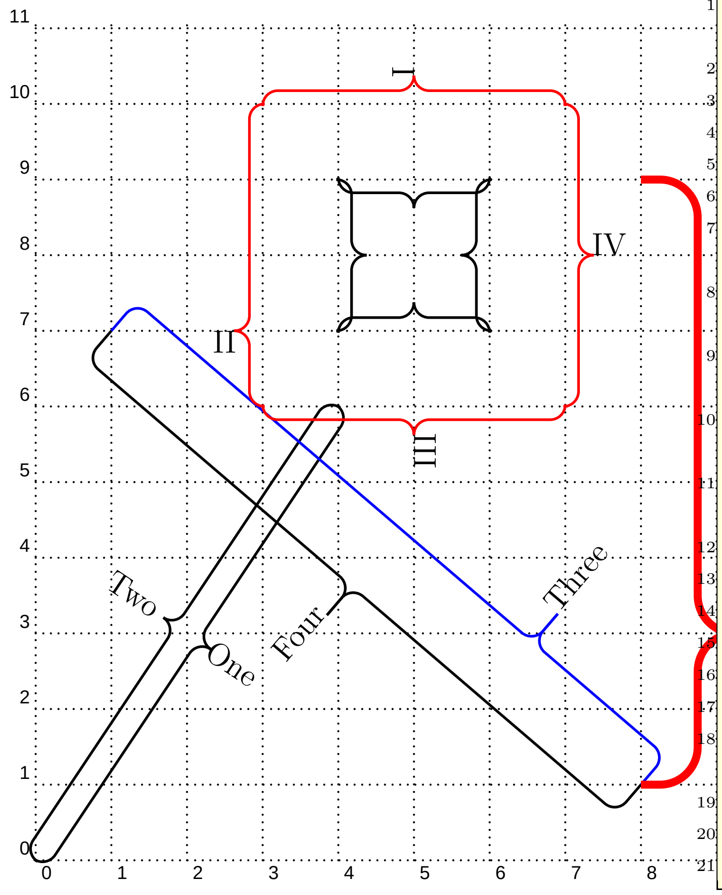
```
\begin{pspicture}(8,2.5)
\psbrace(0,0)(0,2){\fbox{Text}}%
\psbrace[nodesepA=20pt](2,0)(2,2){\fbox{
  Text}}
\psbrace[ref=lC](4,0)(4,2){\fbox{Text}}
\psbrace[ref=lt,rot=90,nodesepB=-15pt
  ](6,0)(6,2){\fbox{Text}}
\psbrace[ref=lt,rot=90,nodesepA=-5pt,
  nodesepB=15pt](8,2)(8,0){\fbox{Text}}
\end{pspicture}
```



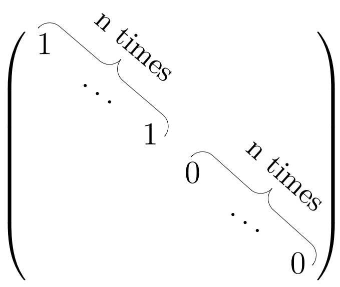
```
\def\someMath{$\int\limits_1^{\infty}\
  frac{1}{x^2}\,dx=1$}
\begin{pspicture}(8,2.5)
\psbrace(0,0)(0,2){\someMath}%
\psbrace[nodesepA=30pt](2,0)(2,2){\
  someMath}
\psbrace[ref=lC](4,0)(4,2){\someMath}
\psbrace[ref=lt,rot=90,nodesepB=-30pt
  ](6,0)(6,2){\someMath}
\psbrace[ref=lt,rot=90,nodesepB=30pt
  ](8,2)(8,0){\someMath}
\end{pspicture}
```



```
\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}
```



```
\def\someMath{$\int\limits_1^{\infty}\
  frac{1}{x^2}\,dx=1$}
\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}
```



```
\[
\begin{pmatrix}
    \Rnode[vref=2ex]{A}{~1} \\
    & \ddots \\
    && \Rnode[href=2]{B}{1} \\
    &&& \Rnode[vref=2ex]{C}{0} \\
    &&&& \ddots \\
    &&&&& \Rnode[href=2]{D}{0}~ \\
\end{pmatrix}
\]
\psbrace[linewidth=0.1pt,rot=-90,nodesep=0.2](B)(A){\small
   n times}
\psbrace[linewidth=0.1pt,rot=-90,nodesep=0.2](D)(C){\small
   n times}
```

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.

```
\begin{framed}
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}
\end{framed}
```

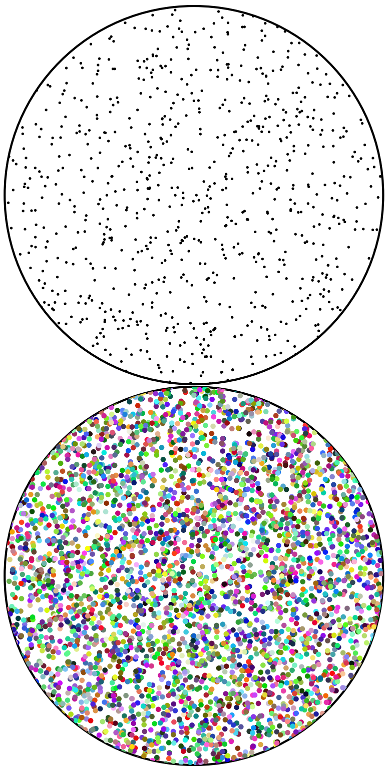
8 Random dots

The syntax of the new macro `\psRandom` is:

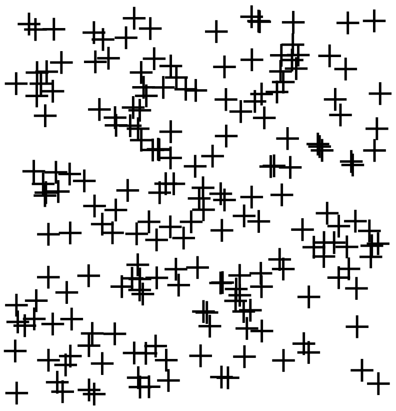
```
\psRandom[<option>]{}
\psRandom[<option>]{<clip path>}
\psRandom[<option>](<xMax,yMax>){<clip path>}
\psRandom[<option>](<xMin,yMin>)(<xMax,yMax>){<clip path>}
```

If there is no area for the dots defined, then `(0,0)(1,1)` in the actual scale is used for placing the dots. This area should be greater than the clipping path to be sure that the dots are placed over the full area. The clipping path can be everything. If no clipping path is given, then the frame `(0,0)(1,1)` in user coordinates is used. The new options are:

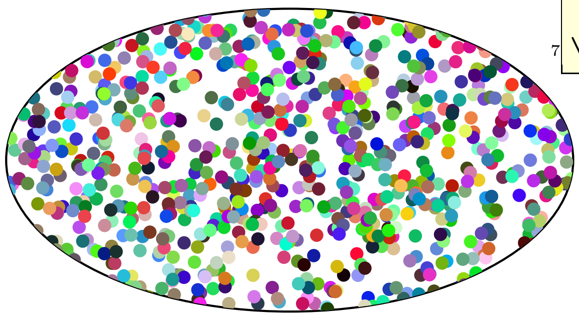
name	default	
`randomPoints`	`1000`	number of random dots
`color`	`false`	random color

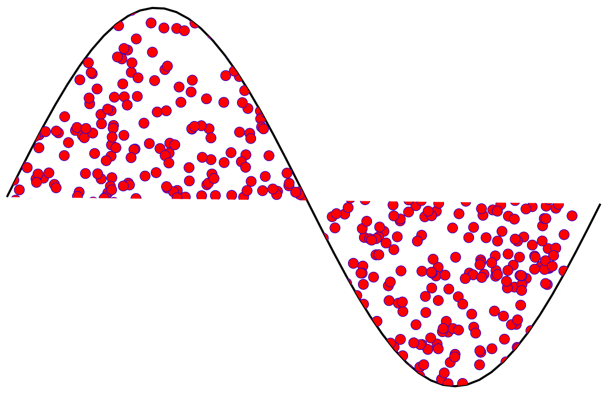


```
\psset{unit=5cm}
\begin{pspicture}(1,1)
  \psRandom[dotsize=1pt,fillstyle=solid](1,1){\pscircle
    (0.5,0.5){0.5}}
\end{pspicture}
\begin{pspicture}(1,1)
  \psRandom[dotsize=2pt,randomPoints=5000,color,%
      fillstyle=solid](1,1){\pscircle(0.5,0.5){0.5}}
\end{pspicture}
```



```
\psset{unit=5cm}
\begin{pspicture}(1,1)
  \psRandom[randomPoints=200,dotsize=8pt,dotstyle
    =+]{}
\end{pspicture}
\begin{pspicture}(1.5,1)
  \psRandom[dotsize=5pt,color](0,0)(1.5,0.8){\
    psellipse(0.75,0.4)(0.75,0.4)}
\end{pspicture}
```



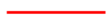
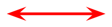
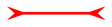
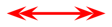
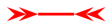
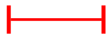
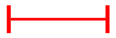
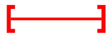
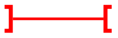
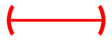
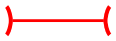
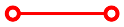
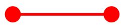
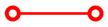
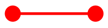
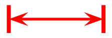
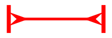
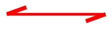
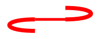


```
\psset{unit=2.5cm}
\begin{pspicture}(0,-1)(3,1)
  \psRandom[dotsize=4pt,dotstyle=o,
    linecolor=blue,fillcolor=red,%
     fillstyle=solid,randomPoints=1000]%
      (0,-1)(3,1){\psplot{0}{3.14}{ x 114
         mul sin }}
\end{pspicture}
```

9 Arrows

9.1 Definition

`pstricks-add` defines the following ”arrows“:

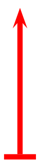
Value	Example	Name
`-`		None
`<->`		Arrowheads.
`>-<`		Reverse arrowheads.
`<<->>`		Double arrowheads.
`>>-<<`		Double reverse arrowheads.
`|-|`		T-bars, flush to endpoints.
`|*-|*`		T-bars, centered on endpoints.
`[-]`		Square brackets.
`]-[`		Reversed square brackets.
`(-)`		Rounded brackets.
`)-(`		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.
`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

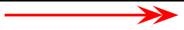
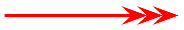
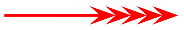
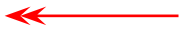
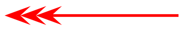
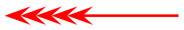
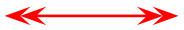
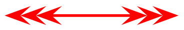
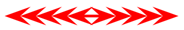
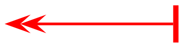
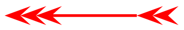
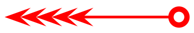
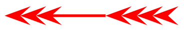
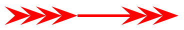
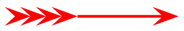
```
\psline[<general options>]{<arrow type>}(A)(B)
\psline[linecolor=red,linewidth=2pt]{|->}(0,0)(0,2)
```



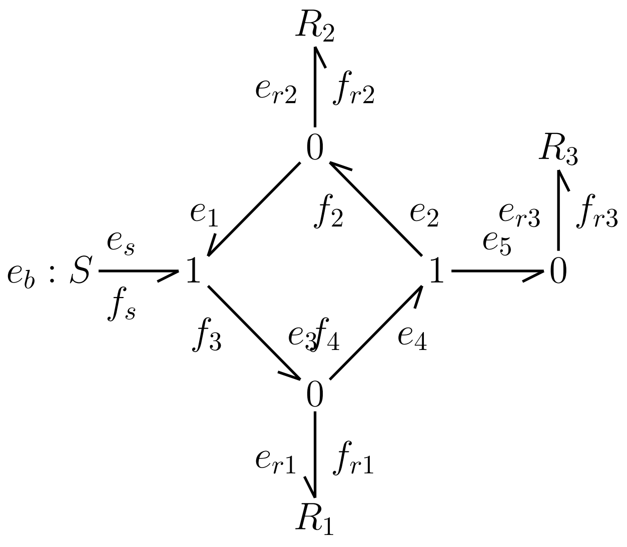
9.2 Multiple arrows

There are two new options which are only valid for the arrow type `<<` or `>>`. `nArrow` sets both, the `nArrowA` and the `nArrowB` parameter. The meaning is declared in the following tables. Without setting one of these parameters the behaviour is like the one described in the old PSTricks manual.

Value	Meaning
`->>`	-A
`<<->>`	A-A
`<<-`	A-
`>>-`	B-
`-<<`	-B
`>>-<<`	B-B
`>>->>`	B-A
`<<-<<`	A-B

Value	Example
`\psline{->>}(0,1ex)(2.3,1ex)`	
`\psline[nArrowsA=3]{->>}(0,1ex)(2.3,1ex)`	
`\psline[nArrowsA=5]{->>}(0,1ex)(2.3,1ex)`	
`\psline{<<-}(0,1ex)(2.3,1ex)`	
`\psline[nArrowsA=3]{<<-}(0,1ex)(2.3,1ex)`	
`\psline[nArrowsA=5]{<<-}(0,1ex)(2.3,1ex)`	
`\psline{<<->>}(0,1ex)(2.3,1ex)`	
`\psline[nArrowsA=3]{<<->>}(0,1ex)(2.3,1ex)`	
`\psline[nArrowsA=5]{<<->>}(0,1ex)(2.3,1ex)`	
`\psline{<<-|}(0,1ex)(2.3,1ex)`	
`\psline[nArrowsA=3]{<<-<<}(0,1ex)(2.3,1ex)`	
`\psline[nArrowsA=5]{<<-o}(0,1ex)(2.3,1ex)`	
`\psline[nArrowsA=3,nArrowsB=4]{<<-<<}(0,1ex)(2.3,1ex)`	
`\psline[nArrowsA=3,nArrowsB=4]{>>->>}(0,1ex)(2.3,1ex)`	
`\psline[nArrowsA=1,nArrowsB=4]{>>->>}(0,1ex)(2.3,1ex)`	

9.3 `hookarrow`



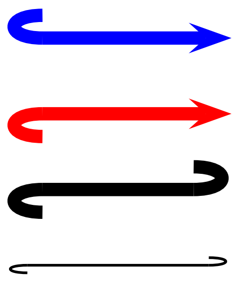
```
\psset{arrowsize=8pt,arrowlength=1,
  linewidth=1pt,nodesep=2pt,shortput=
  tablr}
\large
\begin{psmatrix}[colsep=12mm,rowsep=10mm
  ]
       &   & $R_2$           \\
       &   &  0   &   & $R_3$\\
$e_b:S$ & 1 &      & 1 & 0   \\
       &   &  0              \\
       &   &  $R_1$          \\
\end{psmatrix}
\ncline{h-}{1,3}{2,3}<{$e_{r2}$}>{$f_{r
  2}$}
\ncline{-h}{2,3}{3,2}<{$e_1$}
\ncline{-h}{3,1}{3,2}^{$e_s$}_{$f_{s}$}
\ncline{-h}{3,2}{4,3}>{$e_3$}<{$f_3$}
\ncline{-h}{4,3}{3,4}>{$e_4$}<{$f_4$}
\ncline{-h}{3,4}{2,3}>{$e_2$}<{$f_2$}
\ncline{-h}{3,4}{3,5}^{$e_5$}
\ncline{-h}{3,5}{2,5}<{$e_{r3}$}>{$f_{r
  3}$}
\ncline{-h}{4,3}{5,3}<{$e_{r1}$}>{$f_{r
  1}$}
```

9.4 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:



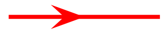
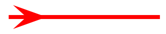
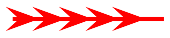
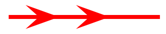
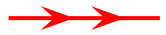
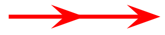
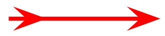
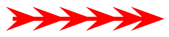
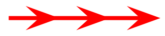
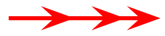
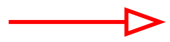
```
\begin{pspicture}(3,4)
\psline[linewidth=5pt,linecolor=blue,hooklength=5mm,hookwidth=-3mm]{H
  ->}(0,3.5)(3,3.5)
\psline[linewidth=5pt,linecolor=red,hooklength=5mm,hookwidth=3mm]{H
  ->}(0,2.5)(3,2.5)
\psline[linewidth=5pt,hooklength=5mm,hookwidth=3mm]{H-H}(0,1.5)
  (3,1.5)
\psline[linewidth=1pt]{H-H}(0,0.5)(3,0.5)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

$$\begin{tikzcd}
E \arrow[r, hook] & W_i(X) \arrow[rr, "\tilde{t}"] \arrow[dr, "W_{ij}"'] & & Y \\
 & & W_j(X) \arrow[ur, "\tilde{s}"'] &
\end{tikzcd}$$

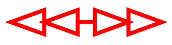
```latex
$\begin{psmatrix}
E&W_i(X)&&Y\\
&&W_j(X)
\psset{arrows=->,nodesep=3pt,linewidth=2pt}
\everypsbox{\scriptstyle}
\ncline[linecolor=red,arrows=H->,%
  hooklength=4mm,hookwidth=2mm]{1,1}{1,2}
\ncline{1,2}{1,4}^{\tilde{t}}
\ncline{1,2}{2,3}<{W_{ij}}
\ncline{2,3}{1,4}>{\tilde{s}}
\end{psmatrix}$
```

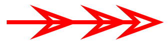
9.5 `ArrowInside` Option

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)`	

Name	Example	Output

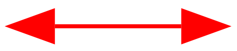
`ArrowFill`	`\psline[ArrowFill=false,%`
	`   arrowinset=0]{<<->>}(0,0)(2,0)`
		

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

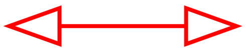
Without the default arrow definition there is only the one inside the line, defined by the type and the position. The position is relative to the length of the whole line. 0.25 means at 25% of the line length. The peak of the arrow gets the coordinates which are calculated by the macro. If you want arrows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

9.6 `ArrowFill` Option

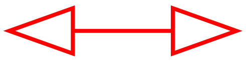
By default all arrows are filled polygons. With the option `ArrowFill=false` there are "white" arrows. Only for the beginning/end arrows they are empty, the inside arrows are overpainted with the line.



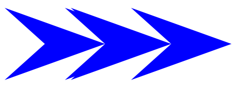
```
\psset{arrowscale=3}
\psline[linecolor=red,arrowinset=0]{<->}(0,0)(3,0)
```



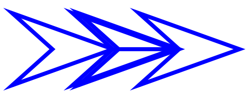
```
\psset{arrowscale=3}
\psline[linecolor=red,arrowinset=0,ArrowFill=false]{<->}(0,0)(3,0)
```



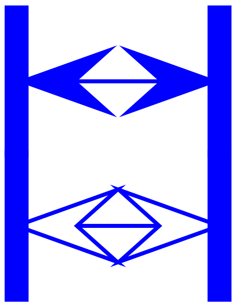
```
\psset{arrowscale=3}
\psline[linecolor=red,arrowinset=0,arrowsize=0.2,ArrowFill=false
  ]{<->}(0,0)(3,0)
```



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



```
\psset{arrowscale=3}
\psline[linecolor=blue,arrowscale=6,ArrowFill=false]{>>->>}(0,0)
  (3,0)
\rule{3cm}{0pt}\\[30pt]
```



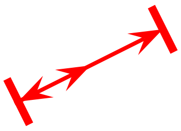
```
\psset{arrowscale=3}
\psline[linecolor=blue,arrowscale=6,ArrowFill=true]{>|->|}(0,0)
  (3,0)
```

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

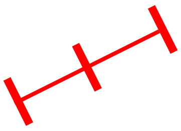
9.7 Examples

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

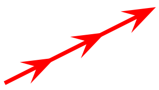
9.7.1 \psline



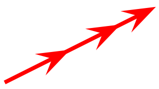
```
\begin{pspicture}(2,2)
\psset{arrowscale=2,ArrowFill=true}
\psline[ArrowInside=->]{|<->|}(2,1)
\end{pspicture}
```



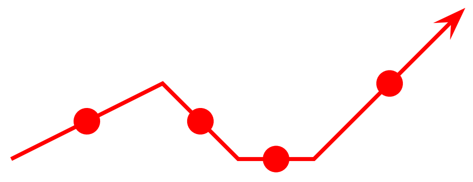
```
\begin{pspicture}(2,2)
\psset{arrowscale=2,ArrowFill=true}
\psline[ArrowInside=-|]{|-|}(2,1)
\end{pspicture}
```



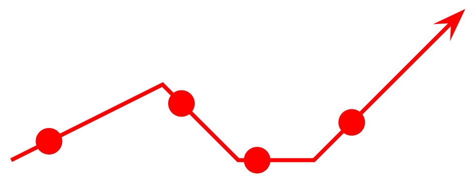
```
\begin{pspicture}(2,2)
\psset{arrowscale=2,ArrowFill=true}
\psline[ArrowInside=->,ArrowInsideNo=2]{->}(2,1)
\end{pspicture}
```



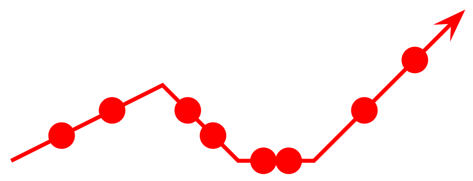
```
\begin{pspicture}(2,2)
\psset{arrowscale=2,ArrowFill=true}
\psline[ArrowInside=->,ArrowInsideNo=2,ArrowInsideOffset=0.1]{->}(2,1)
\end{pspicture}
```



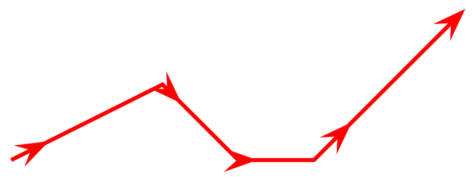
```
\begin{pspicture}(6,2)
\psset{arrowscale=2,ArrowFill=true}
\psline[ArrowInside=-*]{->}(0,0)(2,1)(3,0)(4,0)
  (6,2)
\end{pspicture}
```



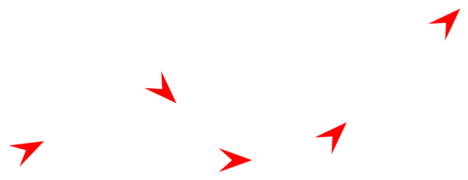
```
\begin{pspicture}(6,2)
\psset{arrowscale=2,ArrowFill=true}
\psline[ArrowInside=-*,ArrowInsidePos
  =0.25]{->}(0,0)(2,1)(3,0)(4,0)(6,2)
\end{pspicture}
```



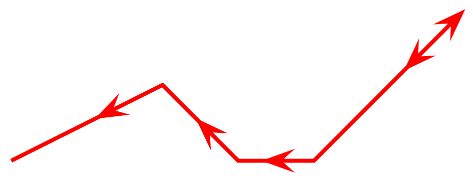
```
\begin{pspicture}(6,2)
\psset{arrowscale=2,ArrowFill=true}
\psline[ArrowInside=-*,ArrowInsidePos=0.25,
  ArrowInsideNo=2]{->}%
   (0,0)(2,1)(3,0)(4,0)(6,2)
\end{pspicture}
```



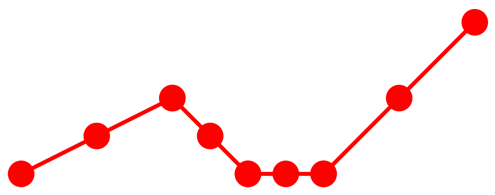
```
\begin{pspicture}(6,2)
\psset{arrowscale=2,ArrowFill=true}
\psline[ArrowInside=->, ArrowInsidePos=0.25]{->}%
        (0,0)(2,1)(3,0)(4,0)(6,2)
\end{pspicture}
```



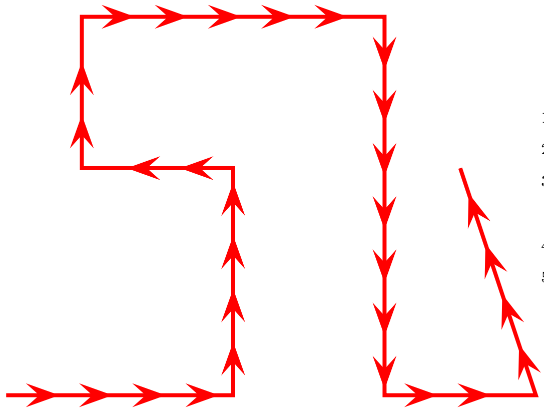
```
\begin{pspicture}(6,2)
\psset{arrowscale=2,ArrowFill=true}
\psline[linestyle=none,ArrowInside=->,
  ArrowInsidePos=0.25]{->}%
        (0,0)(2,1)(3,0)(4,0)(6,2)
\end{pspicture}
```



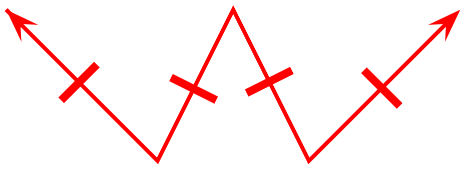
```
\begin{pspicture}(6,2)
\psset{arrowscale=2,ArrowFill=true}
\psline[ArrowInside=-<, ArrowInsidePos=0.75]{->}%
     (0,0)(2,1)(3,0)(4,0)(6,2)
\end{pspicture}
```



```
\begin{pspicture}(6,2)
\psset{arrowscale=2,ArrowFill=true,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)
\end{pspicture}
```

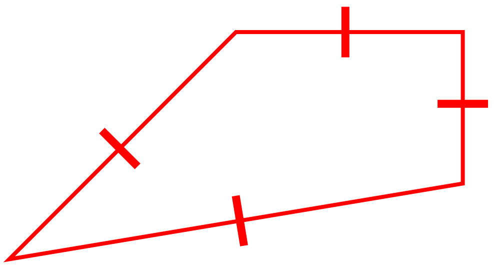


```
\begin{pspicture}(6,5)
\psset{arrowscale=2,ArrowFill=true}
\psline[ArrowInside=->,ArrowInsidePos=20](0,0)
  (3,0)%
       (3,3)(1,3)(1,5)(5,5)(5,0)(7,0)(6,3)
\end{pspicture}
```

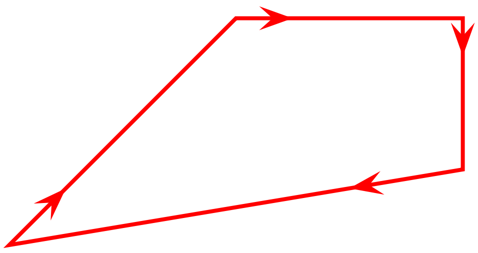


```
\begin{pspicture}(6,2)
\psset{arrowscale=2,ArrowFill=true}
\psline[ArrowInside=-|]{<->}(0,2)(2,0)(3,2)(4,0)
  (6,2)
\end{pspicture}
```

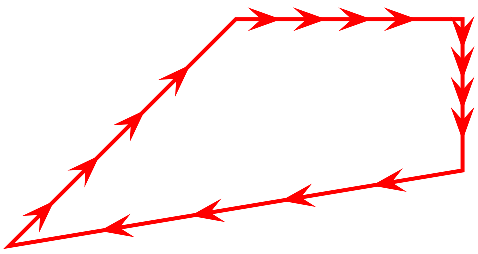
9.7.2 \pspolygon



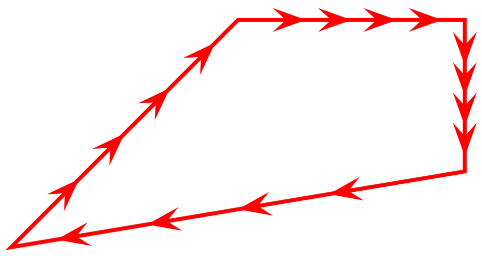
```
\begin{pspicture}(6,3)
\psset{arrowscale=2}
\pspolygon[ArrowInside=-|](0,0)(3,3)(6,3)(6,1)
\end{pspicture}
```



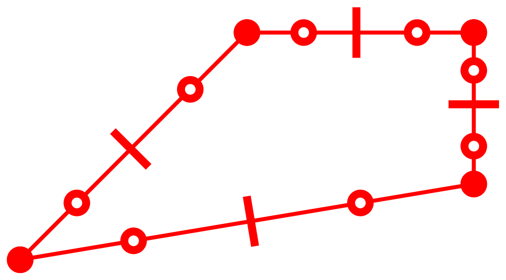
```
\begin{pspicture}(6,3)
\psset{arrowscale=2}
\pspolygon[ArrowInside=->,ArrowInsidePos=0.25]%
     (0,0)(3,3)(6,3)(6,1)
\end{pspicture}
```



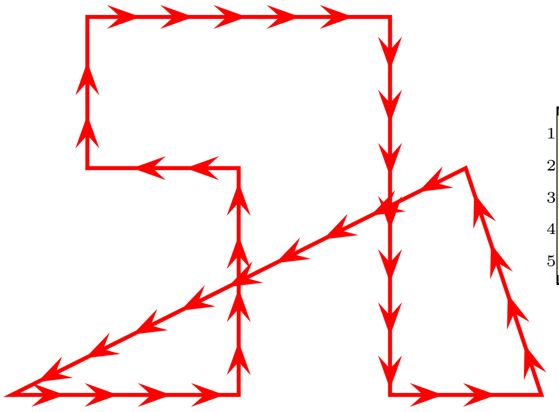
```
\begin{pspicture}(6,3)
\psset{arrowscale=2}
\pspolygon[ArrowInside=->,ArrowInsideNo=4]%
       (0,0)(3,3)(6,3)(6,1)
\end{pspicture}
```



```
\begin{pspicture}(6,3)
\psset{arrowscale=2}
\pspolygon[ArrowInside=->,ArrowInsideNo=4,%
  ArrowInsideOffset=0.1](0,0)(3,3)(6,3)(6,1)
\end{pspicture}
```

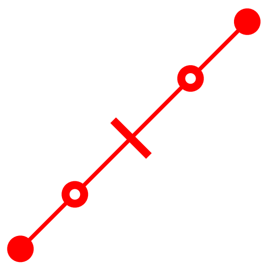


```
\begin{pspicture}(6,3)
\psset{arrowscale=2}
 \pspolygon[ArrowInside=-|](0,0)(3,3)(6,3)(6,1)
 \psset{linestyle=none,ArrowInside=-*}
 \pspolygon[ArrowInsidePos=0](0,0)(3,3)(6,3)(6,1)
 \pspolygon[ArrowInsidePos=1](0,0)(3,3)(6,3)(6,1)
 \psset{ArrowInside=-o}
 \pspolygon[ArrowInsidePos=0.25](0,0)(3,3)(6,3)
   (6,1)
 \pspolygon[ArrowInsidePos=0.75](0,0)(3,3)(6,3)
   (6,1)
\end{pspicture}
```

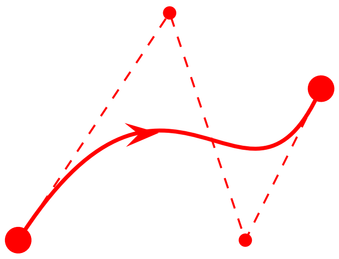


```
\begin{pspicture}(6,5)
\psset{arrowscale=2}
  \pspolygon[ArrowInside=->,ArrowInsidePos=20]%
    (0,0)(3,0)(3,3)(1,3)(1,5)(5,5)(5,0)(7,0)(6,3)
\end{pspicture}
```

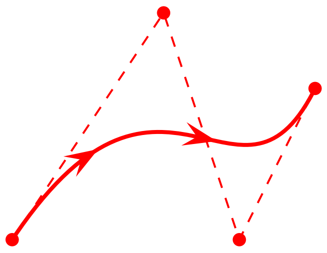
9.7.3 `\psbezier`



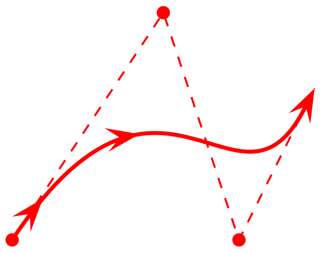
```
\begin{pspicture}(3,3)
\psset{arrowscale=2}
  \psbezier[ArrowInside=-|](1,1)(2,2)(3,3)
  \psset{linestyle=none,ArrowInside=-o}
  \psbezier[ArrowInsidePos=0.25](1,1)(2,2)(3,3)
  \psbezier[ArrowInsidePos=0.75](1,1)(2,2)(3,3)
  \psset{linestyle=none,ArrowInside=-*}
  \psbezier[ArrowInsidePos=0](1,1)(2,2)(3,3)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



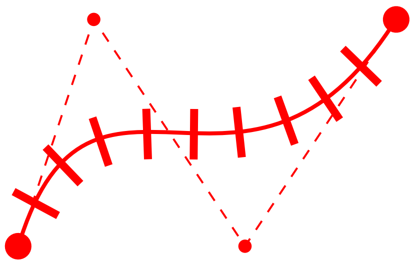
```
\begin{pspicture}(4,3)
\psset{arrowscale=2}
  \psbezier[ArrowInside=->,showpoints=true]%
     {*-*}(2,3)(3,0)(4,2)
\end{pspicture}
```



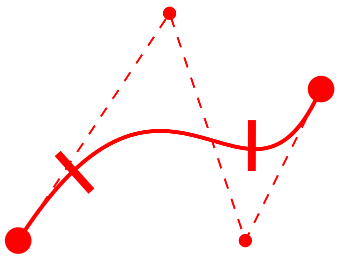
```
\begin{pspicture}(4,3)
\psset{arrowscale=2}
  \psbezier[ArrowInside=->,showpoints=true,%
      ArrowInsideNo=2](2,3)(3,0)(4,2)
\end{pspicture}
```



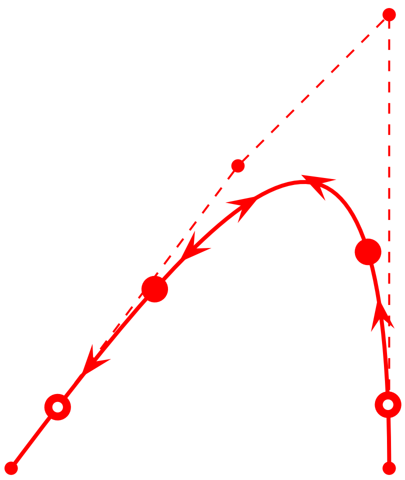
```
\begin{pspicture}(4,3)
\psset{arrowscale=2}
  \psbezier[ArrowInside=->,showpoints=true,%
      ArrowInsideNo=2,ArrowInsideOffset=-0.2]{->}(2,3)(3,0)
        (4,2)
\end{pspicture}
```



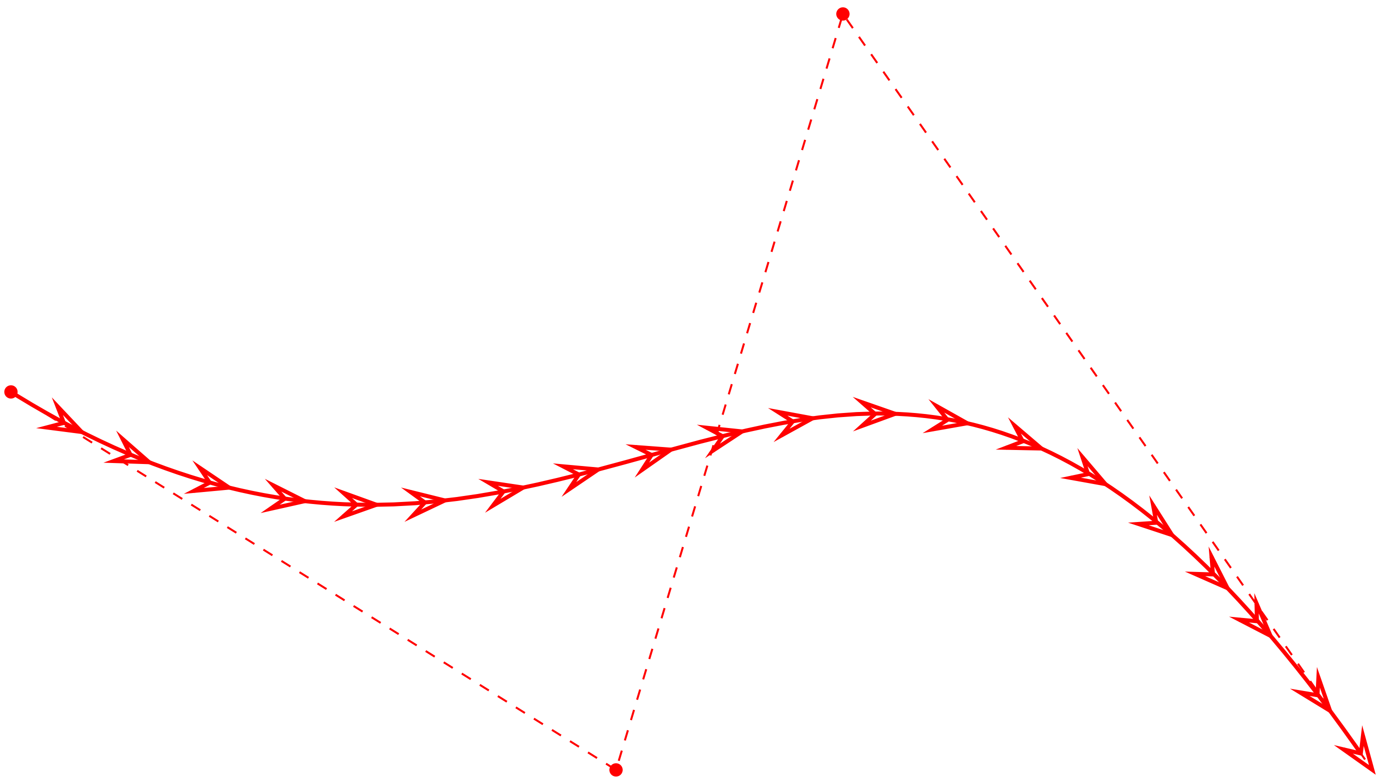
```
\begin{pspicture}(5,3)
\psset{arrowscale=2}
  \psbezier[ArrowInsideNo=9,ArrowInside=-|,%
     showpoints=true]{*-*}(1,3)(3,0)(5,3)
\end{pspicture}
```



```
\begin{pspicture}(4,3)
\psset{arrowscale=2}
  \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)
\end{pspicture}
```



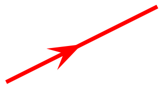
```
\begin{pspicture}(5,6)
\psset{arrowscale=2}
  \pnode(3,4){A}\pnode(5,6){B}\pnode(5,0){C}
  \psbezier[ArrowInside=->,%
     showpoints=true](A)(B)(C)
  \psset{linestyle=none,ArrowInside=-<}
  \psbezier[ArrowInsideNo=4](A)(B)(C)
  \psset{ArrowInside=-o}
  \psbezier[ArrowInsidePos=0.1](A)(B)(C)
  \psbezier[ArrowInsidePos=0.9](A)(B)(C)
  \psset{ArrowInside=-*}
  \psbezier[ArrowInsidePos=0.3](A)(B)(C)
  \psbezier[ArrowInsidePos=0.7](A)(B)(C)
\end{pspicture}
```



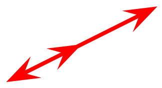
```
\begin{pspicture}(-3,-5)(15,5)
  \psbezier[ArrowInsideNo=19,%
      ArrowInside=->,ArrowFill=false,%
      showpoints=true]{->}(-3,0)(5,-5)(8,5)(15,-5)
\end{pspicture}
```

9.7.4 \pcline

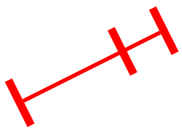
These examples need the package pst-node.



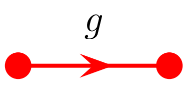
```
\begin{pspicture}(2,1)
\psset{arrowscale=2}
\pcline[ArrowInside=->](0,0)(2,1)
\end{pspicture}
```



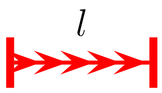
```
\begin{pspicture}(2,1)
\psset{arrowscale=2}
\pcline[ArrowInside=->]{<->}(0,0)(2,1)
\end{pspicture}
```



```
\begin{pspicture}(2,1)
\psset{arrowscale=2}
\pcline[ArrowInside=-|,ArrowInsidePos=0.75]{|-|}(0,0)(2,1)
\end{pspicture}
```



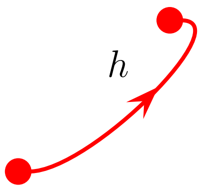
```
\psset{arrowscale=2}
\pcline[ArrowInside=->,ArrowInsidePos=0.65]{*-*}(0,0)(2,0)
\naput[labelsep=0.3]{\large$g$}
```



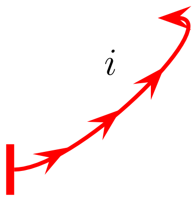
```
\psset{arrowscale=2}
\pcline[ArrowInside=->,ArrowInsidePos=10]{|-|}(0,0)(2,0)
\naput[labelsep=0.3]{\large$l$}
```

9.7.5 \pccurve

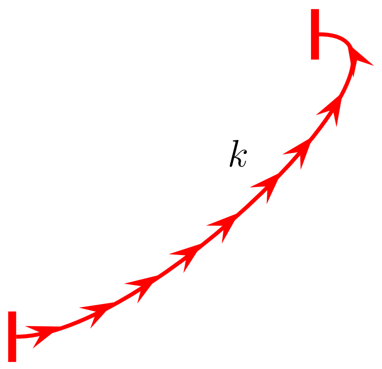
These examples also need the package pst-node.



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



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



```
\begin{pspicture}(4,4)
\psset{arrowscale=2}
\pccurve[ArrowInside=->,ArrowInsidePos=20]{|-|}(0,0)(4,4)
\naput[labelsep=0.3]{\large$k$}
\end{pspicture}
```

10 \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 any non-number character.

11 Color

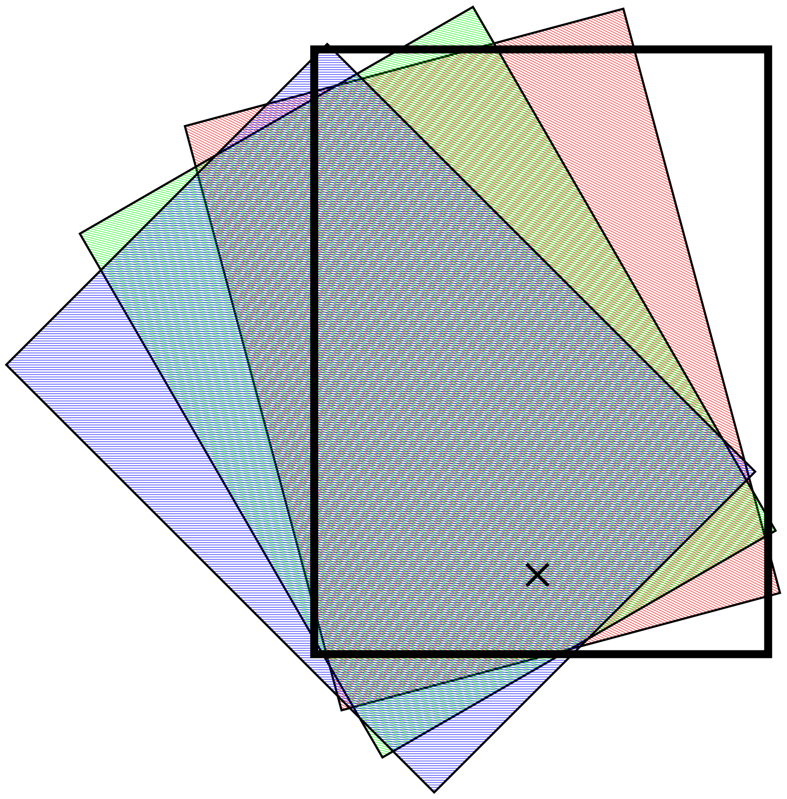
11.1 „Tranparent colors"

`pstricks-add` simulates transparency with hatch lines:

```
\def\defineTColor{\@ifnextchar[{\defineTColor@i}{\defineTColor@i[]}}
\def\defineTColor@i[#1]#2#3{%      transparency "Colors"
  \newpsstyle{#2}{%
     fillstyle=vlines,hatchwidth=0.1\pslinewidth,
     hatchsep=1\pslinewidth,hatchcolor=#3,#1%
  }%
}
\defineTColor{TRed}{red}
```

```
\defineTColor{TGreen}{green}
\defineTColor{TBlue}{blue}
```

There are three predefined "'transparent"' colors `TRed`, `TGreen`, `TBlue`. They are used as `PSTricks`styles and not as colors:



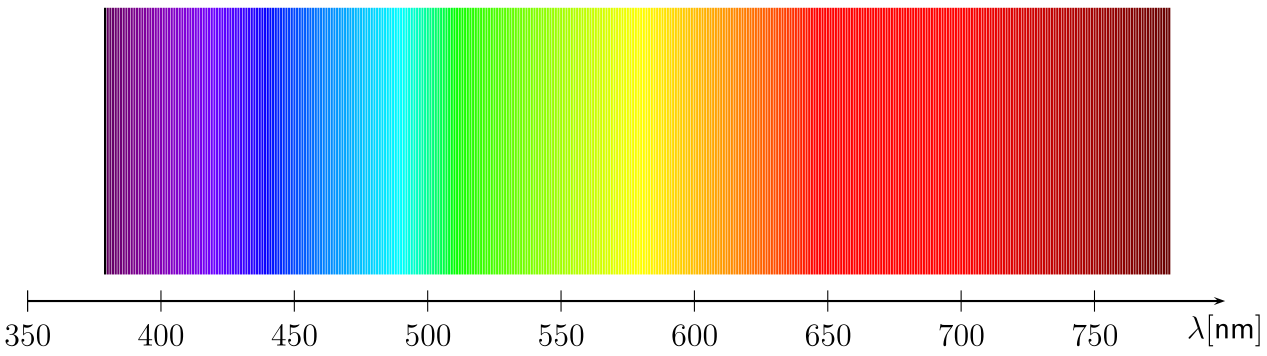
```
\begin{pspicture}(-3,-5)(5,5)
\psframe(-1,-3)(5,5) % objet de base
\psrotate(2,-2){15}{%
  \psframe[style=TRed](-1,-3)(5,5)}
\psrotate(2,-2){30}{%
  \psframe[style=TGreen](-1,-3)(5,5)}
\psrotate(2,-2){45}{%
  \psframe[style=TBlue](-1,-3)(5,5)}
\psframe[linewidth=3pt](-1,-3)(5,5)
\psdots[dotstyle=+,dotangle=45,dotscale=3](2,-2) % centre de la rotation
\end{pspicture}
```

11.2 Calculated colors

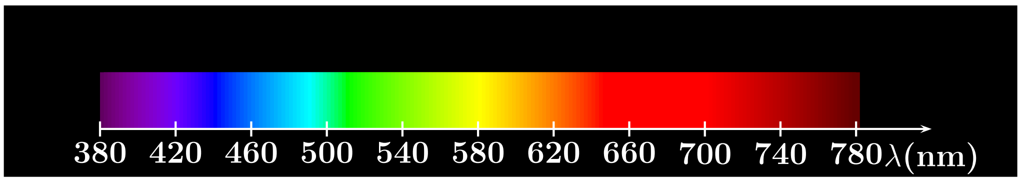
The `xcolor` package (version 2.6) has a new feature for defining colors:

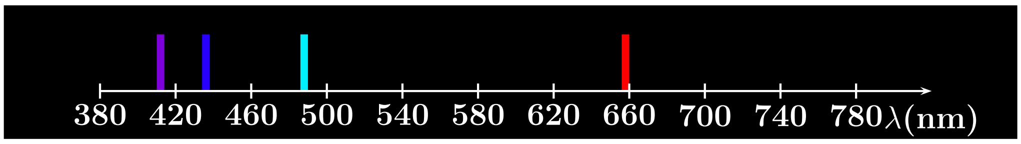
```
\definecolor[ps]{<name>}{<model>}{< PS code >}
```

`model` can be one of the color models, which PostScript will understand, e.g. `rgb`. With this definition the color is calculated on PostScript side.



```
\definecolor[ps]{bl}{rgb}{tx@addDict begin Red Green Blue end}%
\psset{unit=1bp}
\begin{pspicture}(0,-30)(400,100)
\multido{\iLAMBDA=0+1}{400}{%
  \pstVerb{
    \iLAMBDA\space 379 add dup /lambda exch def
    tx@addDict begin wavelengthToRGB end
  }%
  \psline[linecolor=bl](\iLAMBDA,0)(\iLAMBDA,100)%
}
\psaxes[yAxis=false,Ox=350,dx=50bp,Dx=50]{->}(-29,-10)(420,100)
\uput[-90](420,-10){$\lambda$[\textsf{nm}]}
\end{pspicture}
```





Spectrum of hydrogen emission (Manuel Luque)

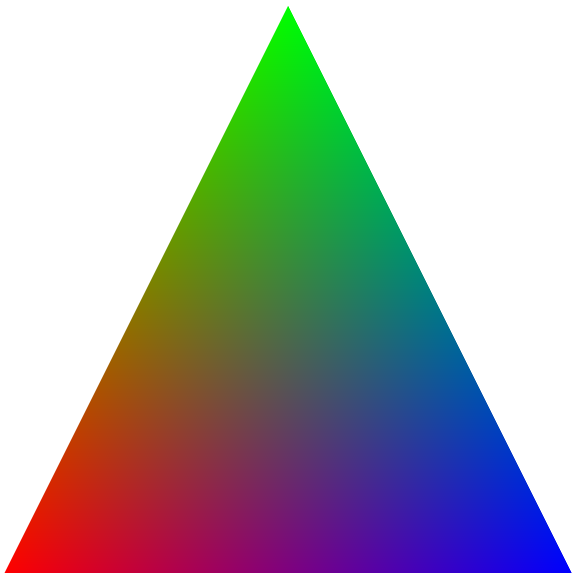
```
\newcommand{\Touch}{%
\psframe[linestyle=none,fillstyle=solid,fillcolor=bl,dimen=middle
  ](0.1,0.75)}
\definecolor[ps]{bl}{rgb}{tx@addDict begin Red Green Blue end}%
% Echelle 1cm <-> 40 nm
%          1 nm <-> 0.025 cm
\psframebox[fillstyle=solid,fillcolor=black]{%
\begin{pspicture}(-1,-0.5)(12,1.5)
\multido{\iLAMBDA=380+2}{200}{%
  \pstVerb{
    /lambda \iLAMBDA\space def
    lambda
    tx@addDict begin  wavelengthToRGB end
  }%
 \rput(! lambda 0.025 mul 9.5 sub 0){\Touch}
}
\multido{\n=0+1,\iDiv=380+40}{11}{%
    \psline[linecolor=white](\n,0.1)(\n,-0.1)
    \uput[270](\n,0){\textbf{\white\iDiv}}}
    \psline[linecolor=white]{->}(11,0)
    \uput[270](11,0){\textbf{\white$\lambda$(nm)}}
\end{pspicture}}

\psframebox[fillstyle=solid,fillcolor=black]{%
\begin{pspicture}(-1,-0.5)(12,1)
  \pstVerb{
    /lambda 656 def
    lambda
    tx@addDict begin  wavelengthToRGB end
  }%
 \rput(! 656 0.025 mul 9.5 sub 0){\Touch}
  \pstVerb{
    /lambda 486 def
    lambda
    tx@addDict begin  wavelengthToRGB end
  }%
 \rput(! 486 0.025 mul 9.5 sub 0){\Touch}
   \pstVerb{
    /lambda 434 def
    lambda
    tx@addDict begin  wavelengthToRGB end
  }%
 \rput(! 434 0.025 mul 9.5 sub 0){\Touch}
  \pstVerb{
    /lambda 410 def
    lambda
    tx@addDict begin  wavelengthToRGB end
  }%
 \rput(! 410 0.025 mul 9.5 sub 0){\Touch}
\multido{\n=0+1,\iDiv=380+40}{11}{%
    \psline[linecolor=white](\n,0.1)(\n,-0.1)
    \uput[270](\n,0){\textbf{\white\iDiv}}
    \psline[linecolor=white]{->}(11,0)
    \uput[270](11,0){\textbf{\white$\lambda$(nm)}}
\end{pspicture}}

Spectrum of hydrogen emission (Manuel Luque)
```

11.3 Gouraud shading

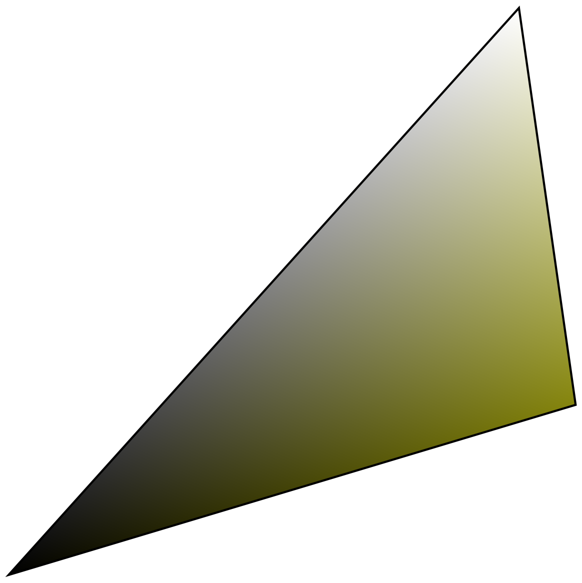
Gouraud shading is a method used in computer graphics to simulate the differing effects of light and colour across the surface of an object. In practice, Gouraud shading is used to achieve smooth lighting on low-polygon surfaces without the heavy computational requirements of calculating lighting for each pixel. The technique was first presented by Henri Gouraud in 1971.
http://www.wikipedia.org

PostScript level 3 supports this kind of shading and it could only be seen with Acroread 7 or younger. Die Syntax ist relativ einfach

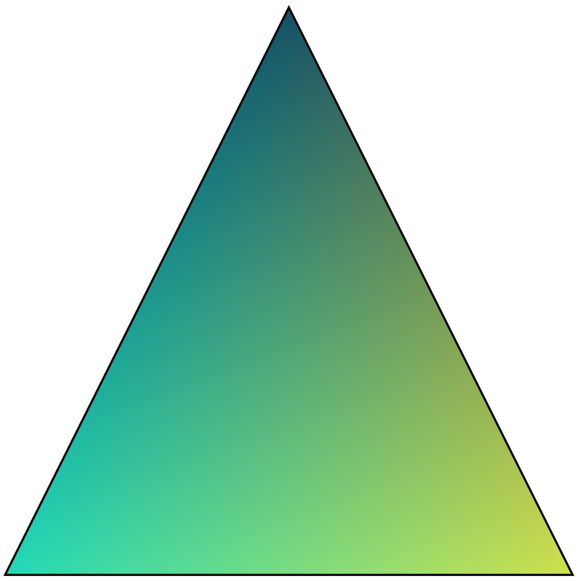
```
\psGTriangle(x1,y1)(x2,y2)(x3,y3){color1}{color2}{color3}
```



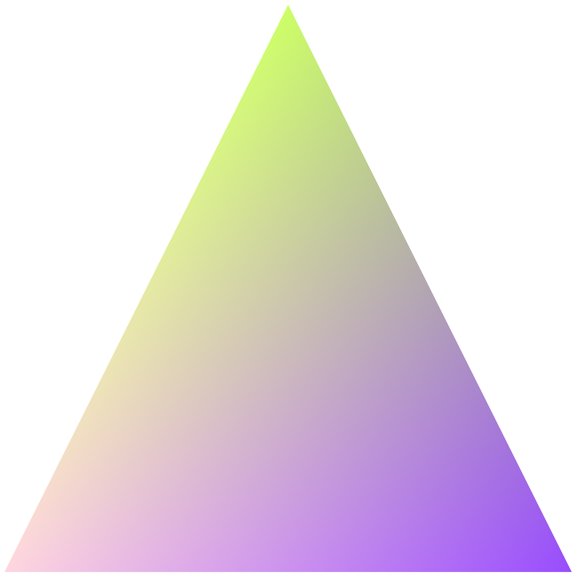
```
\begin{pspicture}(0,-.25)(10,10)
  \psGTriangle(0,0)(5,10)(10,0){red}{green}{blue}
\end{pspicture}
```



```
\begin{pspicture}(0,-.25)(10,10)
  \psGTriangle*(0,0)(9,10)(10,3){black}{white!50}{red!50!green!95}
\end{pspicture}
```



```
\begin{pspicture}(0,-.25)(10,10)
  \psGTriangle*(0,0)(5,10)(10,0){-red!100!green!84!blue!86}
                              {-red!80!green!100!blue!40}
                              {-red!60!green!30!blue!100}
\end{pspicture}
```



```
\definecolor{rose}{rgb}{1.00, 0.84, 0.88}
\definecolor{vertpommepasmure}{rgb}{0.80, 1.0, 0.40}
\definecolor{fushia}{rgb}{0.60, 0.30, 1.0}
\begin{pspicture}(0,-.25)(10,10)
  \psGTriangle(0,0)(5,10)(10,0){rose}{vertpommepasmure}{fushia}
\end{pspicture}
```

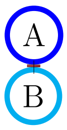
Part II

pst-node

12 \nclineII

The dashed lines are black and white by default. 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}
```

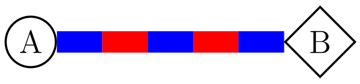
12.1 The options

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

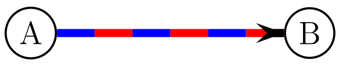
name	meaning
`dashColorI`	first color, default is `black`
`dashColorII`	second color, default is `red`
`dashNo`	The ratio of dashColorI to dashColorII, the default is 0.2

`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!

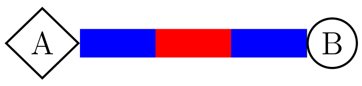
12.2 Examples



```
\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}
```

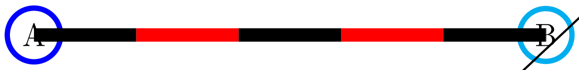


```
\dianode{A}{A}\hspace*{3cm}\circlenode{B}{B}%
\nclineII[dashColorI=blue,linecap=1,dashNo=0.3,linewidth
  =0.5]{A}{B}
```

13 \pclineII

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

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



```
\circlenode[linecolor=blue,linewidth=2pt
  ]{A}{A}%
\hspace*{6cm}\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`.

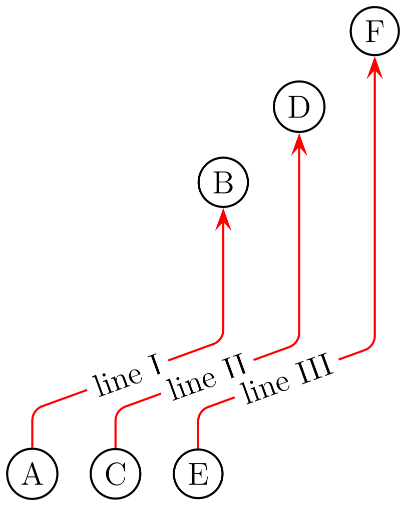
```
\raggedright This macro makes only sense
   when connecting two ''invisible''
  nodes,
like this connection from here\pnode{D}\
  pclineII{->}(D)(C){}
to the above word \verb|pstricks|.
```

14 \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 calculated when an angle `lineAngle` is defined. This angle is the gradient of the intermediate line between the two arms. The syntax of `ncdiag` is

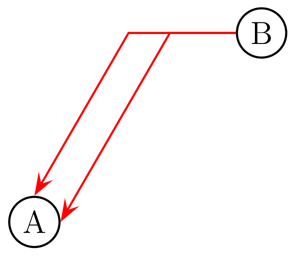
```
\ncdiag[<options>]{<Node A>}{<Node B>}
\pcdiag[<options>](<Node A>)(<Node B>)
```

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

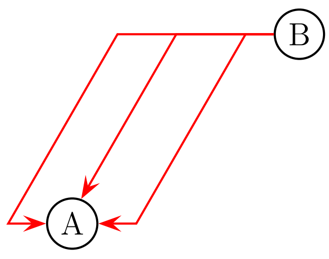


```
\begin{pspicture}(5,6)
  \circlenode{A}{A}\quad\circlenode{C}{C}%
    \quad\circlenode{E}{E}
  \rput(0,4){\circlenode{B}{B}}
  \rput(1,5){\circlenode{D}{D}}
  \rput(2,6){\circlenode{F}{F}}
  \psset{arrowscale=2,linearc=0.2,%
    linecolor=red,armA=0.5, angleA=90,angleB=-90}
  \ncdiag[lineAngle=20]{->}{A}{B}
  \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}
\end{pspicture}
```

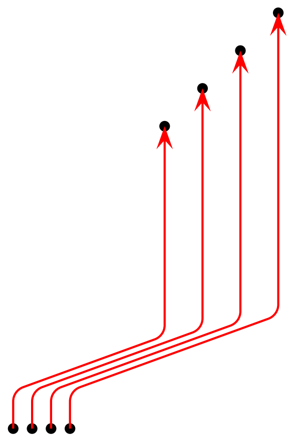
The `ncdiag` macro sets the `armB` dynamically to the calculated value. Any user setting of `armB` is overwritten by the macro. The `armA` could be set to a zero length:



```
\begin{pspicture}(4,3)
  \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}}
\end{pspicture}
```



```
\begin{pspicture}(4,3)
  \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}}
\end{pspicture}
```

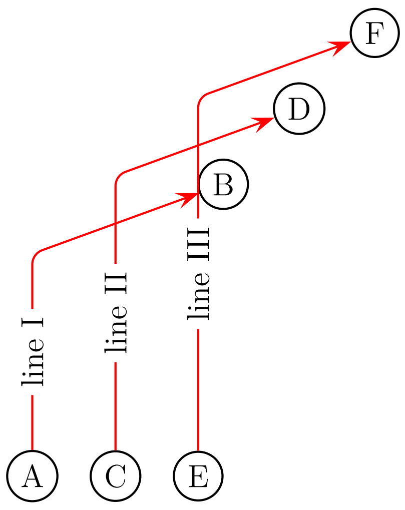


```
\begin{pspicture}(4,5.5)
  \cnode*(0,0){2pt}{A}%
  \cnode*(0.25,0){2pt}{C}%
  \cnode*(0.5,0){2pt}{E}%
  \cnode*(0.75,0){2pt}{G}%
  \cnode*(2,4){2pt}{B}%
  \cnode*(2.5,4.5){2pt}{D}%
  \cnode*(3,5){2pt}{F}%
  \cnode*(3.5,5.5){2pt}{H}%
  {\psset{arrowscale=2,linearc=0.2,%
    linecolor=red,armA=0.5,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

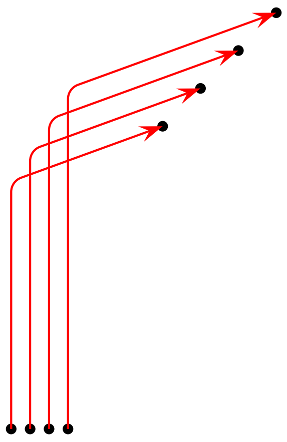
15 `\ncdiagg` and `\pcdiagg`

This is nearly the same than `\ncdiag` except that `armB=0` and the `angleB` value is computed 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>)
```

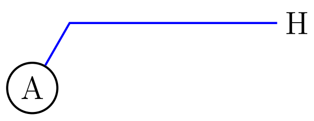


```
\begin{pspicture}(4,6)
  \psset{linecolor=black}
  \circlenode{A}{A}%
  \quad\circlenode{C}{C}%
  \quad\circlenode{E}{E}
  \rput(0,4){\circlenode{B}{B}}
  \rput(1,5){\circlenode{D}{D}}
  \rput(2,6){\circlenode{F}{F}}
  {\psset{arrowscale=2,linearc=0.2,linecolor=red,armA=0.5,
     angleA=90}
  \ncdiagg[lineAngle=-160]{->}{A}{B}
  \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}}
\end{pspicture}
```

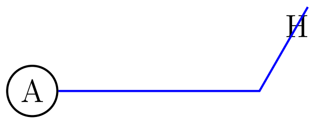


```
\begin{pspicture}(4,6)
  \psset{linecolor=black}
  \cnode*(0,0){2pt}{A}%
  \cnode*(0.25,0){2pt}{C}%
  \cnode*(0.5,0){2pt}{E}%
  \cnode*(0.75,0){2pt}{G}%
  \cnode*(2,4){2pt}{B}%
  \cnode*(2.5,4.5){2pt}{D}%
  \cnode*(3,5){2pt}{F}%
  \cnode*(3.5,5.5){2pt}{H}%
  {\psset{arrowscale=2,linearc=0.2,linecolor=red,armA=0.5,
     angleA=90}
  \pcdiagg[lineAngle=20]{->}(A)(B)
  \pcdiagg[lineAngle=20]{->}(C)(D)
  \pcdiagg[lineAngle=20]{->}(E)(F)
  \pcdiagg[lineAngle=20]{->}(G)(H)}
\end{pspicture}
```

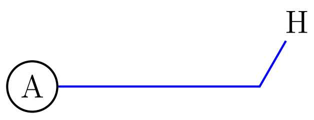
The only catch for `\ncdiagg` is, that you need the right value for `lineAngle`. If the node connection is on the wrong side of the second node, then choose the corresponding angle, e.g.: if 20 is wrong then take −160, the corresponding to 180.



```
\begin{pspicture}(4,1.5)
  \circlenode{a}{A}
  \rput[l](3,1){\rnode{b}{H}}
  \ncdiagg[lineAngle=60,angleA=180,armA=.5,nodesepA=3pt,
    linecolor=blue]{b}{a}
\end{pspicture}
```



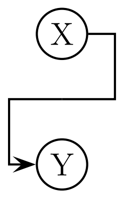
```
\begin{pspicture}(4,1.5)
  \circlenode{a}{A}
  \rput[l](3,1){\rnode{b}{H}}
  \ncdiagg[lineAngle=60,armA=.5,nodesepB=3pt,linecolor=blue]{a
    }{b}
\end{pspicture}
```



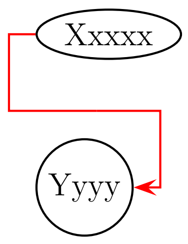
```
\begin{pspicture}(4,1.5)
  \circlenode{a}{A}
  \rput[l](3,1){\rnode{b}{H}}
  \ncdiagg[lineAngle=-120,armA=.5,nodesepB=3pt,linecolor=blue]{
    a}{b}
\end{pspicture}
```

16 \ncbarr

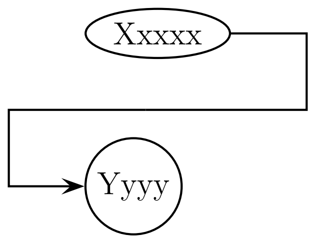
This has the same behaviour as `ncbar`, but has 5 segments and all are horizontal ones. This is the reason why `angleA` must be 0 or alternative 180. All other values are set to 0 by the macro. The intermediate horizontal line is symmetrical to the distance of the two nodes.



```
\psset{arrowscale=2}%
\circlenode{X}{X}\\[1cm]
\circlenode{Y}{Y}
\ncbarr[angleA=0,arrows=->,arrowscale=2]{X}{Y}
```



```
\psset{arrowscale=2}%
\ovalnode{X}{Xxxxx}\\[1cm]
\circlenode{Y}{Yyyy}
\ncbarr[angleA=180,arrows=->,arrowscale=2,linecolor=red]{X}{Y}
```



```
\psset{arrowscale=2}%
\ovalnode{X}{Xxxxx}\\[1cm]
\circlenode{Y}{Yyyy}
\ncbarr[angleA=20,arm=1cm,arrows=->,arrowscale=2]{X}{Y}
```

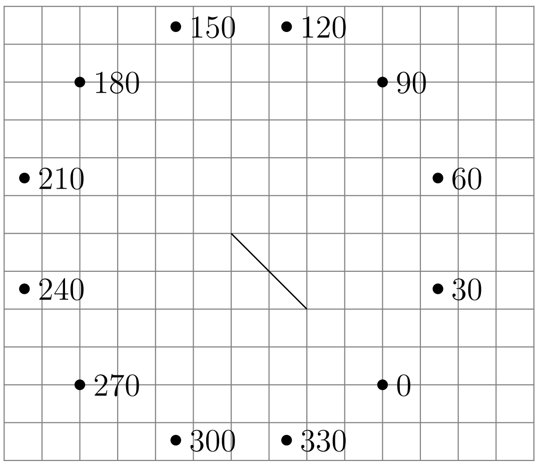
17 \psRelNode

With this macro it is possible to put a node relative to a given line. Parameter are the angle and the length factor:

```
\psRelNode(<P0>)(<P1>){<length factor>}{<end node name>}
\psRelLine[<options>](<P0>)(<P1>){<length factor>}{<end node name>}
```

The length factor depends to the distance of $\overline{P_0P_1}$ and the end node name must be a valid nodename and shouldn't contain any of the special PostScript characters. There are two valid options:

name	default	meaning
`angle`	0	angle between the given line $\overline{P_0P_1}$ and the new one $\overline{P_0P_endNode}$
`trueAngle`	false	defines whether the angle depends to the seen line or to the mathematical one, which respect the scaling factors `xunit` and `yunit`.



```
\begin{pspicture}(7,6)
  \psgrid[gridwidth=0pt,gridcolor=gray,
    gridlabels=0pt,subgriddiv=2]
  \pnode(3,3){A}\pnode(4,2){B}
  \psline[nodesep=-3,linewidth=0.5pt](A)(B)
  \multido{\iA=0+30}{12}{%
    \psRelNode[angle=\iA](A)(B){2}{C}%
    \qdisk(C){2pt}
    \uput[0](C){\iA}}
\end{pspicture}
```

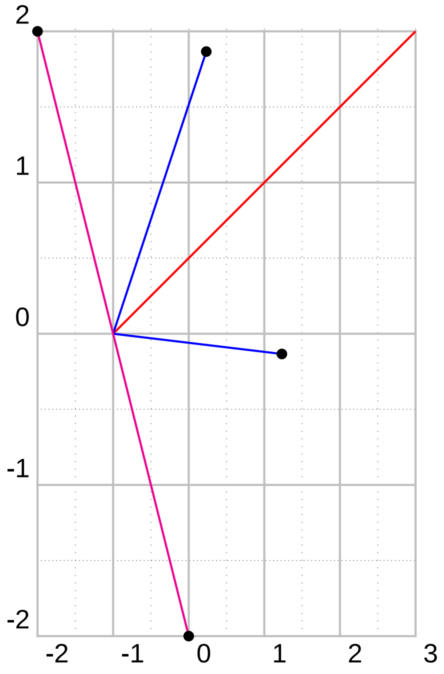
18 \psRelLine

With this macro it is possible to plot lines relative to a given one. Parameter are the angle and the length factor:

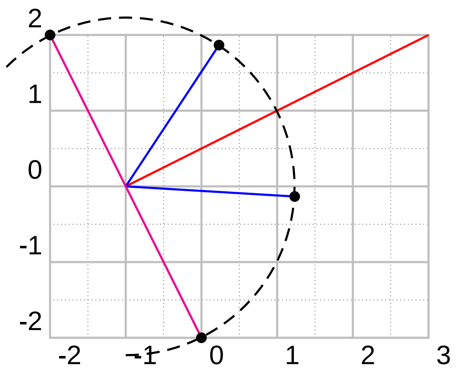
```
\psRelLine(<P0>)(<P1>){<length factor>}{<end node name>}
\psRelLine{<arrows>}(<P0>)(<P1>){<length factor>}{<end node name>}
\psRelLine[<options>](<P0>)(<P1>){<length factor>}{<end node name>}
\psRelLine[<options>]{<arrows>}(<P0>)(<P1>){<length factor>}{<end node name>}
```

The length factor depends to the distance of $\overline{P_0P_1}$ and the end node name must be a valid nodename and shouldn't contain any of the special PostScript characters. There are two valid options which are described in the forgoing section for `\psRelNode`.

The following two figures show the same, the first one with a scaling different to 1 : 1, this is the reason why the end points are on an ellipse and not on a circle like in the second figure.

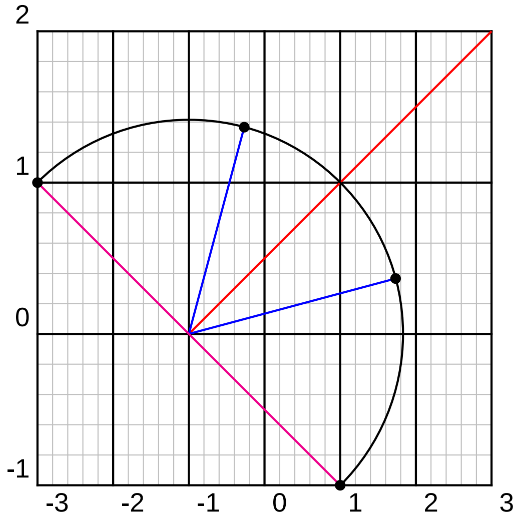


```
\psset{yunit=2,xunit=1}
\begin{pspicture}(-2,-2)(3,2)
\psgrid[subgriddiv=2,subgriddots=10,gridcolor=lightgray]
\pnode(-1,0){A}\pnode(3,2){B}
\psline[linecolor=red](A)(B)
\psRelLine[linecolor=blue,angle=30](-1,0)(B){0.5}{EndNode}
\qdisk(EndNode){2pt}
\psRelLine[linecolor=blue,angle=-30](A)(B){0.5}{EndNode}
\qdisk(EndNode){2pt}
\psRelLine[linecolor=magenta,angle=90](-1,0)(3,2){0.5}{
  EndNode}
\qdisk(EndNode){2pt}
\psRelLine[linecolor=magenta,angle=-90](A)(B){0.5}{EndNode
  }
\qdisk(EndNode){2pt}
\end{pspicture}
```



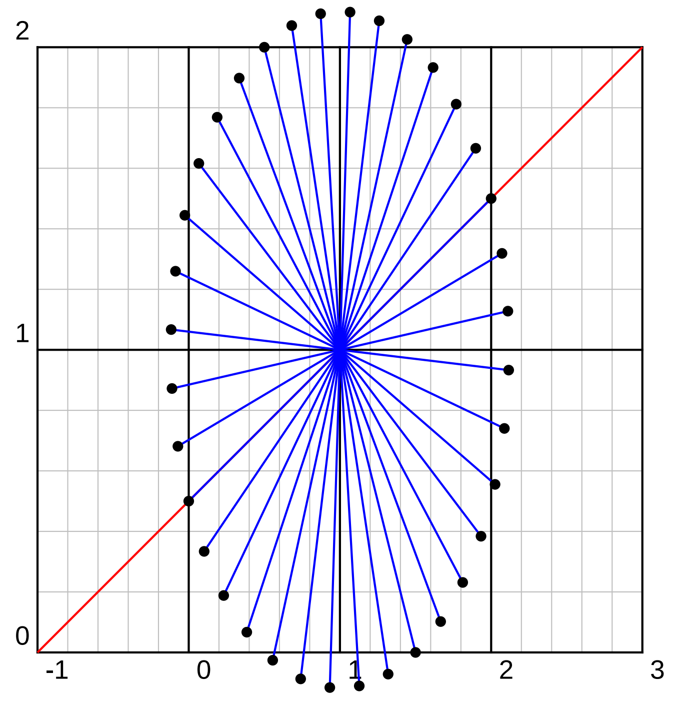
```
\begin{pspicture}(-2,-2)(3,2)
\psgrid[subgriddiv=2,subgriddots=10,gridcolor=lightgray]
\pnode(-1,0){A}\pnode(3,2){B}
\psline[linecolor=red](A)(B)
\psarc[linestyle=dashed](A){2.23}{-90}{135}
\psRelLine[linecolor=blue,angle=30](-1,0)(B){0.5}{EndNode}
\qdisk(EndNode){2pt}
\psRelLine[linecolor=blue,angle=-30](A)(B){0.5}{EndNode}
\qdisk(EndNode){2pt}
\psRelLine[linecolor=magenta,angle=90](-1,0)(3,2){0.5}{
  EndNode}
\qdisk(EndNode){2pt}
\psRelLine[linecolor=magenta,angle=-90](A)(B){0.5}{EndNode
  }
\qdisk(EndNode){2pt}
\end{pspicture}
```

The following figure has also a different scaling, but has set the option `trueAngle`, all angles depends to what "you see".

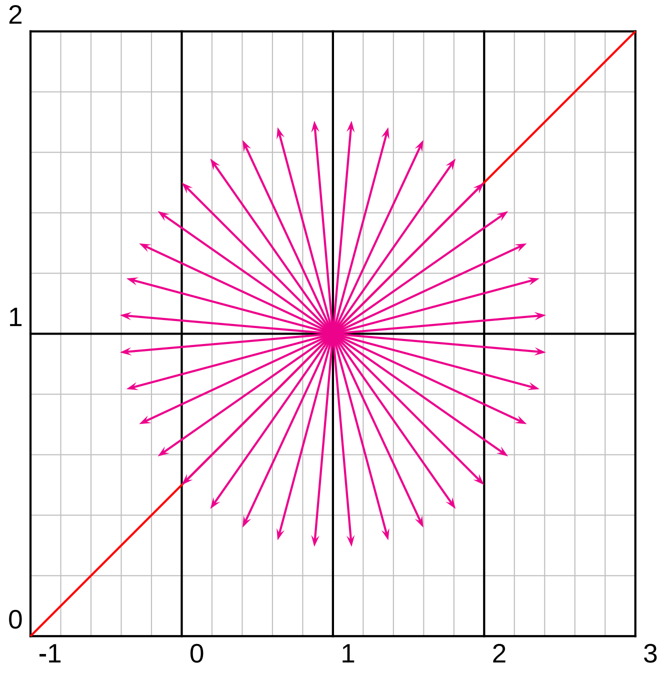


```
\psset{yunit=2,xunit=1}
\begin{pspicture}(-3,-1)(3,2)\psgrid[subgridcolor=
  lightgray]
\pnode(-1,0){A}\pnode(3,2){B}
\psline[linecolor=red](A)(B)
\psarc(A){2.83}{-45}{135}
\psRelLine[linecolor=blue,angle=30,trueAngle](A)(B
  ){0.5}{EndNode}
\qdisk(EndNode){2pt}
\psRelLine[linecolor=blue,angle=-30,trueAngle](A)(
  B){0.5}{EndNode}
\qdisk(EndNode){2pt}
\psRelLine[linecolor=magenta,angle=90,trueAngle](A
  )(B){0.5}{EndNode}
\qdisk(EndNode){2pt}
\psRelLine[linecolor=magenta,angle=-90,trueAngle](
  A)(B){0.5}{EndNode}
\qdisk(EndNode){2pt}
\end{pspicture}
```

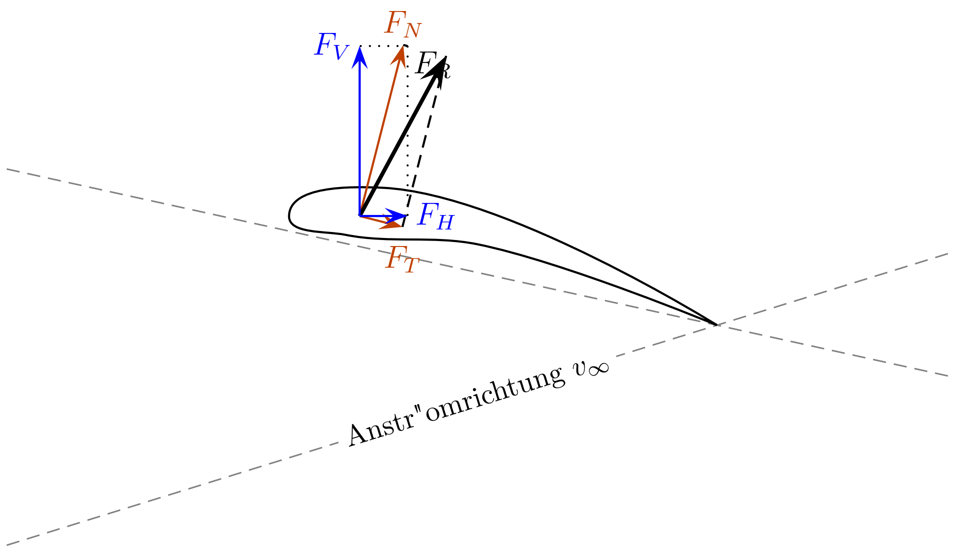
Two examples with using `\multido` to show the behaviour of the options `trueAngle` and `angle`.



```
\psset{yunit=4,xunit=2}
\begin{pspicture}(-1,0)(3,2)\psgrid[
  subgridcolor=lightgray]
\pnode(-1,0){A}\pnode(1,1){B}
\psline[linecolor=red](A)(3,2)
\multido{\iA=0+10}{36}{%
  \psRelLine[linecolor=blue,angle=\iA](B)(
    A){-0.5}{EndNode}
  \qdisk(EndNode){2pt}
}
\end{pspicture}
```



```
\psset{yunit=4,xunit=2}
\begin{pspicture}(-1,0)(3,2)\psgrid[
  subgridcolor=lightgray]
\pnode(-1,0){A}\pnode(1,1){B}
\psline[linecolor=red](A)(3,2)
\multido{\iA=0+10}{36}{%
  \psRelLine[linecolor=magenta,angle=\iA,
    trueAngle]{->}(B)(A){-0.5}{EndNode}
}
\end{pspicture}
```



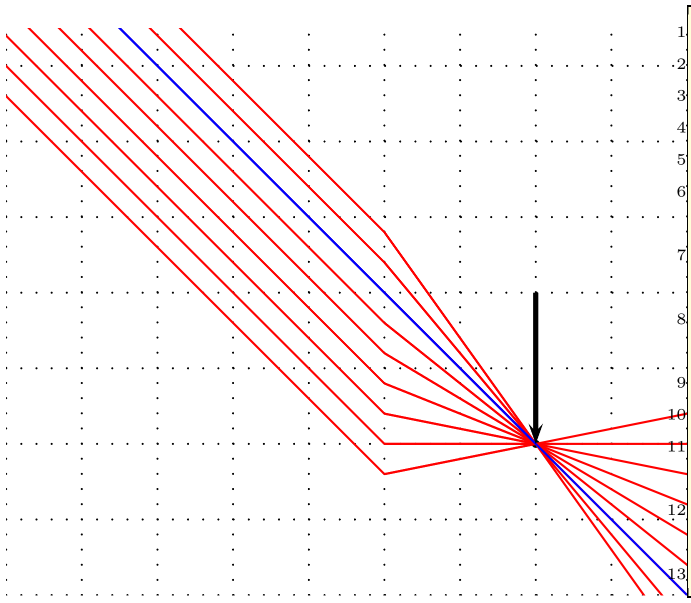
```
\psset{xunit=0.75\linewidth,yunit=0.75\linewidth,trueAngle}%
\end{center}
\begin{pspicture}(1,0.6)%\psgrid
  \pnode(.3,.35){Vk} \pnode(.375,.35){D} \pnode(0,.4){DST1} \pnode
    (1,.18){DST2}
  \pnode(0,.1){A1}   \pnode(1,.31){A1}
  { \psset{linewidth=.02,linestyle=dashed,linecolor=gray}%
    \pcline(DST1)(DST2) % <- Druckseitentangente
    \pcline(A2)(A1) % <- Anstr"omrichtung
    \lput*{:U}{\small Anstr"omrichtung $v_{\infty}$} }%
  \psIntersectionPoint(A1)(A2)(DST1)(DST2){Hk}
  \pscurve(Hk)(.4,.38)(Vk)(.36,.33)(.5,.32)(Hk)
  \psParallelLine[linecolor=red!75!green,arrows=->,arrowscale=2](Vk)(Hk)
    (D){.1}{FtE}
  \psRelLine[linecolor=red!75!green,arrows=->,arrowscale=2,angle=90](D)(
    FtE){4}{Fn}% why "4"?
  \psParallelLine[linestyle=dashed](D)(FtE)(Fn){.1}{Fnr1}
  \psRelLine[linestyle=dashed,angle=90](FtE)(D){-4}{Fnr2} % why "-4"?
  \psline[linewidth=1.5pt,arrows=->,arrowscale=2](D)(Fnr2)
  \psIntersectionPoint(D)([nodesep=2]D)(Fnr1)([offset=-4]Fnr1){Fh}
  \psIntersectionPoint(D)([offset=2]D)(Fnr1)([nodesep=4]Fnr1){Fv}
  \psline[linecolor=blue,arrows=->,arrowscale=2](D)(Fh)
  \psline[linecolor=blue,arrows=->,arrowscale=2](D)(Fv)
  \psline[linestyle=dotted](Fh)(Fnr1)  \psline[linestyle=dotted](Fv)(Fnr
    1)
  \uput{.1}[0](Fh){\blue $F_{H}$}   \uput{.1}[180](Fv){\blue $F_{V}$}
  \uput{.1}[-45](Fnr1){$F_{R}$}     \uput{.1}[90](Fn){\color{red!75!
    green}$F_{N}$}
  \uput{.25}[-90](FtE){\color{red!75!green}$F_{T}$}
\end{pspicture}
```

19 \psParallelLine

With this macro it is possible to plot lines relative to a given one, which is parallel. There is no special parameter here.

```
\psParallelLine(<P0>)(<P1>)(<P2>){<length>}{<end node name>}
\psParallelLine{<arrows>}(<P0>)(<P1>)(<P2>){<length>}{<end node name>}
\psParallelLine[<options>](<P0>)(<P1>)(<P2>){<length>}{<end node name>}
\psParallelLine[<options>]{<arrows>}(<P0>)(<P1>)(<P2>){<length>}{<end node name>}
```

The line starts at P_2, is parallel to $\overline{P_0P_1}$ and the length of this parallel line depends to the length factor. The end node name must be a valid nodename and shouldn't contain any of the special PostScript characters.

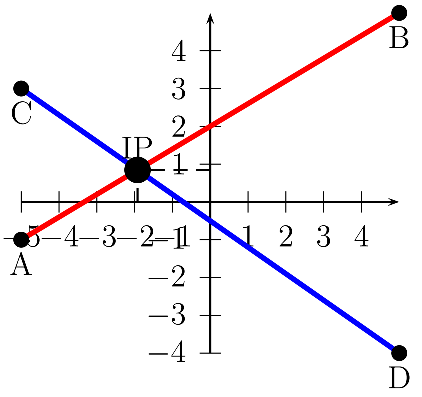


```
\begin{pspicture*}(-5,-4)(5,3.5)
  \psgrid[subgriddiv=0,griddots=5]
  \pnode(2,-2){FF}\qdisk(FF){1.5pt}
  \pnode(-5,5){A}\pnode(0,0){O}
  \multido{\nCountA=-2.4+0.4}{9}{%
    \psParallelLine[linecolor=red](O)(A)
      (0,\nCountA){9}{P1}
    \psline[linecolor=red](0,\nCountA)(
      FF)
    \psRelLine[linecolor=red](0,\nCountA
      )(FF){9}{P2}
  }
  \psline[linecolor=blue](A)(FF)
  \psRelLine[linecolor=blue](A)(FF){5}{
    END1}
  \psline[linewidth=2pt,arrows=->](2,0)(
    FF)
\end{pspicture*}
```

20 \psIntersectionPoint

This macro calculates the intersection point of two lines, given by the four coordinates. There is no special parameter here.

```
\psIntersectionPoint(<P0>)(<P1>)(<P2>)(<P3>){<node name>}
```

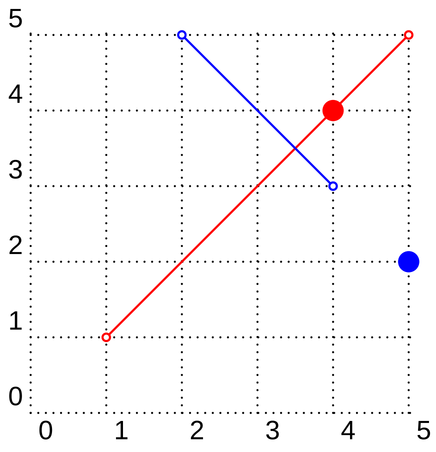


```
\psset{unit=0.5cm}
\begin{pspicture}(-5,-4)(5,5)
  \psaxes{->}(0,0)(-5,-4)(5,5)
  \psline[linecolor=red,linewidth=2pt](-5,-1)(5,5)
  \psline[linecolor=blue,linewidth=2pt](-5,3)(5,-4)
  \qdisk(-5,-1){3pt}\uput[-90](-5,-1){A}
  \qdisk(5,5){3pt}\uput[-90](5,5){B}
  \qdisk(-5,3){3pt}\uput[-90](-5,3){C}
  \qdisk(5,-4){3pt}\uput[-90](5,-4){D}
  \psIntersectionPoint(-5,-1)(5,5)(-5,3)(5,-4){IP}
  \qdisk(IP){5pt}\uput{0.3}[90](IP){IP}
  \psline[linestyle=dashed](IP|0,0)(IP)(0,0|IP)
\end{pspicture}
```

21 \setLNode and \setLCNode

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

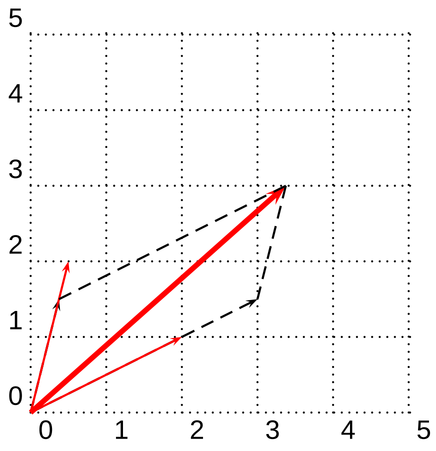
```
\setLNode(P1)(P2){value}{Node name}
```



```
\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}
```

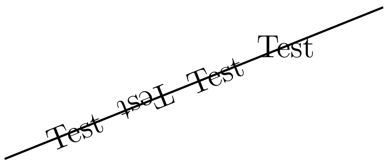


```
\begin{pspicture}(5,5)
\psgrid[subgriddiv=0,griddots=10]
\psset{linecolor=black}
\psline[linestyle=dashed]{->}(3,1.5)
\psline[linestyle=dashed]{->}(0.375,1.5)
\psset{linecolor=red}
\psline{->}(2,1)\psline{->}(0.5,2)
\setLCNode(2,1){1.5}(0.5,2){0.75}{PI}
\psline[linewidth=2pt]{->}(PI)
\psset{linecolor=black}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

22 \nlput

`\ncput` allows to set a label relative to the first node of the last node connection. With `\nlput` this can be done absolute to a given node. The syntax is different to the other node connection makros.

```
\nlput[options](A)(B){distance}{text}
```



```
\begin{pspicture}(5,2)
\pnode(0,0){A}
\pnode(5,2){B}
\ncline{A}{B}
\nlput[nrot=:U](A)(B){1cm}{Test}
\nlput[nrot=:D](A)(B){2cm}{Test}
\nlput[nrot=:U](A)(B){3cm}{Test}
\nlput(A)(B){4cm}{Test}
\end{pspicture}
```

Part III

pst-plot

23 New options

The option `tickstyle=full|top|bottom` is no more working in the `pstricks-add` package, because everything can be set by the `ticksize` option.

Table 2: All new parameters for pst-plot

Name	Type	Default
algebraic	false|true	false
comma	false|true	false
xAxis	false|true	true
yAxis	false|true	true
xyAxes	false|true	true
xDecimals	<number> or empty	{}
yDecimals	<number> or empty	{}
xyDecimals	<number> or empty	{}
ticks	<all|x|y|none>	all
labels	<all|x|y|none>	all
subticks	<number>	0
xsubticks	<number>	0
ysubticks	<number>	0
ticksize	<length [length]>	-4pt 4pt
subticksize	<number>	0.75
tickwidth	<length>	0.5\pslinewidth
subtickwidth	<length>	0.25\pslinewidth
tickcolor	<color>	black
xtickcolor	<color>	black
ytickcolor	<color>	black
subtickcolor	<color>	darkgray
xsubtickcolor	<color>	darkgray
ysubtickcolor	<color>	darkgray
ticklinestyle	solid | dashed | dotted | none	solid
subticklinestyle	solid | dashed | dotted | none	solid

Name	Type	Default
xlabelFactor	<anything>	{\ empty}
ylabelFactor	<anything>	{\ empty}
xlogBase	<number> or empty	{}
ylogBase	<number> or empty	{}
xylogBase	<number> or empty	{}
logLines	<none|x|y|all>	none
ignoreLines	<number>	0
nStep	<number>	1
nStart	<number>	0
nEnd	<number> or empty	{}
xStep	<number>	0
yStep	<number>	0
xStart	<number> or empty	{}
yStart	<number> or empty	{}
xEnd	<number> or empty	{}
yEnd	<number> or empty	{}
plotNo	<number>	1
plotNoMax	<number>	1
xAxisLabel	<anything>	{\ empty}
yAxisLabel	<anything>	{\ empty}
xAxisLabelPos	<(x,y)> or empty	{\ empty}
yAxisLabelPos	<(x,y)> or empty	{\ empty}
llx	<length>	0pt
lly	<length>	0pt
urx	<length>	0pt
ury	<length>	0pt
polarplot	false|true	false
trigLabels	false|true	false
ChangeOrder	false|true	false

23.1 `algebraic`[1]

By default the function of `\psplot` has to be described in Reversed Polish Notation. The option `algebraic` allows to do this in the common algebraic notation. E.g.:

RPN	algebraic
`x ln`	`ln(x)`
`x cos 2.71 x neg 10 div exp mul`	`cos(x)*2.71^(-x/10)`
`1 x div cos 4 mul`	`4*cos(1/x)`
`t cos t sin`	`cos(t)|sin(t)`

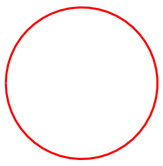
Setting the option `algebraic` to `true`, allow the user to describe all expression to be written in the classical algebraic notation (infix notation). The four arithmetic operarions are obviously defined `+-*/`, and also the exponential operator `^`. The natural priorities are used : $3+4\times5^5 = 3 + (4 \times (5^5))$, and by default the computation is done from left to right. The following functions are defined :

`sin`, `cos`, `tan`, `acos`, `asin`	in radians
`log`, `ln`	
`ceiling`, `floor`, `truncate`, `round`	
`sqrt`	square root
`abs`	absolute value
`fact`	for the factorial
`SUM`	for building sums
`IFTE`	for an easy case structure

These options can be used with **all** plot macros.

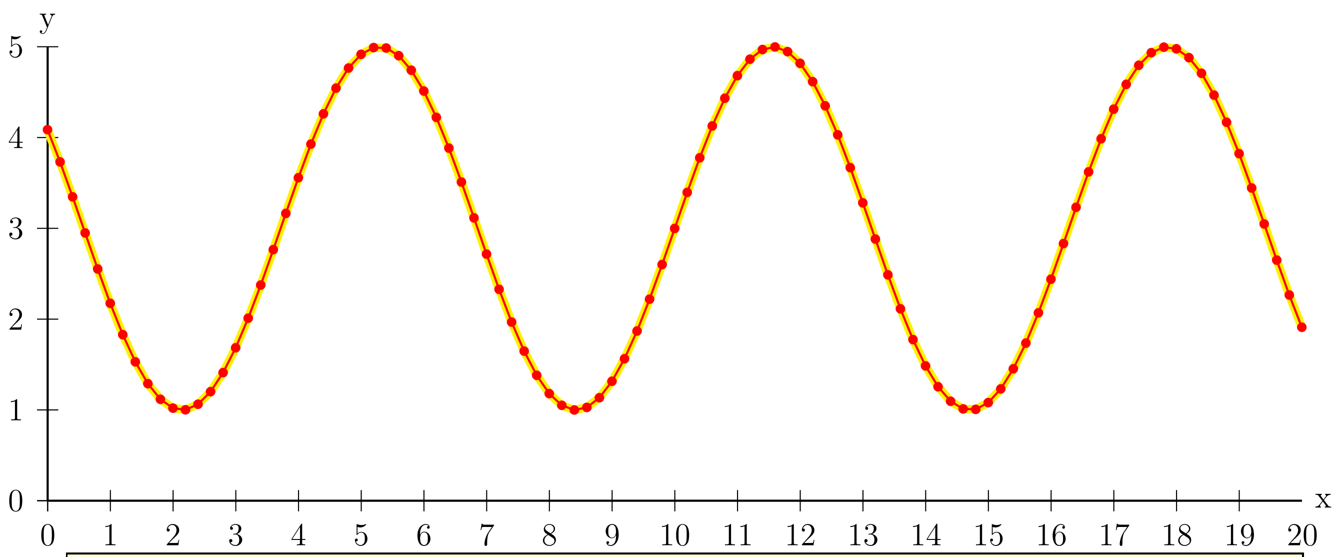
Using the option `algebraic` implies that all angles have to be used in the radian unit!

For the `\parametricplot` the two parts must be divided by the `|` character:

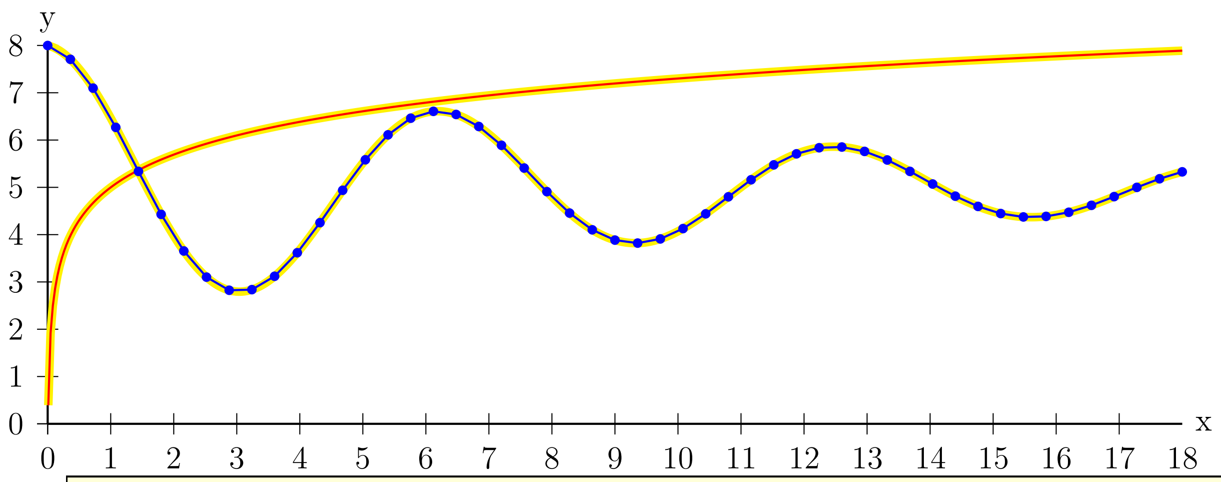


```
\begin{pspicture}(-0.5,-0.5)(0.5,0.5)
\parametricplot[algebraic,linecolor=red]{-3.14}{3.14}{cos(t)|sin(t)}
\end{pspicture}
```

[1]This part is adapted from the package `pst-eqdf`, written by Dominique Rodriguez.



```
\psset{lly=-0.5cm}
\psgraph(-10,-3)(10,2){\linewidth}{6cm}
  \psset{algebraic=true, plotpoints=101}
  \psplot[linecolor=yellow, linewidth=4\pslinewidth]{-10}{10}{2*sin(x)}%
  \psplot[linecolor=red, showpoints=true]{-10}{10}{2*sin(x)}
\endpsgraph
```

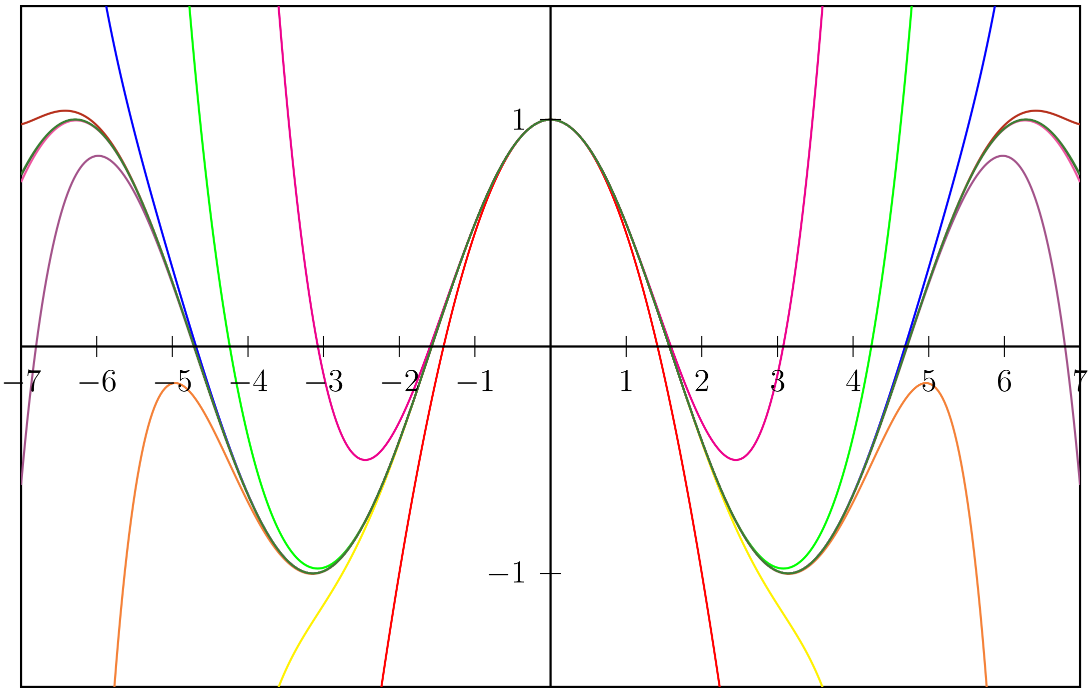


```
\psset{lly=-0.5cm}
\psgraph(0,-5)(18,3){15cm}{5cm}
  \psset{algebraic,plotpoints=501}
  \psplot[linecolor=yellow, linewidth=4\pslinewidth]{0.01}{18}{ln(x)}%
  \psplot[linecolor=red]{0.01}{18}{ln(x)}
  \psplot[linecolor=yellow,linewidth=4\pslinewidth]{0}{18}{3*cos(x)*2.71^(-x/10)}
  \psplot[linecolor=blue,showpoints=true,plotpoints=51]{0}{18}{3*cos(x)*2.71^(-x/10)}
\endpsgraph
```

23.1.1 Using the `SUM` function

Syntax: `SUM(<index name>,<start>,<step>,<end>,<function>)`

Let's plot the first development of cosine with polynomials: $\sum_{n=0}^{+\infty} \frac{(-1)^n x^{2n}}{n!}$.

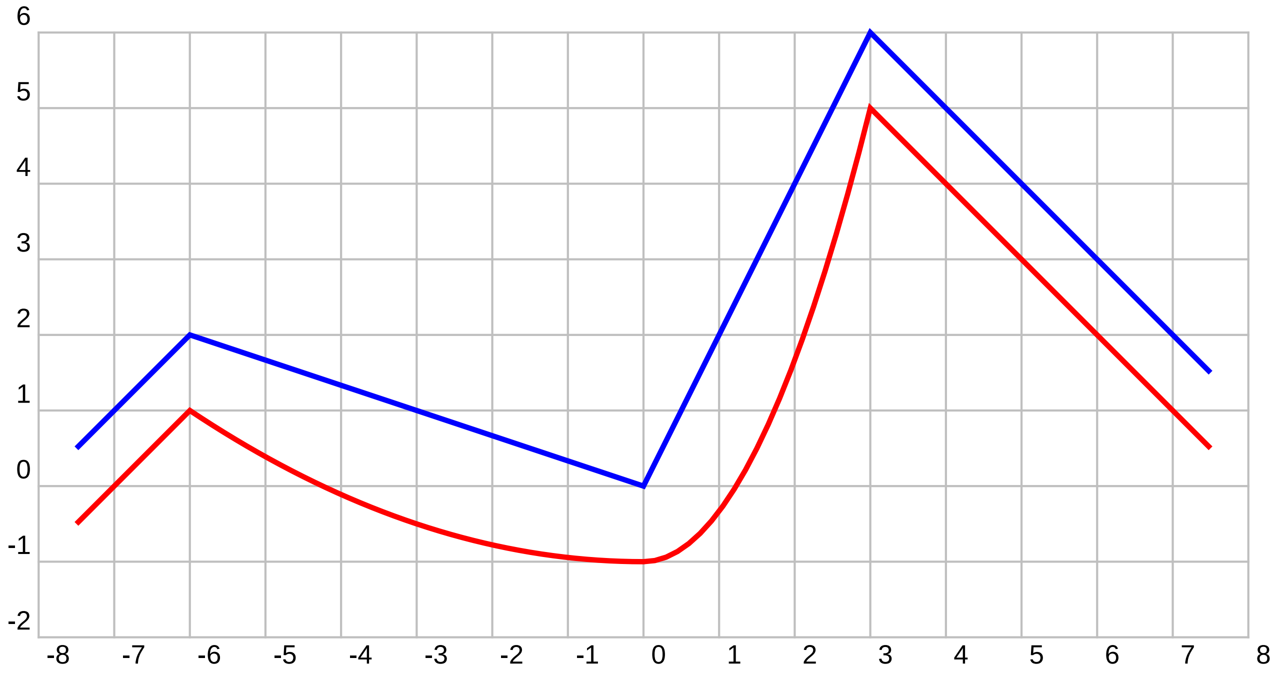


```
\psset{algebraic=true,plotpoints=501, yunit=3}
\def\getColor#1{\ifcase#1 black\or red\or magenta\or yellow\or green\or
  Orange\or blue\or
  DarkOrchid\or BrickRed\or Rhodamine\or OliveGreen\fi}
\begin{pspicture}(-7,-1.5)(7,1.5)
  \psclip{\psframe(-7,-1.5)(7,1.5)}
    \psplot{-7}{7}{cos(x)}
    \multido{\n=1+1}{10}{%
      \psplot[linecolor=\getColor{\n}]{-7}{7}{%
         SUM(ijk,0,1,\n,(-1)^ijk*x^(2*ijk)/fact(2*ijk))}}
   \endpsclip
  \psaxes(0,0)(-7,-1.5)(7,1.5)
\end{pspicture}
```

23.1.2 Using the `IFTE` function

Syntax: `IFTE(<condition>,<true part>,<false part>)`

Nesting of several `IFTE` are possible and seen in the following examples. A classical example is a piece wise linear function.



```
\begin{pspicture}(-7.5,-2.5)(7.5,6)\psgrid[subgriddiv=1,gridcolor=lightgray]
  \psset{algebraic=true, plotpoints=21,linewidth=2pt}
  \psplot[linecolor=blue]{-7.5}{7.5}{IFTE(x<-6,8+x,IFTE(x<0,-x/3,IFTE(x<3,2*x,9-x)))}
  \psplot[linecolor=red, plotpoints=101]{-7.5}{7.5}{%
      IFTE(2*x<-2^2*sqrt(9),7+x,IFTE(x<0,x^2/18-1,IFTE(x<3,2*x^2/3-1,8-x)))}%
\end{pspicture}
```

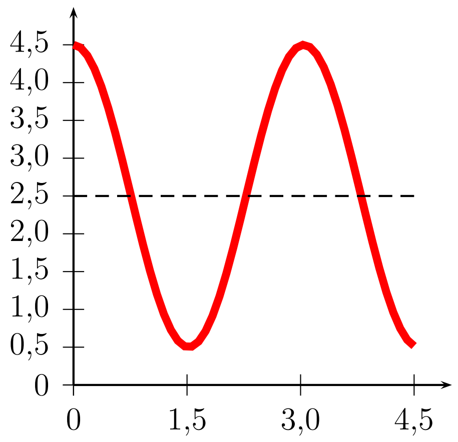
When you program a piece-wise defined function you must take care that a plotting point must be put on each point where the description changes. Use `showpoints=true` to see what's going on, when there is a problem. You are on the save side, when you choose a big number for `plotpoints`.

23.2 comma

Syntax:

```
comma=false|true
```

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.



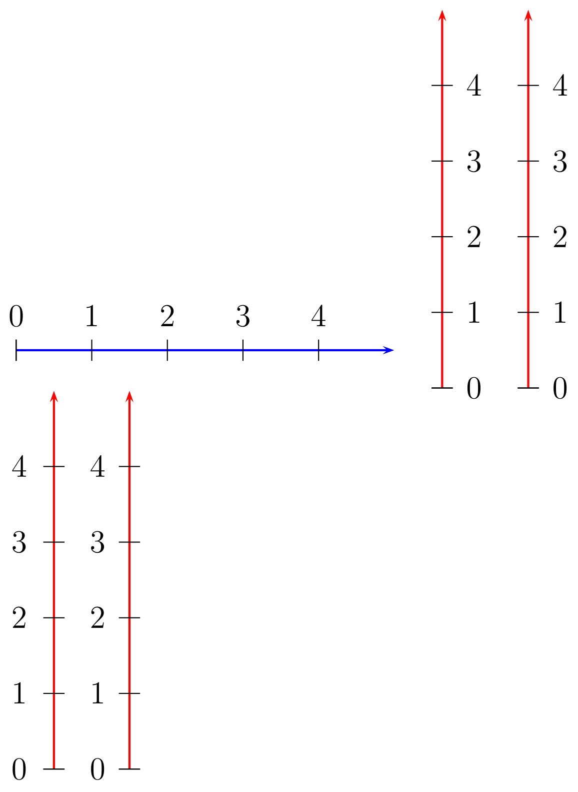
```
\begin{pspicture}(-0.5,-0.5)(5,5.5)
\psaxes[Dx=1.5,Dy=0.5,comma]{->}(5,5)
\psplot[linecolor=red,linewidth=3pt]{0}{4.5}%
   {x 180 mul 1.52 div cos 2 mul 2.5 add}
\psline[linestyle=dashed](0,2.5)(4.5,2.5)
\end{pspicture}
```

23.3 xyAxes, xAxis and yAxis

Syntax:

```
xyAxes=true|false
xAxis=true|false
yAxis=true|false
```

Sometimes there is only a need for one axis 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.



```
\begin{pspicture}(5,1)
\psaxes[yAxis=false,linecolor=blue
  ]{->}(0,0.5)(5,0.5)
\end{pspicture}
\begin{pspicture}(1,5)
\psaxes[xAxis=false,linecolor=red
  ]{->}(0.5,0)(0.5,5)
\end{pspicture}
\begin{pspicture}(1,5)
\psaxes[xAxis=false,linecolor=red
  ]{->}(0.5,0)(0.5,5)
\end{pspicture}\hspace{2em}
\begin{pspicture}(1,5)
\psaxes[xAxis=false,linecolor=red,
  labelsep=-20pt]{->}(0.5,0)(0.5,5)
\end{pspicture}%
\begin{pspicture}(1,5)
\psaxes[xAxis=false,linecolor=red
  ]{->}(0.5,0)(0.501,5)
\end{pspicture}%
```

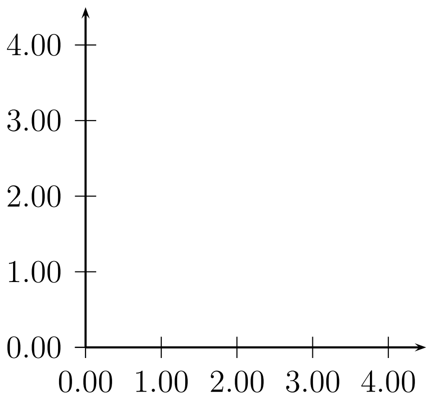
As seen in the example, a single y axis gets the labels on the right side. This can be changed in two ways, first with the option `labelsep` and second with a very short and therefore invisible x-axis (right example).

23.4 xyDecimals, xDecimals and yDecimals

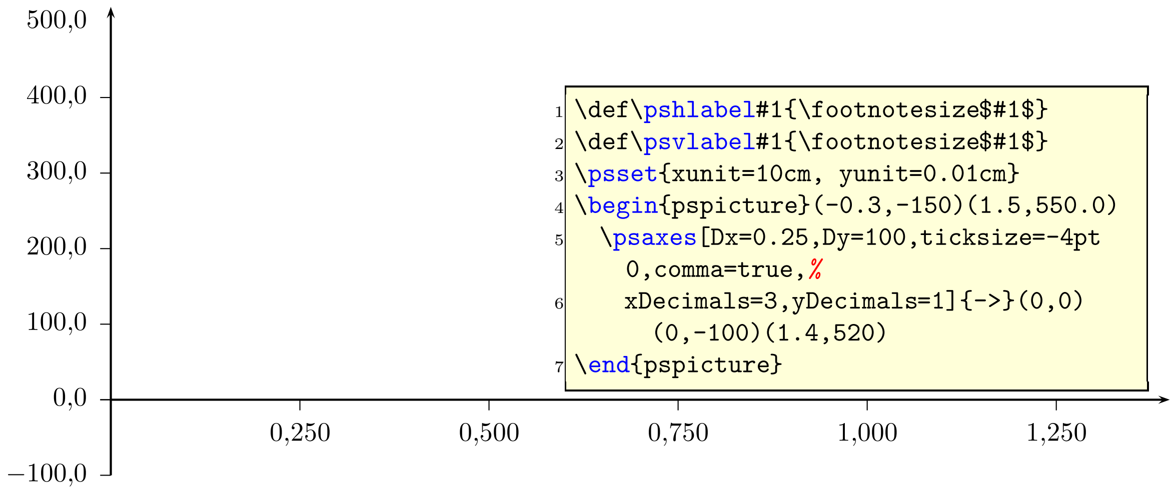
Syntax:

```
xyDecimals=<number>
xDecimals=<any>
yDecimals=<any>
```

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



```
\begin{pspicture}(-1.5,-0.5)(5,4.75)
  \psaxes[xyDecimals=2]{->}(0,0)(4.5,4.5)
\end{pspicture}
```

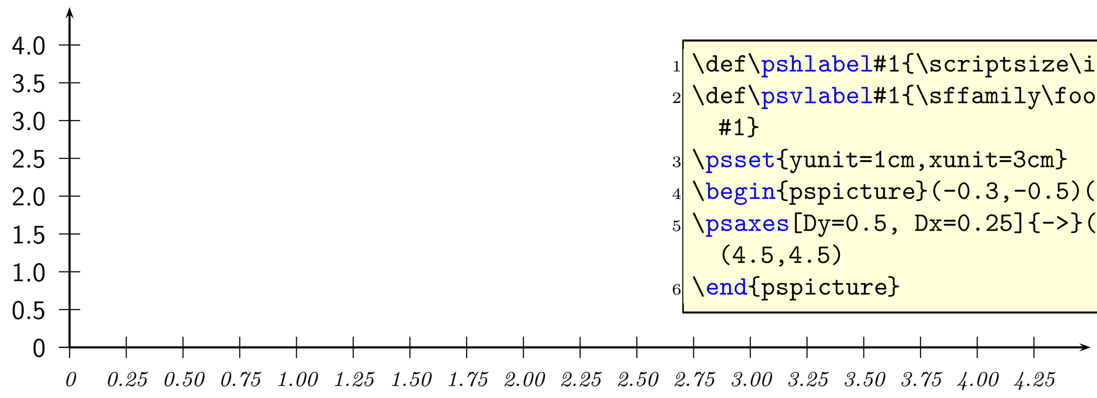


```
\def\pshlabel#1{\footnotesize$#1$}
\def\psvlabel#1{\footnotesize$#1$}
\psset{xunit=10cm, yunit=0.01cm}
\begin{pspicture}(-0.3,-150)(1.5,550.0)
  \psaxes[Dx=0.25,Dy=100,ticksize=-4pt
    0,comma=true,%
    xDecimals=3,yDecimals=1]{->}(0,0)
      (0,-100)(1.4,520)
\end{pspicture}
```

23.5 Changing the label style

There are no special keywords to change the labelstyle for the `\psaxes` macro. With a redefinition of the two macros `\pshlabel` and `\psvlabel` it is possible to set both axes in any shape. Like the default `pst-plot` package the coordinates are printed in mathmode, changing the fontsize to italic needs textmode.

```
\def\pshlabel#1{\scriptsize\itshape #1}
\def\psvlabel#1{\sffamily\footnotesize #1}
```



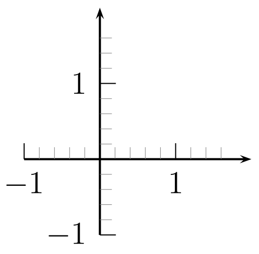
```
\def\pshlabel#1{\scriptsize\itshape #1}
\def\psvlabel#1{\sffamily\footnotesize
  #1}
\psset{yunit=1cm,xunit=3cm}
\begin{pspicture}(-0.3,-0.5)(5,4.75)
\psaxes[Dy=0.5, Dx=0.25]{->}(0,0)
  (4.5,4.5)
\end{pspicture}
```

23.6 ticks

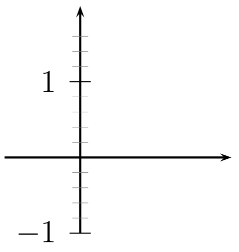
Syntax:

`ticks=all|x|y|none`

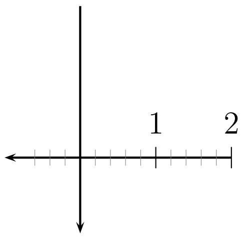
This option is also already in the `pst-plot` package and only mentioned here for some completeness.



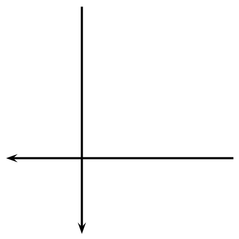
```
\psset{ticksize=6pt}
\begin{pspicture}(-1,-1)(2,2)
\psaxes[ticks=all,subticks=5]{->}(0,0)(-1,-1)(2,2)
\end{pspicture}
```



```
\begin{pspicture}(-1,-1)(2,2)
\psaxes[ticks=y,subticks=5]{->}(0,0)(-1,-1)(2,2)
\end{pspicture}
```



```
\begin{pspicture}(-1,-1)(2,2)
\psaxes[ticks=x,subticks=5]{->}(0,0)(2,2)(-1,-1)
\end{pspicture}
```



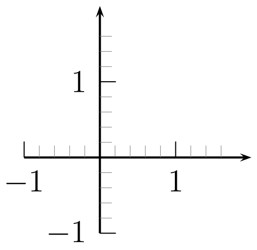
```
\begin{pspicture}(-1,-1)(2,2)
\psaxes[ticks=none,subticks=5]{->}(0,0)(2,2)(-1,-1)
\end{pspicture}
```

23.7 labels

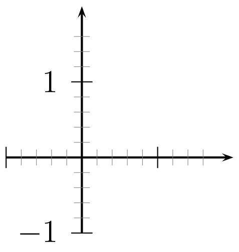
Syntax:

`labels=all|x|y|none`

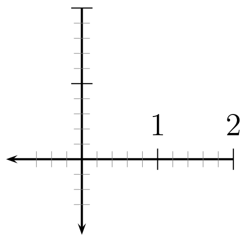
This option is also already in the `pst-plot` package and only mentioned here for some completeness.



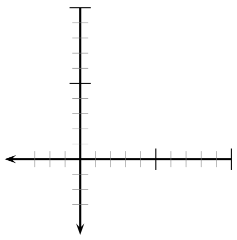
```
\psset{ticksize=6pt}
\begin{pspicture}(-1,-1)(2,2)
\psaxes[labels=all,subticks=5]{->}(0,0)(-1,-1)(2,2)
\end{pspicture}
```



```
\begin{pspicture}(-1,-1)(2,2)
\psaxes[labels=y,subticks=5]{->}(0,0)(-1,-1)(2,2)
\end{pspicture}
```



```
\begin{pspicture}(-1,-1)(2,2)
\psaxes[labels=x,subticks=5]{->}(0,0)(2,2)(-1,-1)
\end{pspicture}
```



```
\begin{pspicture}(-1,-1)(2,2)
\psaxes[labels=none,subticks=5]{->}(0,0)(2,2)(-1,-1)
\end{pspicture}
```

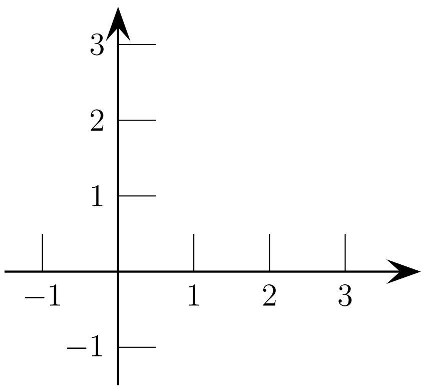
23.8 `ticksize`, `xticksize`, `yticksize`

With this new option the recent `tickstyle` option of `pst-plot` is obsolete and no more supported by `pstricks-add`.

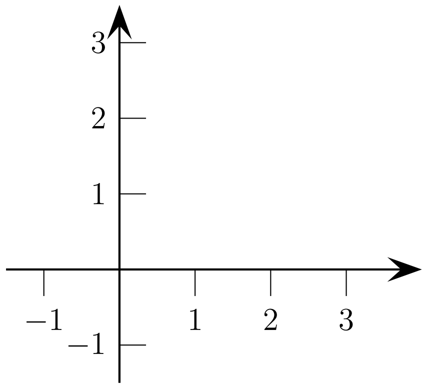
Syntax:

```
ticksize=value[unit]
ticksize=value[unit] value[unit]
xticksize=value[unit]
xticksize=value[unit] value[unit]
yticksize=value[unit]
yticksize=value[unit] value[unit]
```

`ticksize` sets both values. The first one is left/below and the optional second one is right/above of the coordinate axis. The old setting `tickstyle=bottom` is now easy to realize, e.g.: `ticksize=-6pt 0`, or vice versa, if the coordinates are set from positive to negative values.

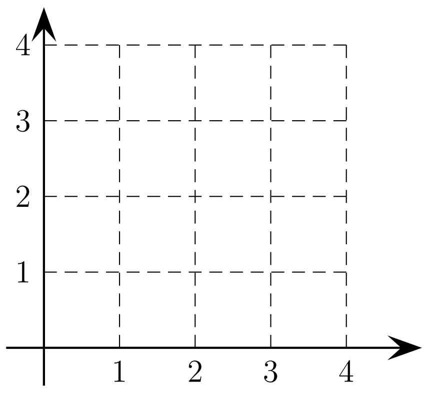


```
\psset{arrowscale=3}
\begin{pspicture}(-1.5,-1.5)(4,3.5)
  \psaxes[ticksize=0.5cm]{->}(0,0)(-1.5,-1.5)(4,3.5)
\end{pspicture}
```



```
\psset{arrowscale=3}
\begin{pspicture}(-1.5,-1.5)(4,3.5)
  \psaxes[xticksize=-10pt 0,yticksize=0 10pt
    ]{->}(0,0)(-1.5,-1.5)(4,3.5)
\end{pspicture}
```

A grid is also possible by setting the values to the max/min coordinates.



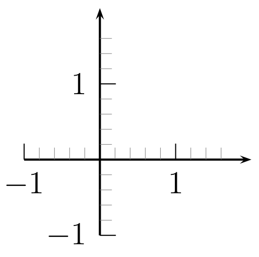
```
\psset{arrowscale=3}
\begin{pspicture}(-.5,-.5)(5,4.5)
  \psaxes[ticklinestyle=dashed,ticksize=0 4cm
    ]{->}(0,0)(-.5,-.5)(5,4.5)
\end{pspicture}
```

23.9 subticks

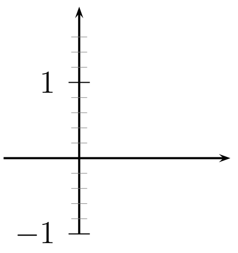
Syntax:

```
subticks=<number>
```

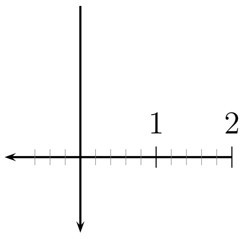
By default subticks cannot have labels.



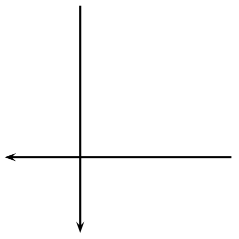
```
\psset{ticksize=6pt}
\begin{pspicture}(-1,-1)(2,2)
\psaxes[ticks=all,subticks=5]{->}(0,0)(-1,-1)(2,2)
\end{pspicture}
```



```
\begin{pspicture}(-1,-1)(2,2)
\psaxes[ticks=y,subticks=5]{->}(0,0)(-1,-1)(2,2)
\end{pspicture}
```



```
\begin{pspicture}(-1,-1)(2,2)
\psaxes[ticks=x,subticks=5]{->}(0,0)(2,2)(-1,-1)
\end{pspicture}
```



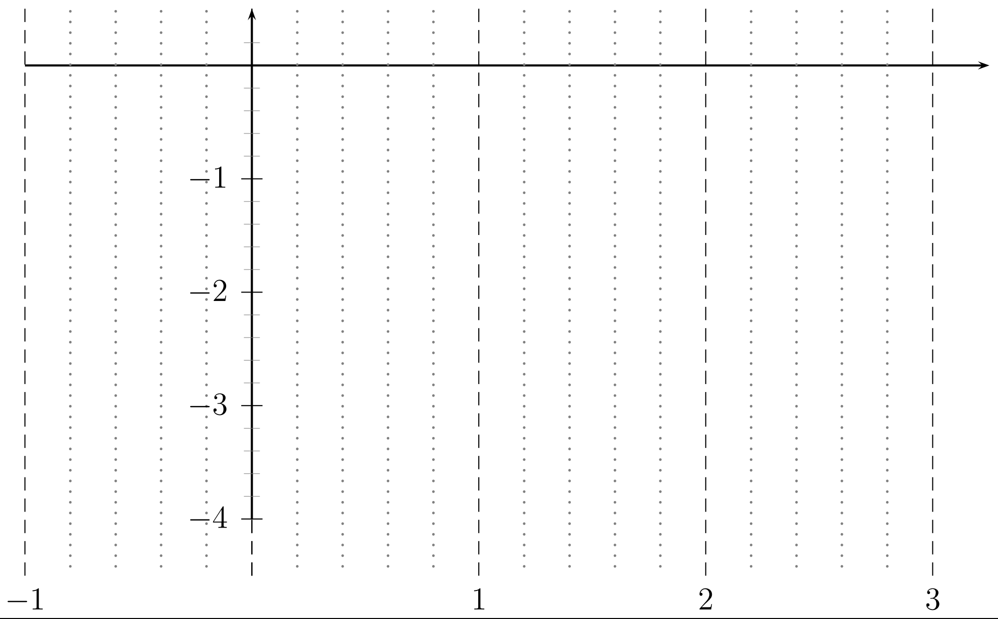
```
\begin{pspicture}(-1,-1)(2,2)
\psaxes[ticks=none,subticks=5]{->}(0,0)(2,2)(-1,-1)
\end{pspicture}
```

23.10 `subticksize`, `xsubticksize`, `ysubticksize`

Syntax:

```
subticksize=value
xsubticksize=value
ysubticksize=value
```

`subticksize` sets both values, which are relative to the ticksize length and can have any number. 1 sets it to the same length as the main ticks.



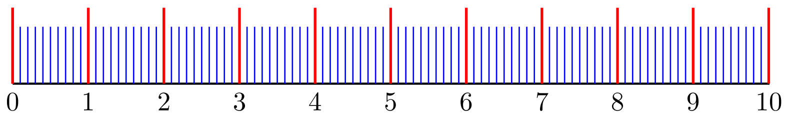
```
\psset{yunit=1.5cm,xunit=3cm}
\begin{pspicture}(-1.25,-4.75)(3.25,.75)
  \psaxes[xticksize=-4.5 0.5,ticklinestyle=dashed,subticks=5,xsubticksize=1,%
     ysubticksize=0.75,xsubticklinestyle=dotted,xsubtickwidth=1pt,
     subtickcolor=gray]{->}(0,0)(-1,-4)(3.25,0.5)
\end{pspicture}
```

23.11 `tickcolor`, `subtickcolor`

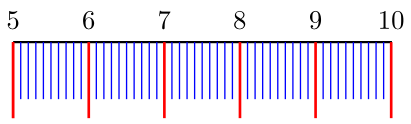
Syntax:

```
tickcolor=<color>
xtickcolor=<color>
ytickcolor=<color>
subtickcolor=<color>
xsubtickcolor=<color>
ysubtickcolor=<color>
```

`tickcolor` and `subtickcolor` set both for the x- and the y-Axis.



```
\def\pshlabel#1{\footnotesize$#1$}
\begin{pspicture}(0,-0.75)(10,1)
\psaxes[labelsep=2pt,yAxis=false,%
  labelsep=-10pt,ticksize=0 10mm,subticks=10,subticksize=0.75,%
  tickcolor=red,subtickcolor=blue,tickwidth=1pt,%
  subtickwidth=0.5pt](10.01,0)
\end{pspicture}
```



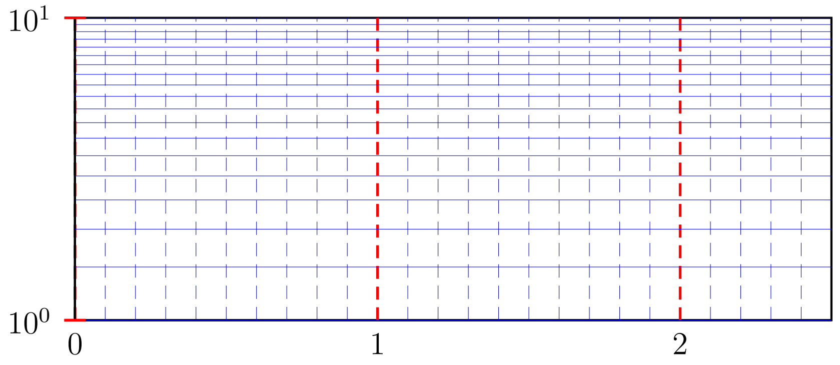
```
\def\pshlabel#1{\footnotesize$#1$}
\begin{pspicture}(5,-0.75)(10,1)
\psaxes[labelsep=2pt,yAxis=false,%
  labelsep=5pt,ticksize=0 -10mm,subticks=10,subticksize
    =0.75,%
  tickcolor=red,subtickcolor=blue,tickwidth=1pt,%
  subtickwidth=0.5pt,Ox=5](5,0)(5,0)(10.01,0)
\end{pspicture}
```

23.12 ticklinestyle and subticklinestyle

Syntax:

```
ticklinestyle=solid|dashed|dotted|none
xticklinestyle=solid|dashed|dotted|none
yticklinestyle=solid|dashed|dotted|none
subticklinestyle=solid|dashed|dotted|none
xsubticklinestyle=solid|dashed|dotted|none
ysubticklinestyle=solid|dashed|dotted|none
```

`ticklinestyle` and `subticklinestyle` set both values for the x and y axis. The value `none` doesn't really makes sense, because it is the same to `[sub]ticklines=0`

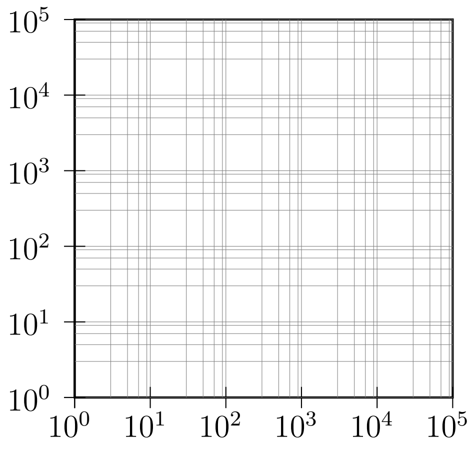


```
\psset{unit=4cm}
\pspicture(-0.15,-0.15)(2.5,1)
  \psaxes[axesstyle=frame,logLines=y,xticksize=0 1,xsubticksize=1,%
    ylogBase=10,tickcolor=red,subtickcolor=blue,tickwidth=1pt,%
    subticks=20,xsubticks=10,xticklinestyle=dashed,%
    xsubticklinestyle=dashed](2.5,1)
\endpspicture
```

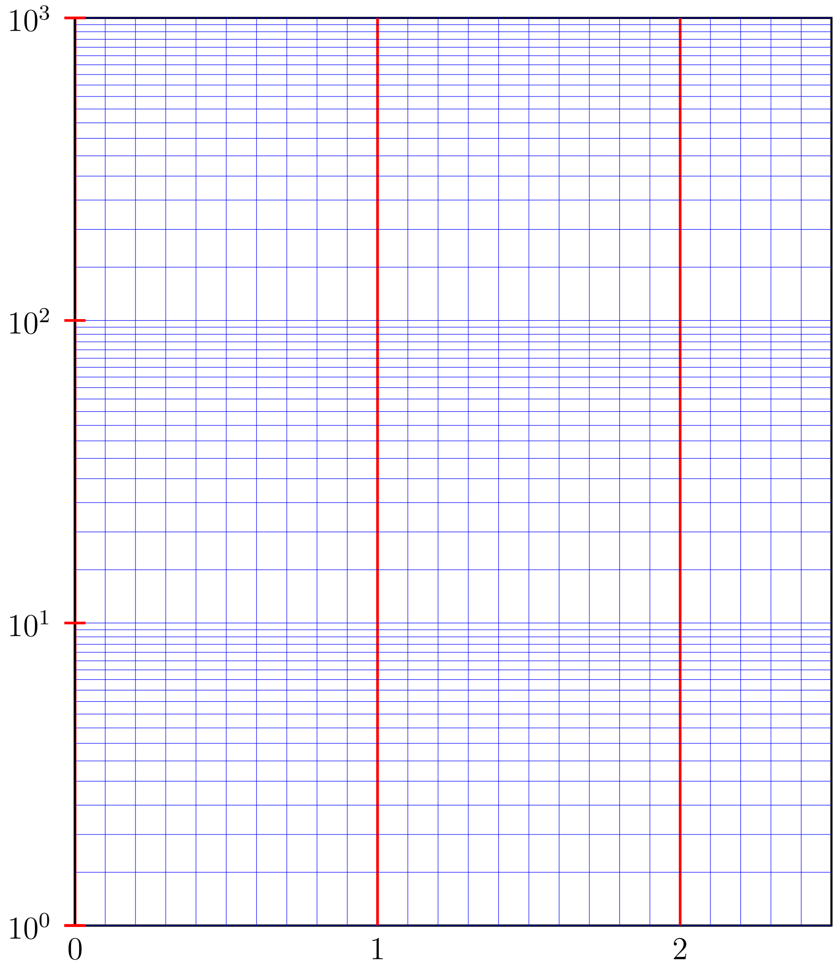
23.13 loglines

Syntax:

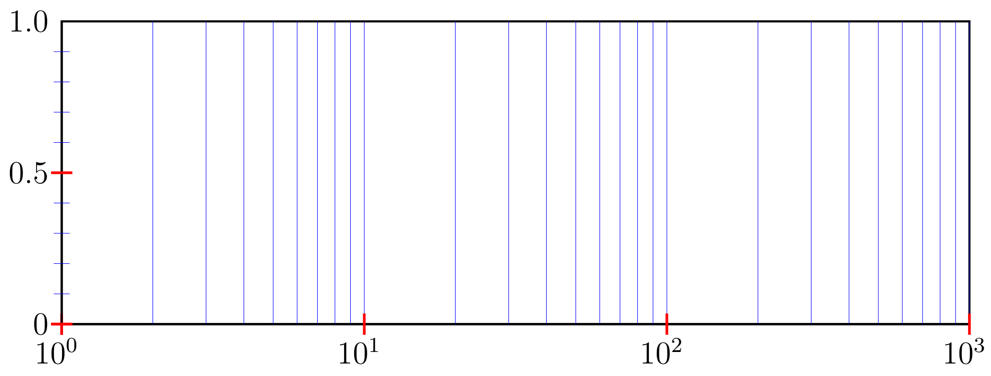
`loglines=all|x|y`



```
\pspicture(0,-1)(5,5)
   \psaxes[subticks=5,axesstyle=frame,xylogBase=10,
     logLines=all](5,5)
\endpspicture
```



```
\psset{unit=4cm}
\pspicture(-0.15,-0.15)(2.5,3)
  \psaxes[axesstyle=frame,logLines=y,xticksize=0 3,xsubticksize=1,%
    ylogBase=10,tickcolor=red,subtickcolor=blue,tickwidth=1pt,%
    subticks=20,xsubticks=10](2.5,3)
\endpspicture
```



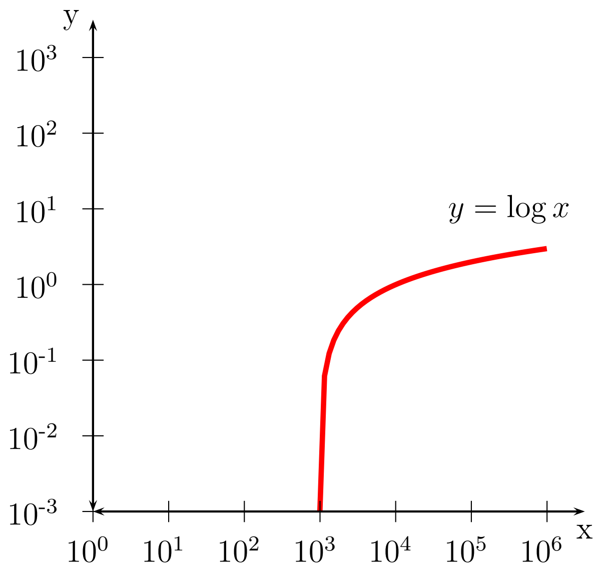
```
\psset{unit=4}
\pspicture(-0.5,-0.3)(3,1.2)
  \psaxes[axesstyle=frame,logLines=x,xlogBase=10,Dy=0.5,%
    tickcolor=red,subtickcolor=blue,tickwidth=1pt,ysubticks=5,xsubticks=10](3,1)
\endpspicture
```

23.14 xylogBase, xlogBase and ylogBase

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

23.14.1 xylogBase

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 the following example.



```
\begin{pspicture}(-3.5,-3.5)(3.5,3.5)
  \psplot[linewidth=2pt,linecolor=red
    ]{0.001}{3}{x log}
  \psaxes[xylogBase=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}
```

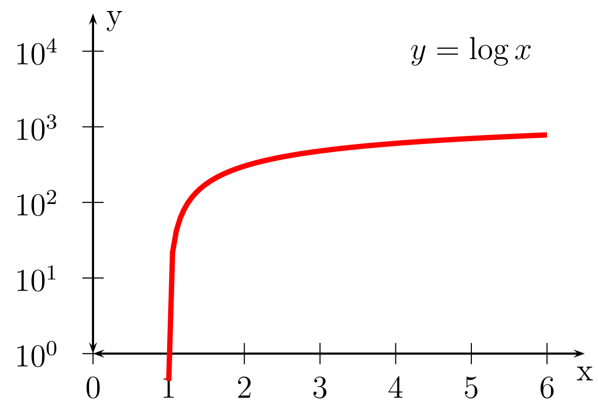
23.14.2 `ylogBase`

The values for the **psaxes** y-coordinate are now the exponents to the base 10 and for the right function to the base e: $10^{-3}\dots10^{1}$ which corresponds to the given y-intervall $-3\dots1.5$, 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

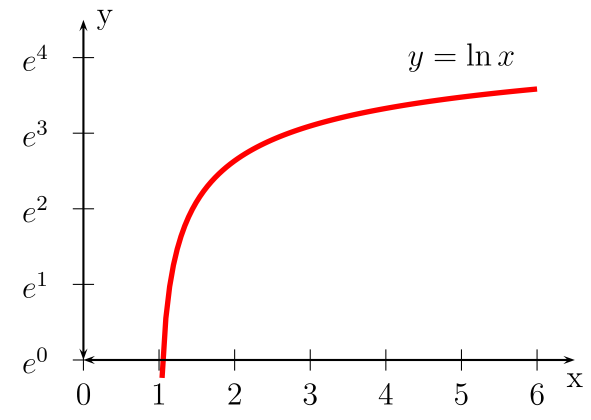
$$y = \log\{\log x\}$$

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

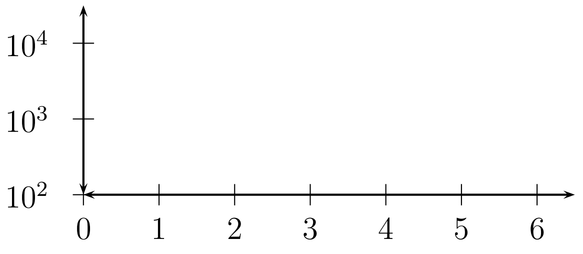
with an drawing intervall of $1.001\dots6$.



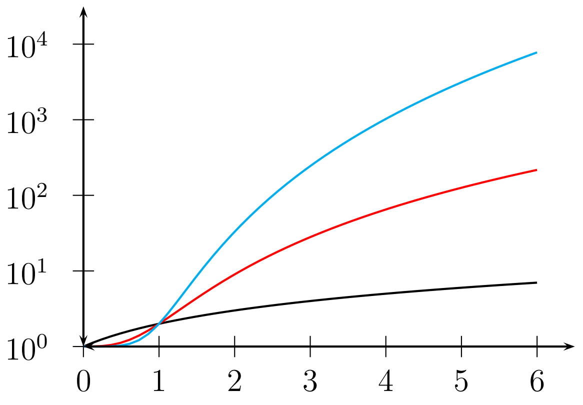
```
\begin{pspicture}(-0.5,-3.5)(6.5,1.5)
  \psaxes[ylogBase=10]{<->}(0,-3)(6.5,1.5)
  \uput[-90](6.5,-3){x}
  \uput[0](0,1.4){y}
  \rput(5,1){$y=\log x$}
  \psplot[linewidth=2pt,%
  plotpoints=100,linecolor=red]{1.001}{6}{x log
     log} % log(x)
\end{pspicture}
```



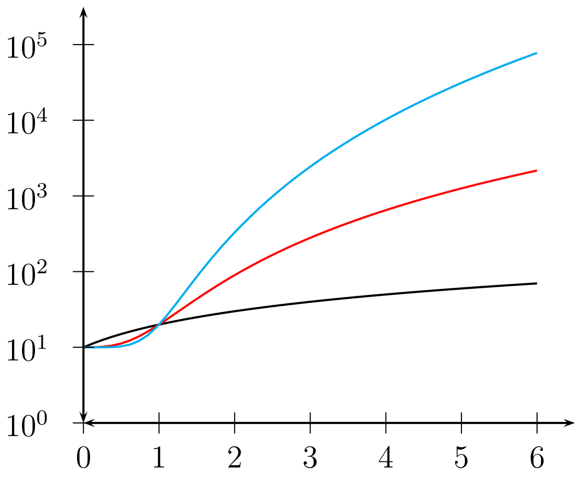
```
\begin{pspicture}(-0.5,-3.5)(6.5,1.5)
  \psplot[linewidth=2pt,plotpoints=100,
    linecolor=red]%
    {1.04}{6}{/ln {log 0.4343 div} def x ln ln}
       % log(x)
  \psaxes[ylogBase=e]{<->}(0,-3)(6.5,1.5)
  \uput[-90](6.5,-3){x}
  \uput[0](0,1.5){y}
  \rput(5,1){$y=\ln x$}
\end{pspicture}
```



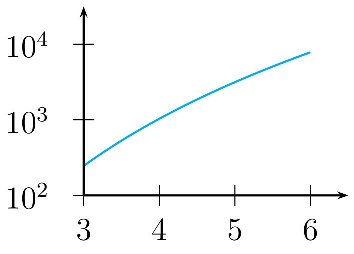
```
  \begin{pspicture}(-0.5,1.75)(6.5,4.5)
    \psaxes[ylogBase=10,Oy=2]{<->}(0,2)(0,2)
      (6.5,4.5)
  \end{pspicture}
```



```
  \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}
```



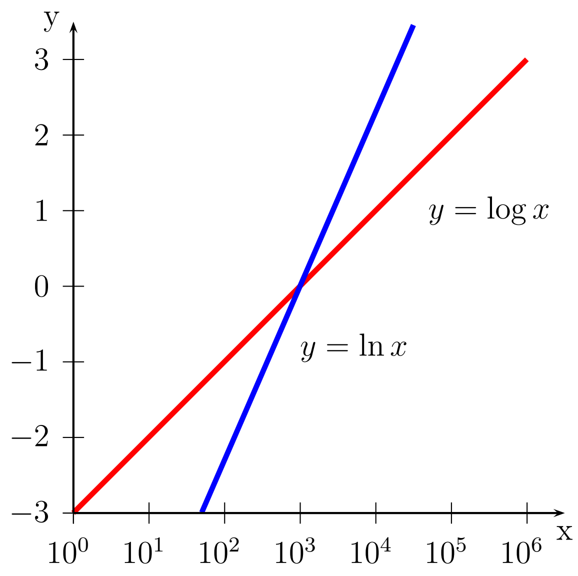
```
\begin{pspicture}(-0.5,-1.25)(6.5,4.5)
  \psplot{0}{6}{x x cos add log}
                              % x   + cox(x)
  \psplot[linecolor=red]{0}{6}{x 3 exp x cos
    add log} % x^3 + cos(x)
  \psplot[linecolor=cyan]{0}{6}{x 5 exp x cos
    add log} % x^5 + cos(x)
  \psaxes[ylogBase=10]{<->}(0,-1)(0,-1)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



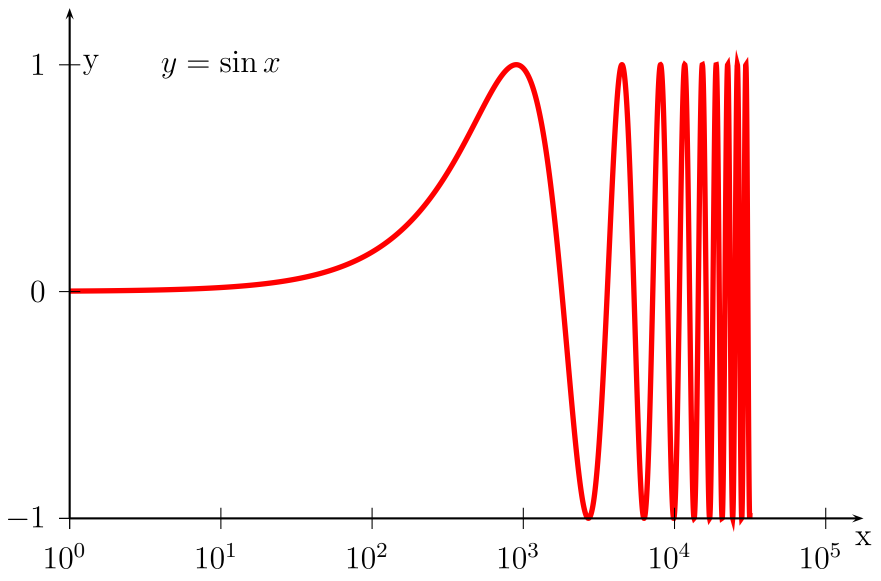
```
\begin{pspicture}(2.5,1.75)(6.5,4.5)
  \psplot[linecolor=cyan]{3}{6}{x 5 exp x cos add log} % x^5 +
    cos(x)
  \psaxes[ylogBase=10,Ox=3,Oy=2]{->}(3,2)(3,2)(6.5,4.5)
\end{pspicture}
```

23.14.3 xlogBase

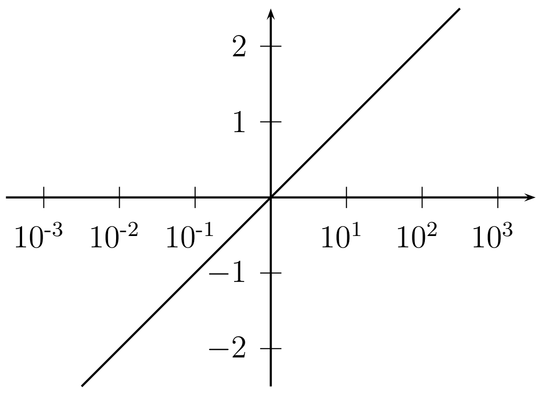
Now we have to use the easy math function $y = x$ because the x axis is still $\log x$.



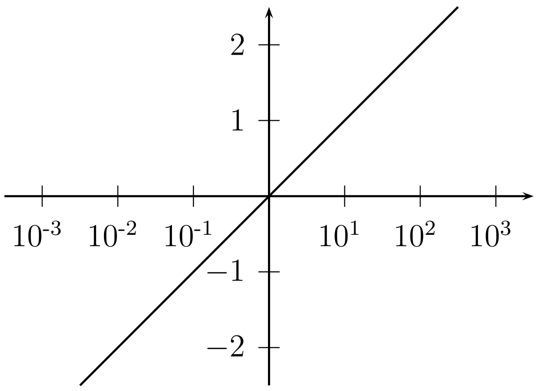
```
\begin{pspicture}(-3.5,-3.5)(3.5,3.5)
  \psplot[linewidth=2pt,linecolor=red]{-3}{3}{x
    } % log(x)
  \psplot[linewidth=2pt,linecolor=blue
    ]{-1.3}{1.5}{x 0.4343 div} % ln(x)
  \psaxes[xlogBase=10,Oy=-3]{->}(-3,-3)
    (3.5,3.5)
  \uput[-90](3.5,-3){x}
  \uput[180](-3,3.5){y}
  \rput(2.5,1){$y=\log x$}
  \rput[lb](0,-1){$y=\ln x$}
\end{pspicture}
```



```
\psset{yunit=3cm,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 }
  \psaxes[xlogBase=10,Oy=-1]{->}(-1,-1)(4.25,1.25)
\end{pspicture}
```



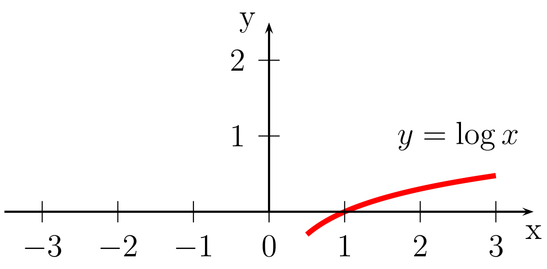
```
\begin{pspicture}(-3.5,-2.5)(3.5,2.5)
  \psaxes[xlogBase=10]{->}(0,0)(-3.5,-2.5)
    (3.5,2.5)
  \psplot{-2.5}{2.5}{10 x exp log}
\end{pspicture}
```



```
\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}
```

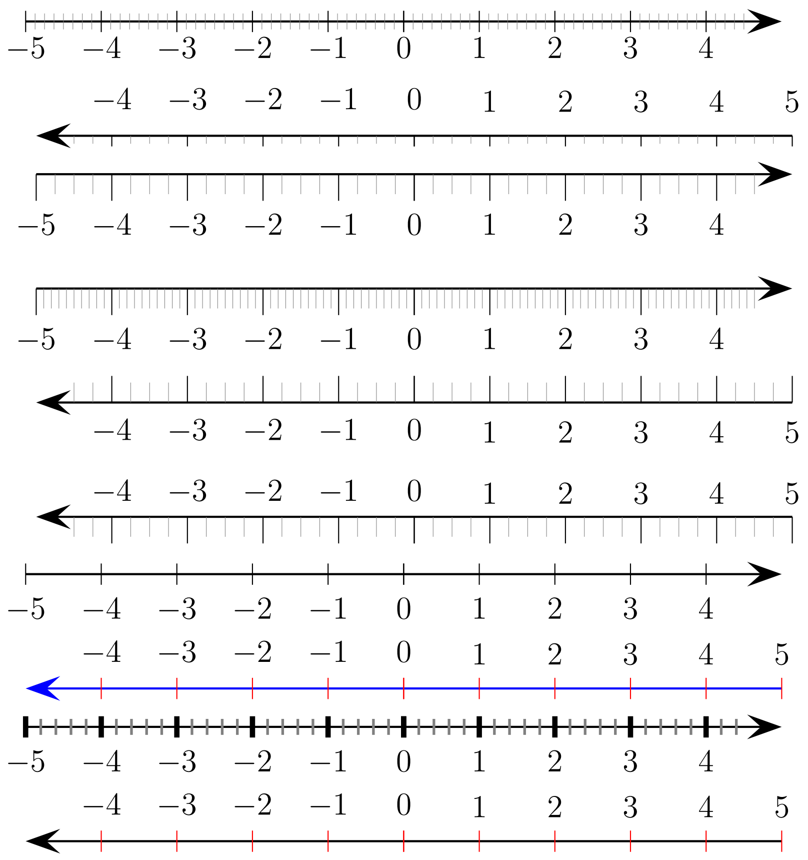
23.14.4 No logstyle (`xylogBase={}`)

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

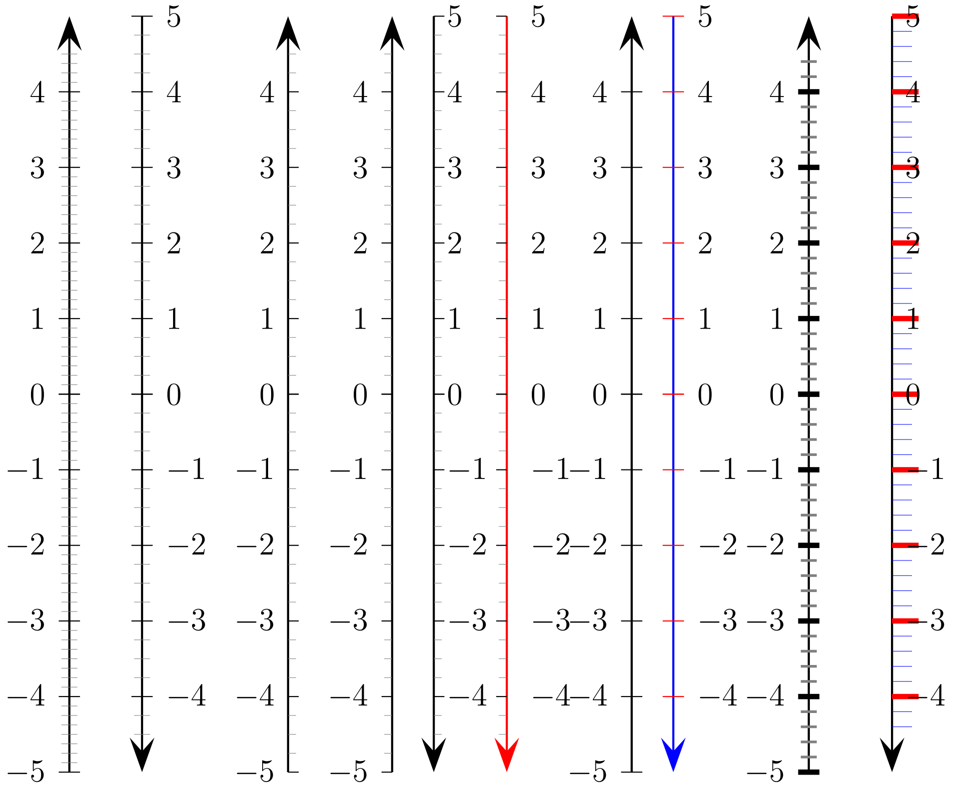


```
\begin{pspicture}(-3.5,-0.5)(3.5,2.5)
  \psplot[linewidth=2pt,linecolor=red,xylogBase
    ={}]{0.5}{3}{x log} % log(x)
  \psaxes{->}(0,0)(-3.5,0)(3.5,2.5)
  \uput[-90](3.5,0){x}
  \uput[180](0,2.5){y}
  \rput(2.5,1){$y=\log x$}
\end{pspicture}
```

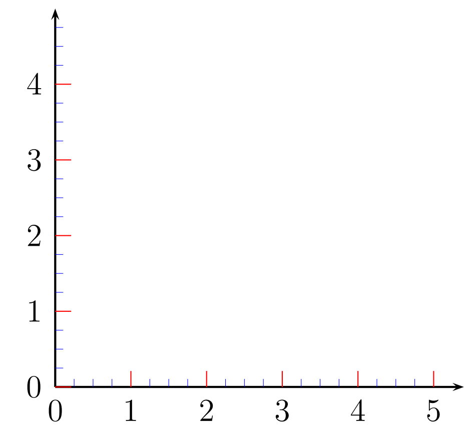
23.15 subticks, tickwidth and subtickwidth



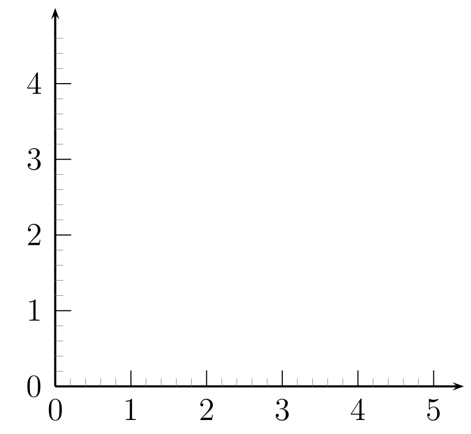
```
\psset{arrowscale=3}
\psaxes[labelsep=2pt,yAxis=false,subticks=8]{->}(0,0)(-5,-1)(5,1)\\[1cm]
\psaxes[yAxis=false,subticks=4,ticksize=-4pt 0]{->}(0,0)(5,1)(-5,-1)\\
\psaxes[yAxis=false,subticks=4,ticksize=-10pt 0]{->}(0,0)(-5,-5)(5,5)\\[1cm]
\psaxes[yAxis=false,subticks=10,ticksize=0 -10pt,labelsep=15pt]{->}(0,0)(-5,-5)
  (5,5)\\[1cm]
\psaxes[yAxis=false,subticks=4,ticksize=0 10pt,labelsep=-15pt]{->}(0,0)(5,5)
  (-5,-5)\\[1cm]
\psaxes[yAxis=false,subticks=4,ticksize=0 -10pt]{->}(0,0)(5,5)(-5,-5)\\[0.25cm]
\psaxes[yAxis=false,subticks=0]{->}(0,0)(-5,-5)(5,5)\\[1cm]
\psaxes[yAxis=false,subticks=0,tickcolor=red,linecolor=blue]{->}(0,0)(5,5)(-5,-5)
  \\
\psaxes[yAxis=false,subticks=5,tickwidth=2pt,subtickwidth=1pt]{->}(0,0)(-5,-5)
  (5,5)\\[1cm]
\psaxes[yAxis=false,subticks=0,tickcolor=red]{->}(0,0)(5,5)(-5,-5)
```



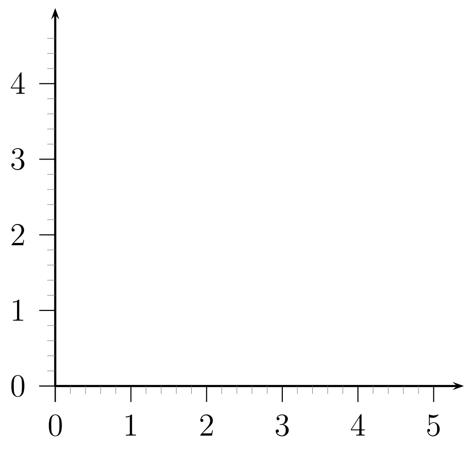
```
\psset{arrowscale=3}
\psaxes[xAxis=false,subticks=8]{->}(0,0)(-5,-5)(5,5)\hspace{2em}
\psaxes[xAxis=false,subticks=4]{->}(0,0)(5,5)(-5,-5)\hspace{4em}
\psaxes[xAxis=false,subticks=4,ticksize=0 4pt]{->}(0,0)(-5,-5)(5,5)\hspace{3em}
\psaxes[xAxis=false,subticks=4,ticksize=-4pt 0]{->}(0,0)(-5,-5)(5,5)\hspace{1em}
\psaxes[xAxis=false,subticks=4,ticksize=0 4pt]{->}(0,0)(5,5)(-5,-5)\hspace{2em}
\psaxes[xAxis=false,subticks=4,ticksize=-4pt 0,linecolor=red]{->}(0,0)(5,5)
  (-5,-5)\hspace{4em}
\psaxes[xAxis=false,subticks=0]{->}(0,0)(-5,-5)(5,5)\hspace{1em}
\psaxes[xAxis=false,subticks=0,tickcolor=red,linecolor=blue]{->}(0,0)(5,5)(-5,-5)
  \hspace{4em}
\psaxes[xAxis=false,subticks=5,tickwidth=2pt,subtickwidth=1pt]{->}(0,0)(-5,-5)
  (5,5)\hspace{2em}
\psaxes[xAxis=false,subticks=5,tickcolor=red,tickwidth=2pt,%
  ticksize=10pt,subtickcolor=blue,subticksize=0.75]{->}(0,0)(5,5)(-5,-5)
```



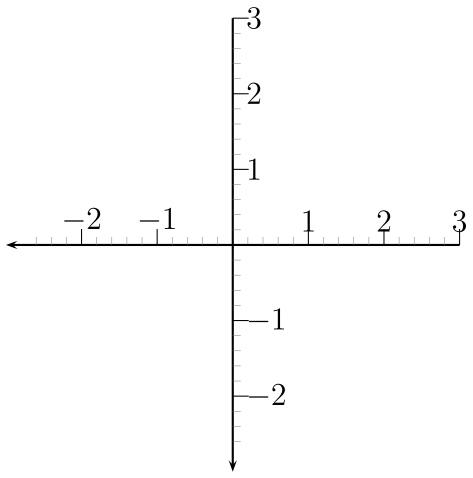
```
\pspicture(5,5.5)
\psaxes[subticks=4,ticksize=6pt,subticksize=0.5,%
  tickcolor=red,subtickcolor=blue]{->}(5.4,5)
\endpspicture
```



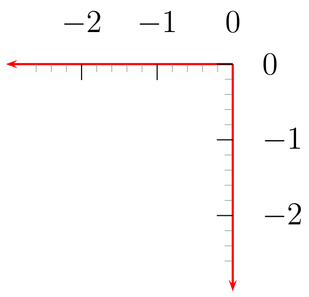
```
\pspicture(5,5.5)
  \psaxes[subticks=5,ticksize=0 6pt,subticksize
    =0.5]{->}(5.4,5)
\endpspicture
```



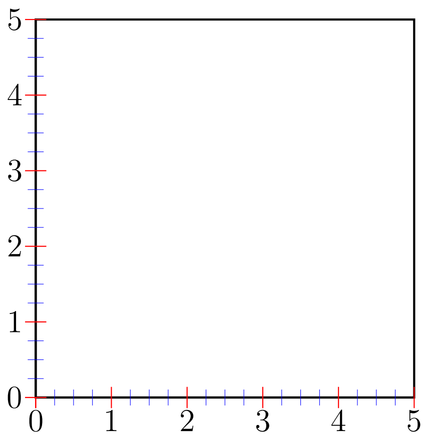
```
\pspicture(5,5.5)
  \psaxes[subticks=5,ticksize=-6pt 0,subticksize
    =0.5]{->}(5.4,5)
\endpspicture
```



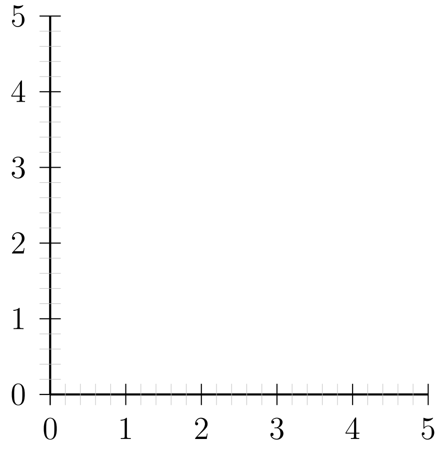
```
\pspicture(-3,-3)(3,3.5)
   \psaxes[subticks=5,ticksize=0 6pt,subticksize
     =0.5]{->}(0,0)(3,3)(-3,-3)
\endpspicture
```



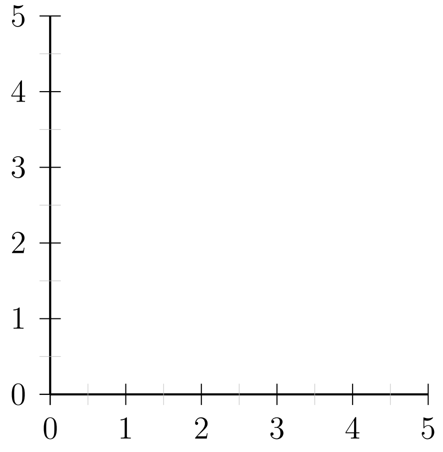
```
\pspicture(0,0.5)(-3,-3)
   \psaxes[subticks=5,ticksize=-6pt 0,subticksize
     =0.5,linecolor=red]{->}(-3,-3)
\endpspicture
```



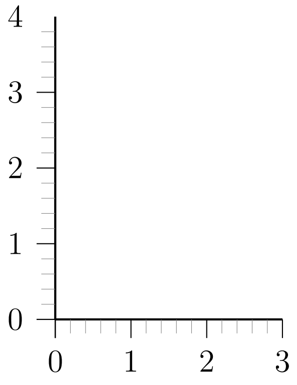
```
\psset{axesstyle=frame}
\pspicture(5,5.5)
   \psaxes[subticks=4,tickcolor=red,subtickcolor=blue
     ](5,5)
\endpspicture
```



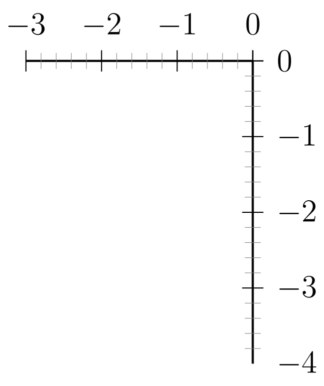
```
\pspicture(5,5.5)
  \psaxes[subticks=5,subticksize=1,subtickcolor=
    lightgray](5,5)
\endpspicture
```



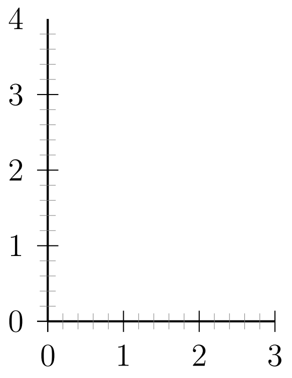
```
\pspicture(5,5.5)
  \psaxes[subticks=2,subticksize=1,subtickcolor=
    lightgray](5,5)
\endpspicture
```



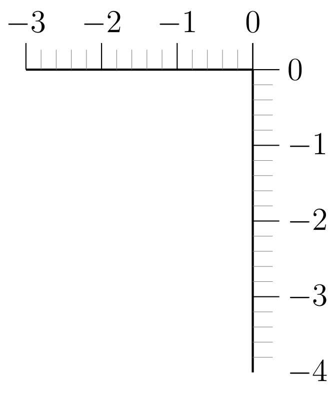
```
\pspicture(3,4.5)
  \psaxes[subticks=5,ticksize=-7pt 0](3,4)
\endpspicture
```



```
\pspicture(0,1)(-3,-4)
   \psaxes[subticks=5](-3,-4)
\endpspicture
```



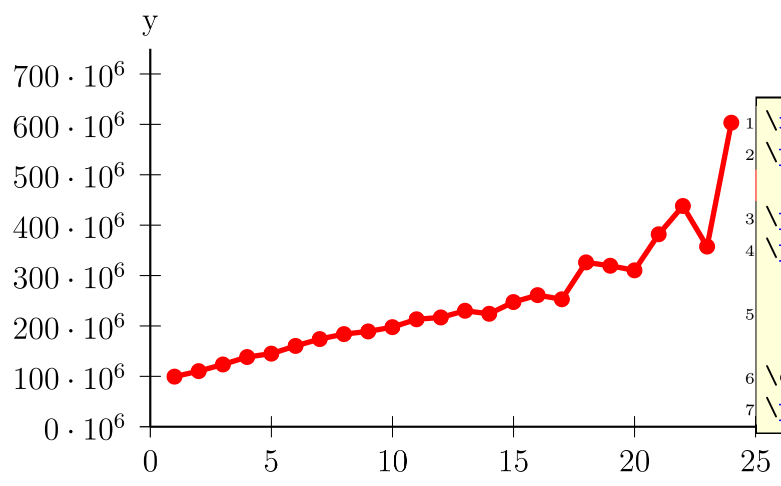
```
\pspicture(3,4.5)
   \psaxes[axesstyle=axes,subticks=5](3,4)
\endpspicture
```



```
\pspicture(0,1)(-3,-4)
  \psaxes[axesstyle=axes,subticks=5,%
    ticksize=0 10pt,labelsep=13pt](-3,-4)
\endpspicture
```

23.16 `xlabelFactor` and `ylabelFactor`

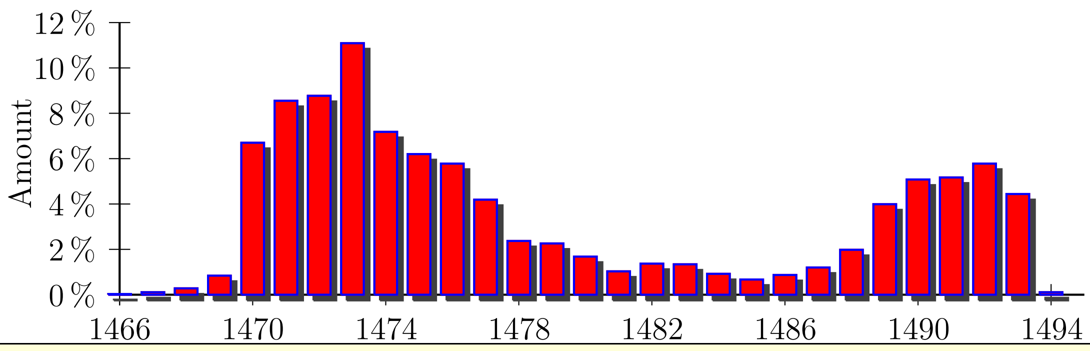
When having big numbers as data records then it makes sense to write the values as $< number > \cdot 10^{<exp>}$ These new options allow to define the additional part of the value.



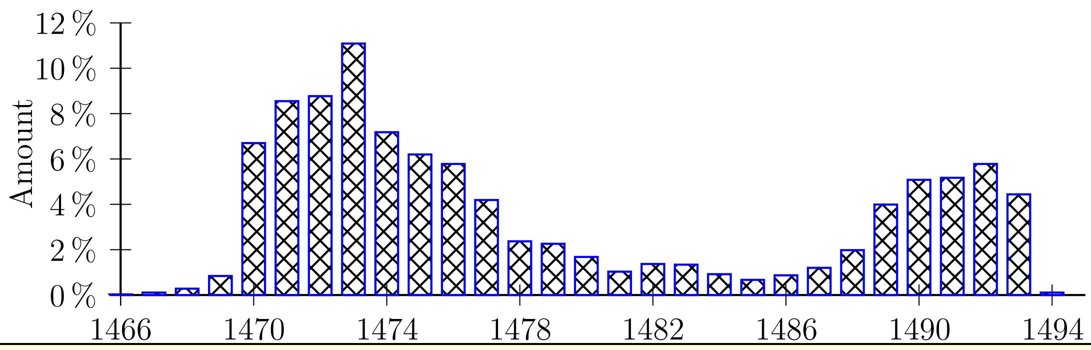
```
\readdata{\data}{demo1.dat}
\pstScalePoints(1,0.000001){}{}% (x,y){
  additional x operator}{y op}
\psset{llx=-1cm,lly=-1cm}
\psgraph[ylabelFactor={\cdot 10^6},Dx=5,
  Dy=100](0,0)(25,750){8cm}{5cm}
   \listplot[linecolor=red, linewidth=2
     pt, showpoints=true]{\data}
\endpsgraph
\pstScalePoints(1,1){}{}% reset
```

23.17 Plot style bar and option barwidth

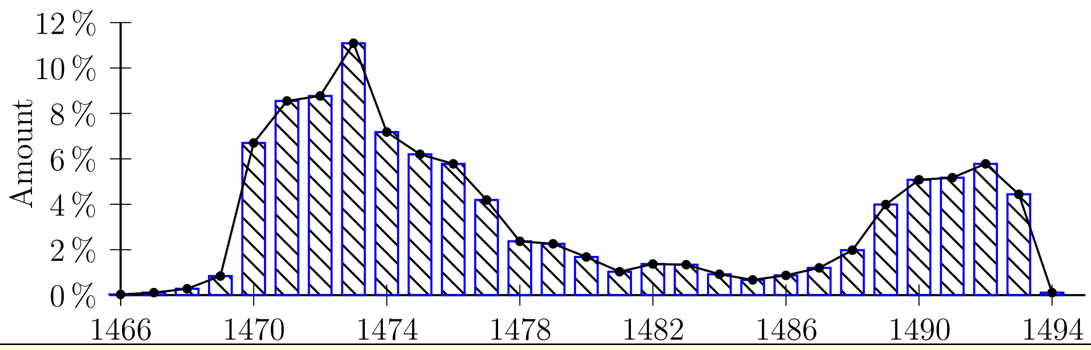
This option allows to draw bars for the data records. The width of the bars is controlled by the option `barwidth`, which is set by default to value of `0.25cm`, which is the total width.



```
\psset{xunit=.44cm,yunit=.3cm}
\begin{pspicture}(-2,-1.5)(29,13)
  \psaxes[axesstyle=axes,Ox=1466,Oy=0,Dx=4,Dy=2,%
     ylabelFactor={\,\%}]{-}(29,12)
  \listplot[shadow=true,linecolor=blue,plotstyle=bar,barwidth=0.3cm,
     fillcolor=red,fillstyle=solid]{\barData}
  \rput{90}(-3,6.25){Amount}
\end{pspicture}
```



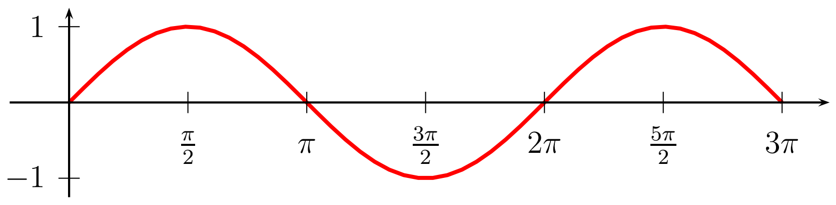
```
\psset{xunit=.44cm,yunit=.3cm}
\begin{pspicture}(-2,-1.5)(29,13)
  \psaxes[axesstyle=axes,Ox=1466,Oy=0,Dx=4,Dy=2,%
     ylabelFactor={\,\%}]{-}(29,12)
  \listplot[linecolor=blue,plotstyle=bar,barwidth=0.3cm,
     fillcolor=red,fillstyle=crosshatch]{\barData}
  \rput{90}(-3,6.25){Amount}
\end{pspicture}
```



```
\psset{xunit=.44cm,yunit=.3cm}
\begin{pspicture}(-2,-1.5)(29,13)
  \psaxes[axesstyle=axes,Ox=1466,Oy=0,Dx=4,Dy=2,%
     ylabelFactor={\,\%}]{-}(29,12)
  \listplot[linecolor=blue,plotstyle=bar,barwidth=0.3cm,
     fillcolor=red,fillstyle=vlines]{\barData}
  \listplot[showpoints=true]{\barData}
  \rput{90}(-3,6.25){Amount}
\end{pspicture}
```

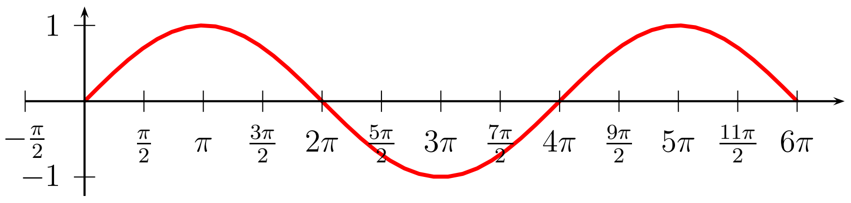
23.18 Axis with trigonmetrical units

With the option `trigLabels=true` the labels on the x axis are trigonometrical ones:

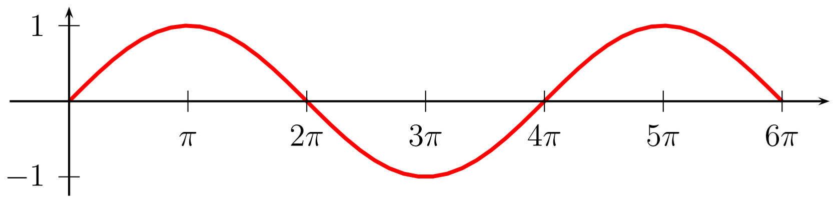


```latex
\begin{pspicture}(-0.5,-1.25)(10,1.25)
  \psplot[linecolor=red,linewidth=1.5pt]%
     {0}{9.424777961}{x 180 mul 3.141592654 div sin}
  \psaxes[xunit=1.570796327,showorigin=false,trigLabels]{->}(0,0)(-0.5,-1.25)(6.4,1.25)
\end{pspicture}
```

With the value of `xunit` one can change the labels.



```latex
\begin{pspicture}(-0.5,-1.25)(10,1.25)
  \psplot[linecolor=red,linewidth=1.5pt]%
     {0}{9.424777961}{x 180 mul 3.141592654 div sin}
  \psaxes[xunit=0.7853981635,showorigin=false,trigLabels]{->}(0,0)(-1,-1.25)(12.8,1.25)
\end{pspicture}
```



```latex
\begin{pspicture}(-0.5,-1.25)(10,1.25)
  \psplot[linecolor=red,linewidth=1.5pt]%
     {0}{9.424777961}{x 180 mul 3.141592654 div sin}
  \psaxes[xunit=0.7853981635,showorigin=false,trigLabels,Dx=2]{->}(0,0)(-1,-1.25)
    (12.8,1.25)
\end{pspicture}
```

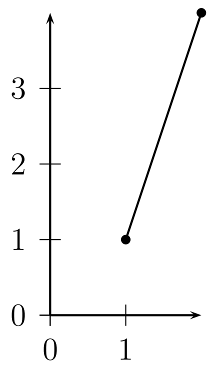
23.19 New options for \readdata

By default the macros `\readdata` reads every data record, which could be annoying when you have some text lines at top of your data files or when there are more than 10000 records to read.

`pstricks-add` defines two additional keys `ignoreLines` and `nStep`, which allows to ignore preceeding lines, e.g. `ignoreLines=2`, or to read only a selected part of the data records, e.g. `nStep=10`, only every 10th records is saved.

```
\readdata[ignoreLines=2]{\dataA}{stressrawdata.dat}
\readdata[nStep=10]{\dataA}{stressrawdata.dat}
```

The default value for `ignoreLines` is 0 and for `nStep` is 1. the following data file has two text lines which shall be ignored by the `\readdata` macro:



```
\begin{filecontents*}{pstricks-add-data9.dat}
some nonsense in this line ï¿½ï¿½ï¿½time forcex forcey
0 0.2
1 1
2 4
\end{filecontents*}
\readdata[ignoreLines=2]{\data}{pstricks-add-data9.dat}
\pspicture(2,4)
  \listplot[showpoints=true]{\data}
  \psaxes{->}(2,4)
\endpspicture
```

23.20 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`
`ChangeOrder`	`false`

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 [4]):

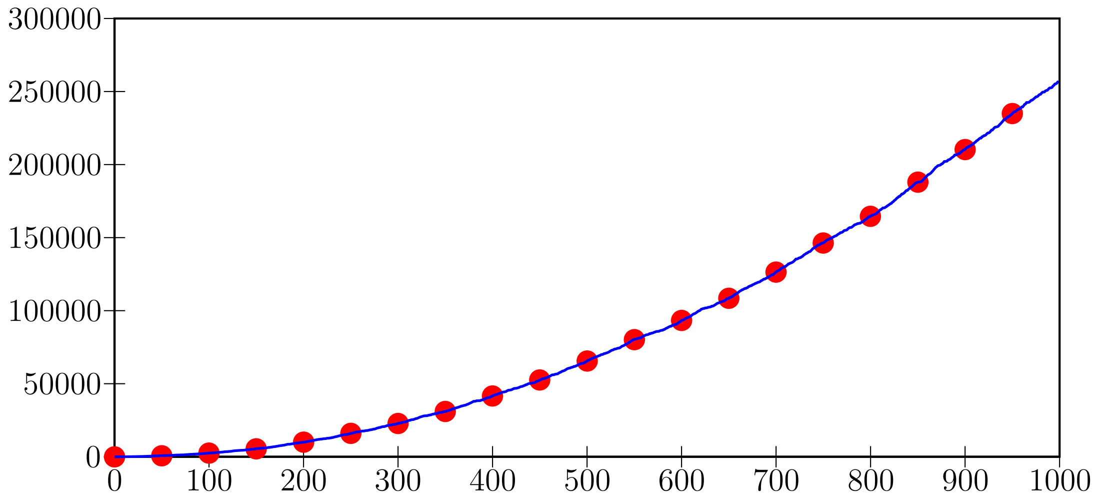
```
\readdata[nStep=10]{\data}{/home/voss/data/data1.dat}
```

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 50th data records is read and plotted.

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

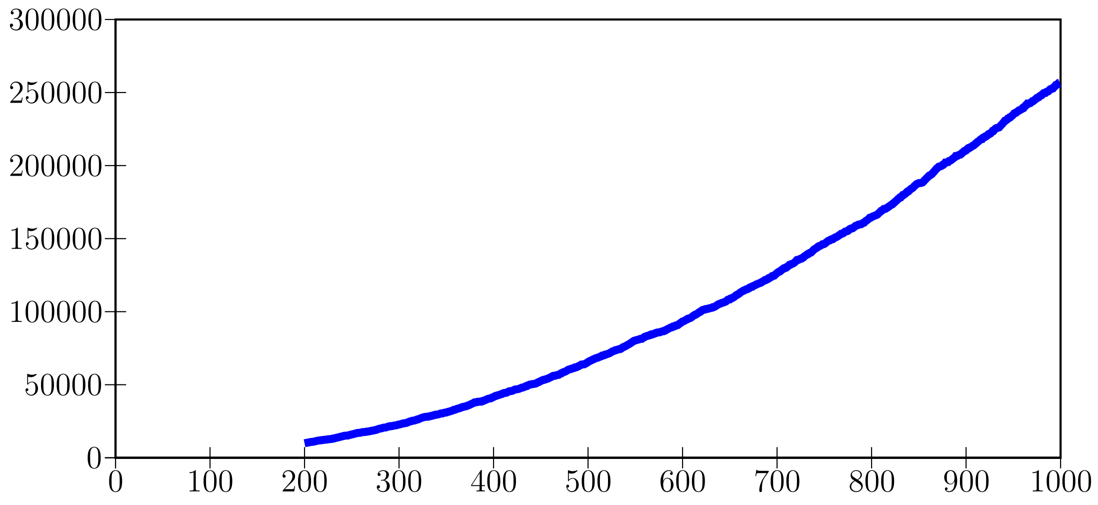
23.20.1 Example for `nStep/xStep`

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



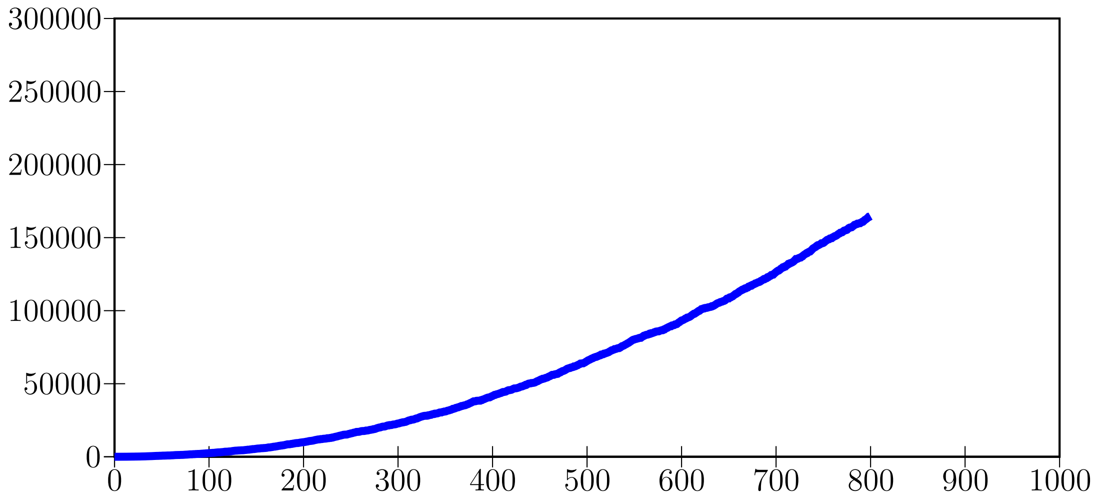
```
\readdata{\data}{examples/data.dat}
\psset{xunit=0.125mm,yunit=0.0002mm}
\begin{pspicture}(-80,-30000)(1000,310000)
\psaxes[axesstyle=frame,Dx=100,dx=100,Dy=50000,dy=50000](1000,300000)
\listplot[nStep=50,linewidth=3pt,linecolor=red,plotstyle=dots]{\data}
\listplot[linewidth=1pt,linecolor=blue]{\data}
\end{pspicture}
```

23.20.2 Example for `nStart`/`xStart`



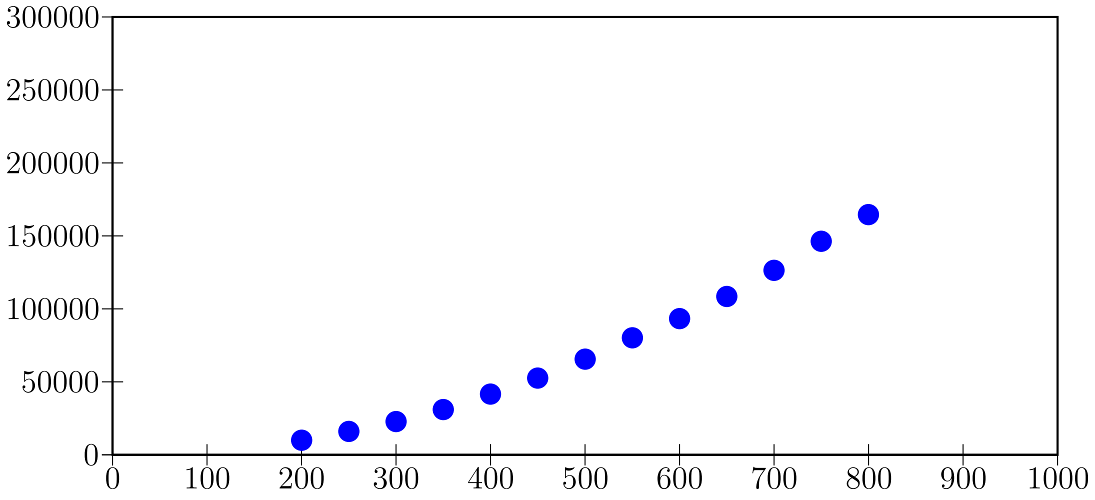
```
\readdata{\data}{examples/data.dat}
\psset{xunit=0.125mm,yunit=0.0002mm}
\begin{pspicture}(-80,-30000)(1000,310000)
\psaxes[axesstyle=frame,Dx=100,dx=100,Dy=50000,dy=50000](1000,300000)
\listplot[nStart=200,linewidth=3pt,linecolor=blue]{\data}
\end{pspicture}
```

23.20.3 Example for `nEnd/xEnd`



```
\readdata{\data}{examples/data.dat}
\psset{xunit=0.125mm,yunit=0.0002mm}
\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}
```

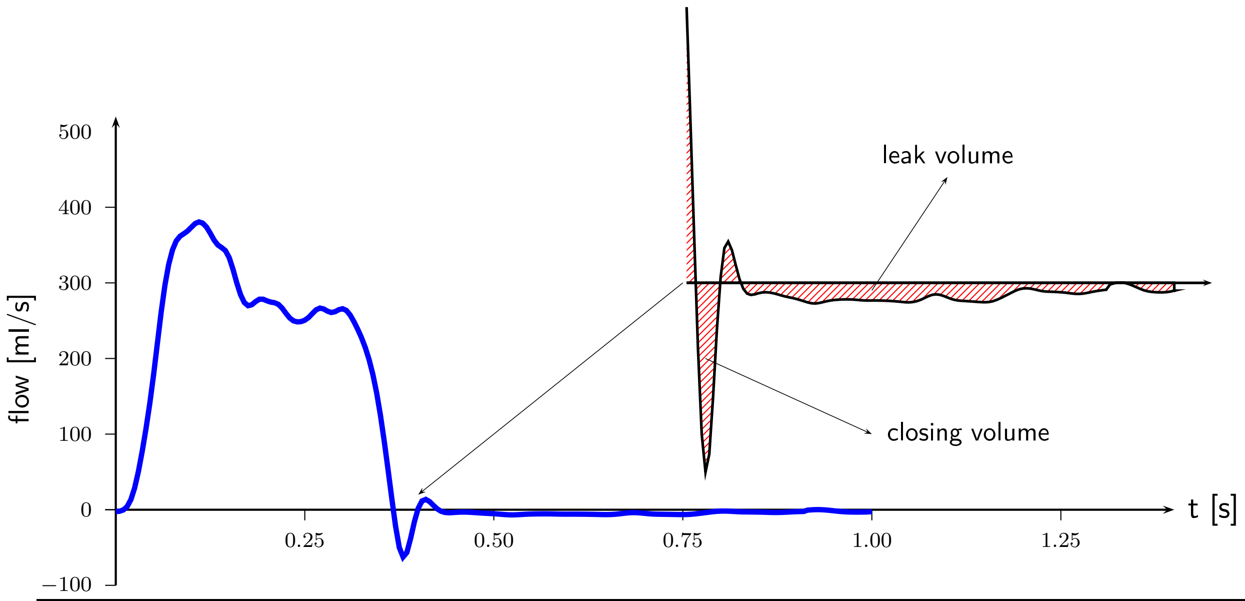
23.20.4 Example for all new options



```
\readdata{\data}{examples/data.dat}
\psset{xunit=0.125mm,yunit=0.0002mm}
\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}
```

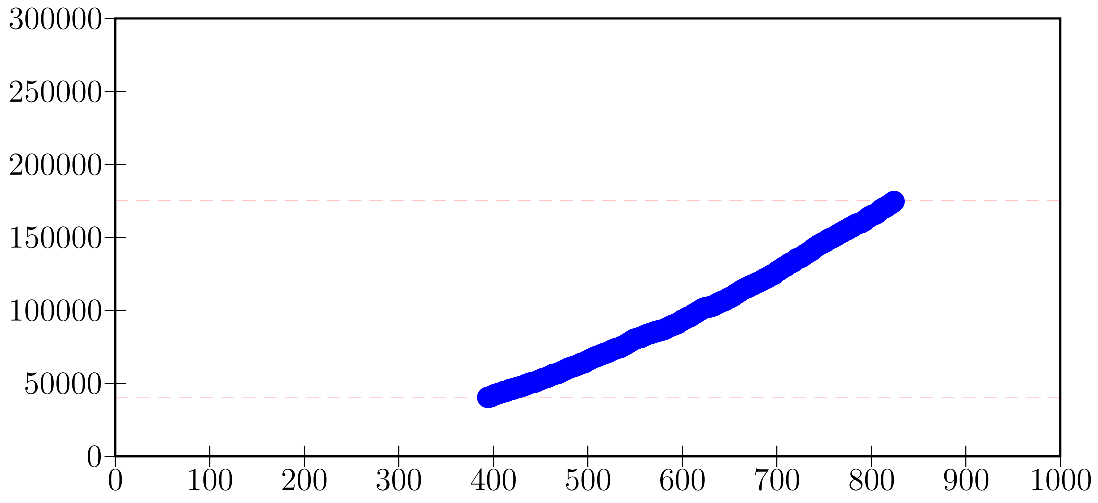
23.20.5 Example for `xStart`

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.



```
\def\pshlabel#1{\scriptsize\sffamily$#1$}
\def\psvlabel#1{\sffamily\scriptsize$#1$}
\psset{xunit=10cm, yunit=0.01cm}
\readdata{\data}{examples/data3.dat}
\begin{pspicture}(-0.1,-100)(1.5,700.0)
  \psaxes[Dx=0.25,Dy=100,dy=100\psyunit,ticksize=-4pt 0]{->}(0,0)(0,-100)(1.4,520)
  \uput[0](1.4,0){\textsf{t [s]}}
  \rput(-0.125,200){\psrotateleft{\small\sffamily flow [ml/s]}}
  \listplot[linewidth=2pt, linecolor=blue]{\data}
  \rput(0.4,300){
    \pscustom[yunit=0.04cm, linewidth=1pt]{%
      \listplot[xStart=0.355]{\data}
      \psline(1,-2.57)(1,0)(0.355,0)
      \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}
```

23.20.6 Example for yStart/yEnd



```
\readdata{\data}{examples/data.dat}
\psset{xunit=0.125mm,yunit=0.0002mm}
\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}
```

23.20.7 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:

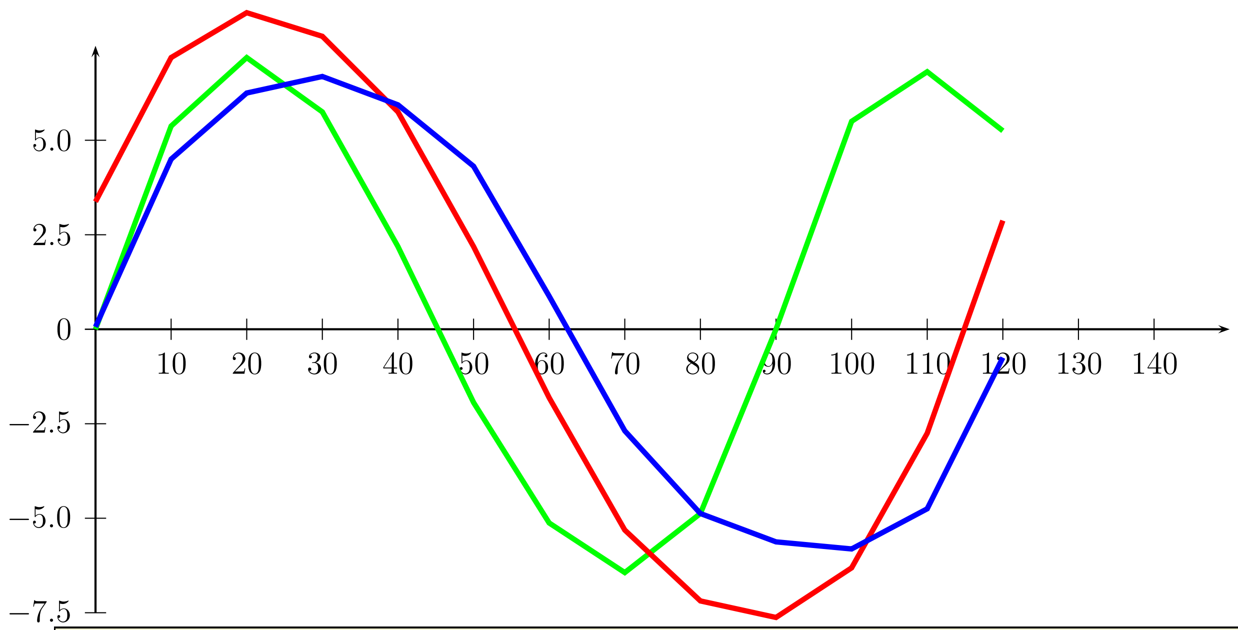
```
x y1 y2 y3 y4 ... yMax
x y1 y2 y3 y4 ... yMax
...
```

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
]%
```

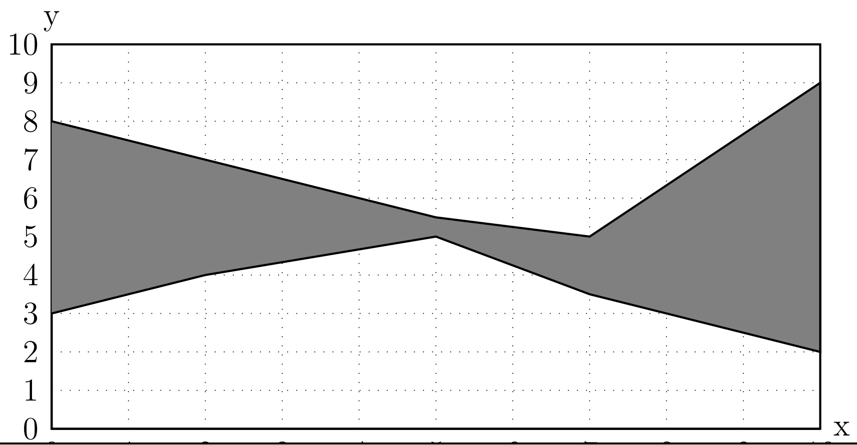
which holds data records for multiple plots (`x y1 y2 y3`). This can be plotted without any modification to the data file:



```
\readdata\Data{examples/dataMul.dat}
\psset{xunit=0.1cm, yunit=0.5cm,lly=-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}
```

23.20.8 Example for `changeOrder`

It is only possible to fill the region between two listplots with `\pscustom` if one of both has the values in a reverse order. Otherwise we do not get a closed path. With the option `ChangeOrder` the values are used in a reverse order:



```
\begin{filecontents*}{test.dat}
  0 3 8
  2 4 7
  5 5 5.5
  7 3.5 5
  10 2 9
\end{filecontents*}
\begin{psgraph}[axesstyle=frame,ticklinestyle=dotted,ticksize=0 10](0,0)(10,10){4in}{2
  in}%
   \readdata{\data}{test.dat}%
   \pscustom[fillstyle=solid,fillcolor=gray]{%
     \listplot[plotNo=2,plotNoMax=2]{\data}%
     \listplot[plotNo=1,plotNoMax=2,ChangeOrder]{\data}}
\end{psgraph}
```

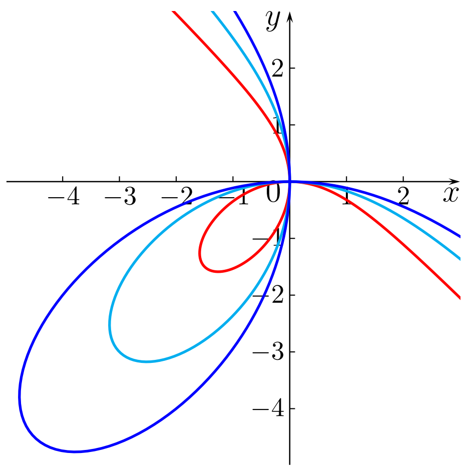
24 Polar plots

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

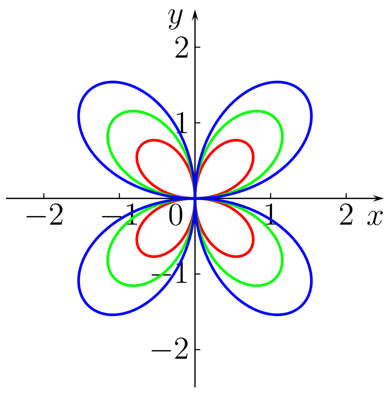
```
\psplot[polarplot=true,...]{<start angle>}{<end angle>}{<r(alpha)>}
```

The equation in PostScript code is interpreted as a function $r = f(\alpha)$, e.g. for the circle with radius 1 as $r = \sqrt{\sin^2 x + \cos^2 x}$:

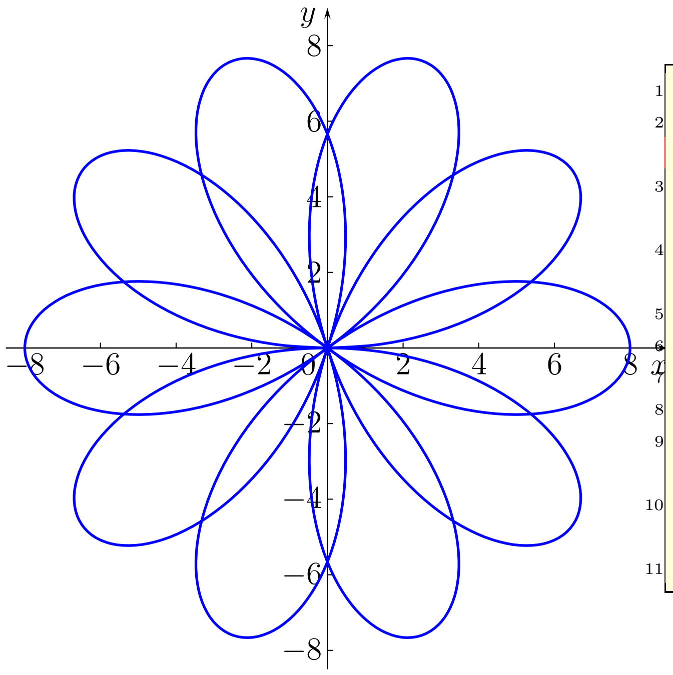
```
x sin dup mul x cos dup mul add sqrt
```



```
\resetOptions
\def\pshlabel#1{\footnotesize$#1$}
\def\psvlabel#1{\footnotesize$#1$}
\psset{plotpoints=200,unit=0.75}
\begin{pspicture}*(-5,-5)(3,3)
  \psaxes[labelsep=.75mm,arrowlength=1.75,ticksize=2
    pt,%
    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}
```



```
\resetOptions
\psset{plotpoints=200,unit=1}
\begin{pspicture}(-2.5,-2.5)(2.5,2.5)% Ulrich Dirr
 \psaxes[labelsep=.75mm,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
  ,%
  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}
```

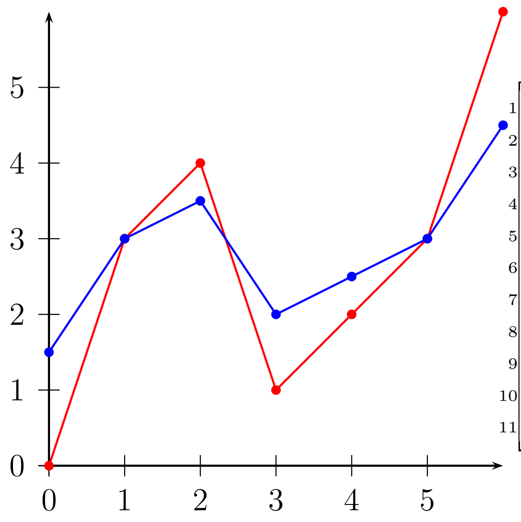
25 New commands and environments

25.1 `\pstScalePoints`

The syntax is

```
\pstScalePoints(xScale,xScale){xPS}{yPS}
```

`xScale,yScale` are decimal values as scaling factors, the `xPs` and `yPS` are additional PostScript code to the x- and y-values of the data records. This macro is only valid for the `\listplot` macro!



```
\def\data{%
  0 0 1 3 2 4 3 1
  4 2 5 3 6 6 }
\begin{pspicture}(-0.5,-1)(6,6)
  \psaxes{->}(0,0)(6,6)
  \listplot[showpoints=true,%
    linecolor=red]{\data}
  \pstScalePoints(1,0.5){}{3 add}
  \listplot[showpoints=true,%
    linecolor=blue]{\data}
\end{pspicture}
```

`\pstScalePoints(1,0.5){}{3 add}` means that **first** the value 3 is added to the y values and **second** this value is scaled with the factor 0.5. As seen for the blue line for $x = 0$ we get $y(0) = (0 + 3) \cdot 0.5 = 1.5$.

Changes with `\pstScalePoints` are always global to all following `\listplot` macros. This is the reason why it is a good idea to reset the values at the end of the `pspicture` environment.

```
\pstScalePoints(1,1){}{}
```

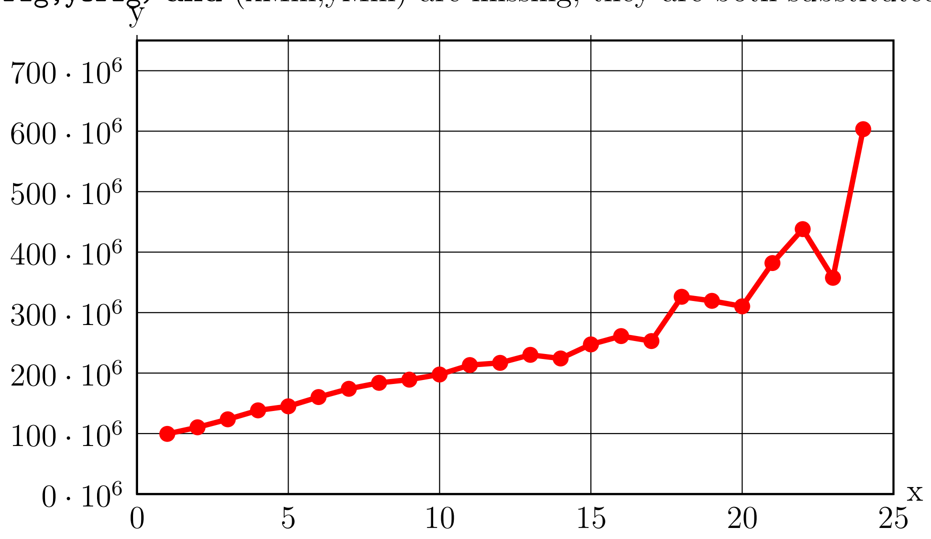
25.2 psgraph environment

This new environment does the scaling, it expects as parameter the values (without units!) for the coordinate system and the values of the physical width and height (with units!). The syntax is:

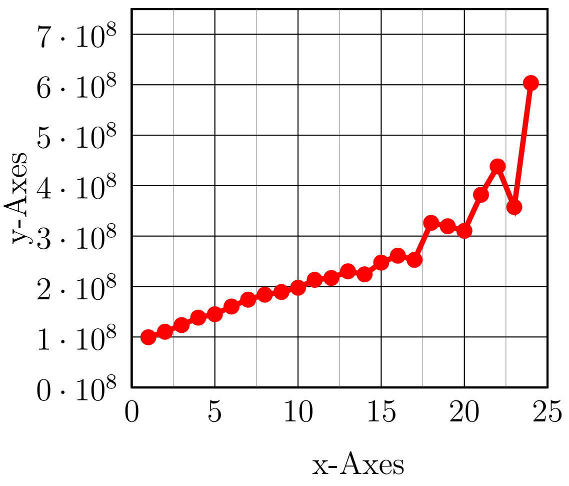
```
\psgraph[<axes options>]{<arrows>}%
    (xOrig,yOrig)(xMin,yMin)(xMax,yMax){xLength}{yLength}
...
\endpsgraph

\begin{psgraph}[<axes options>]{<arrows>}%
    (xOrig,yOrig)(xMin,yMin)(xMax,yMax){xLength}{yLength}
...
\end{psgraph}
```

where the options are valid **only** for the the `\psaxes` macro. The first two arguments have the usual `PSTricks` behaviour.

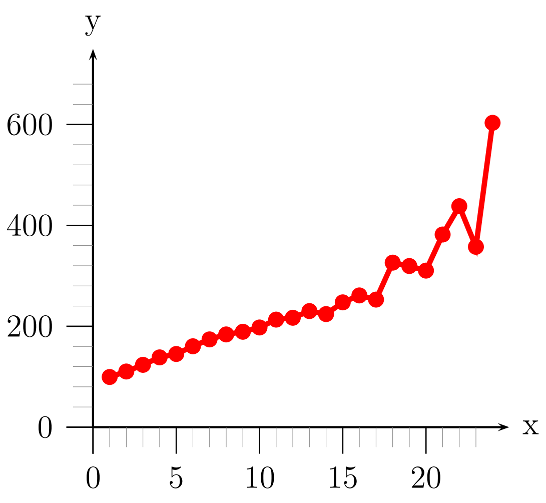
- if `(xOrig,yOrig)` is missing, it is substituted to $(0, 0)$;
- if `(xOrig,yOrig)` **and** (xMin,yMin) are missing, they are both substituted to $(0, 0)$.



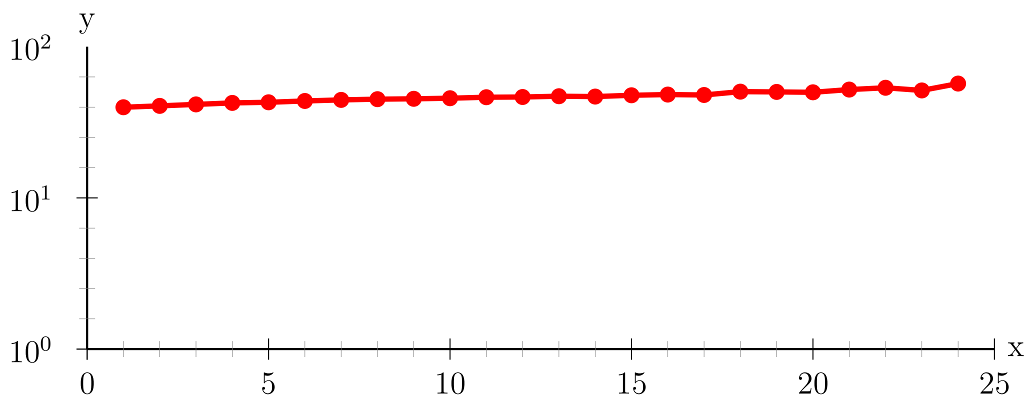
```
\readdata{\data}{demo1.dat}
\pstScalePoints(1,0.000001){}{}% (x,y){additional x operator}{y op}
\psset{llx=-1cm,lly=-1cm}
\psgraph[axesstyle=frame,xticksize=0 759,yticksize=0 25,%
    subticks=0,ylabelFactor={\cdot 10^6},%
    Dx=5,dy=100\psyunit,Dy=100](0,0)(25,750){10cm}{6cm} % parameters
   \listplot[linecolor=red, linewidth=2pt, showpoints=true]{\data}
\endpsgraph
```



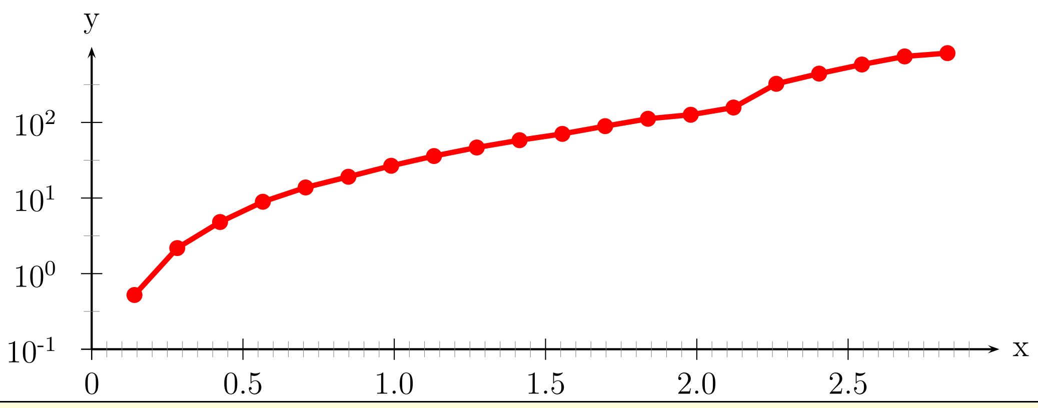
```
\readdata{\data}{demo1.dat}
\psset{xAxisLabel=x-Axes,yAxisLabel=y-Axes,llx
  =-1cm,%
   xAxisLabelPos={3cm,-1cm},yAxisLabelPos={-1.5
     cm,2.5cm}}
\pstScalePoints(1,0.00000001){}{}
\begin{psgraph}[axesstyle=frame,xticksize=0
  7.5,yticksize=0 25,subticksize=1,%
     ylabelFactor={\cdot 10^8},Dx=5,Dy=1,
       xsubticks=2](0,0)(25,7.5){5.5cm}{5cm}
  \listplot[linecolor=red, linewidth=2pt,
    showpoints=true]{\data}
\end{psgraph}
```



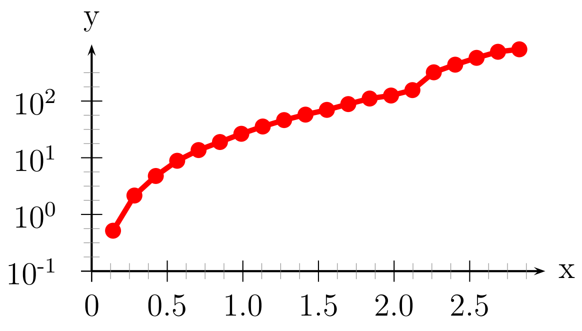
```
\readdata{\data}{demo1.dat}
\psset{llx=-0.5cm,lly=-1cm}
\pstScalePoints(1,0.000001){}{}
\psgraph[arrows=->,Dx=5,dy=200\psyunit,Dy=200,%
    subticks=5,ticksize=-10pt 0,tickwidth=0.5pt,%
    subtickwidth=0.1pt](0,0)(25,750){5.5cm}{5cm}
\listplot[linecolor=red,linewidth=2pt,showpoints=
  true,]{\data}
\endpsgraph
```



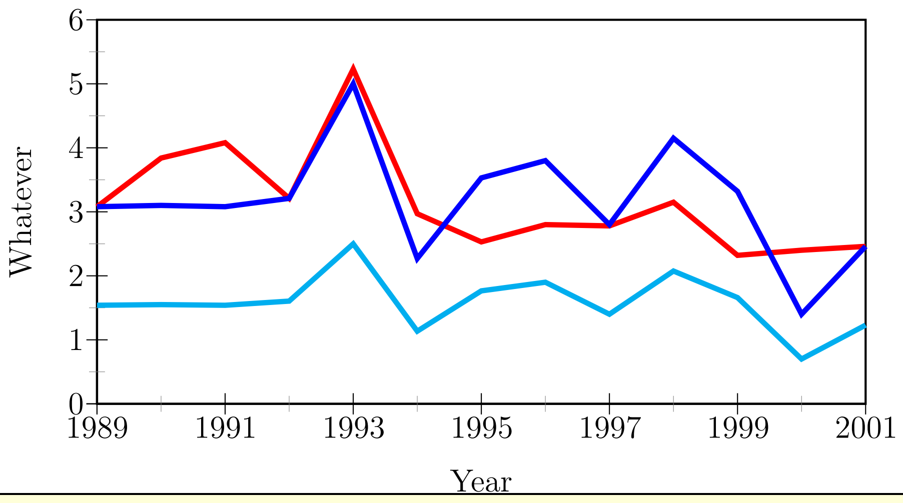
```
\pstScalePoints(1,0.2){}{log}
\psset{lly=-0.75cm}
\psgraph[ylogBase=10,Dx=5,Dy=1,subticks=5](0,0)(25,2){12cm}{4cm}
  \listplot[linecolor=red, linewidth=2pt, showpoints=true]{\data}
\endpsgraph
```



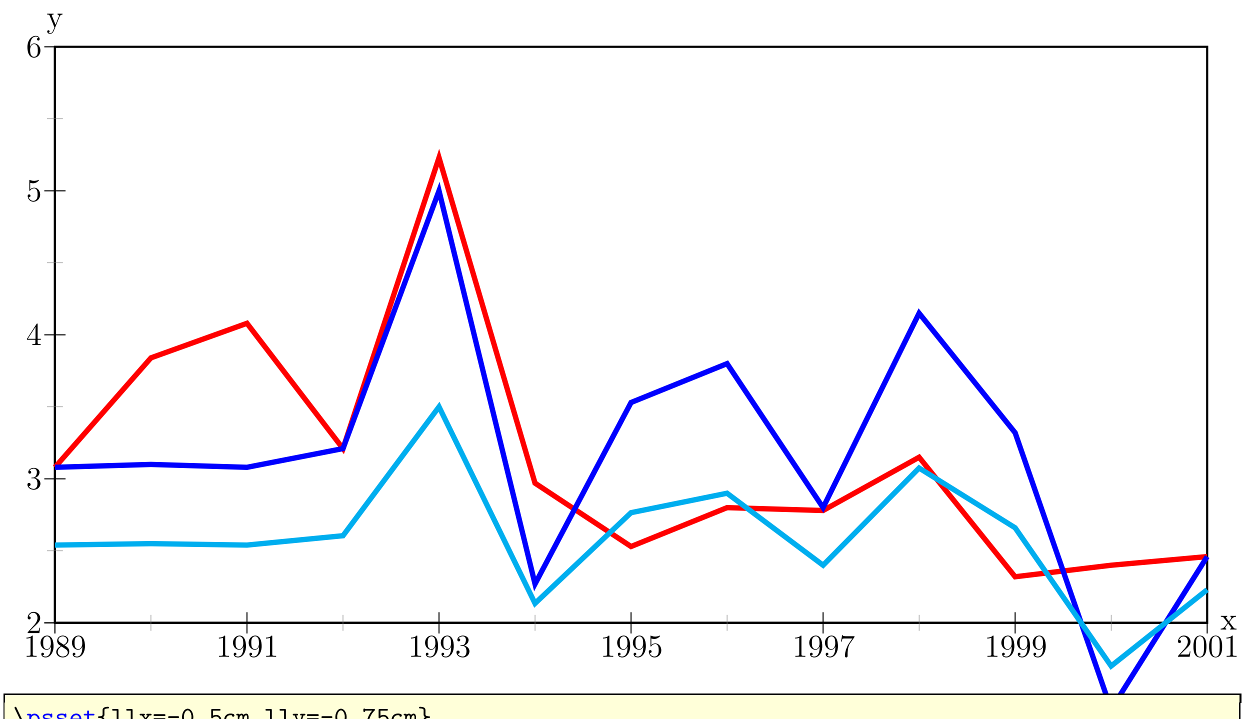
```
\readdata{\data}{demo0.dat}
\psset{lly=-0.5cm}
\pstScalePoints(1,1){}{log}
\begin{psgraph}[arrows=->,Dx=0.5,ylogBase=10,Oy=-1,xsubticks=10,%
    ysubticks=2](0,-3)(3,1){12cm}{4cm}
  \listplot[linecolor=red, linewidth=2pt, showpoints=true]{\data}
\end{psgraph}
```



```
\readdata{\data}{demo0.dat}
\pstScalePoints(1,1){}{log}
\psgraph[arrows=->,Dx=0.5,ylogBase=10,Oy=-1,
  subticks=4](0,-3)(3,1){6cm}{3cm}
  \listplot[linecolor=red, linewidth=2pt,
    showpoints=true]{\data}
\endpsgraph
```

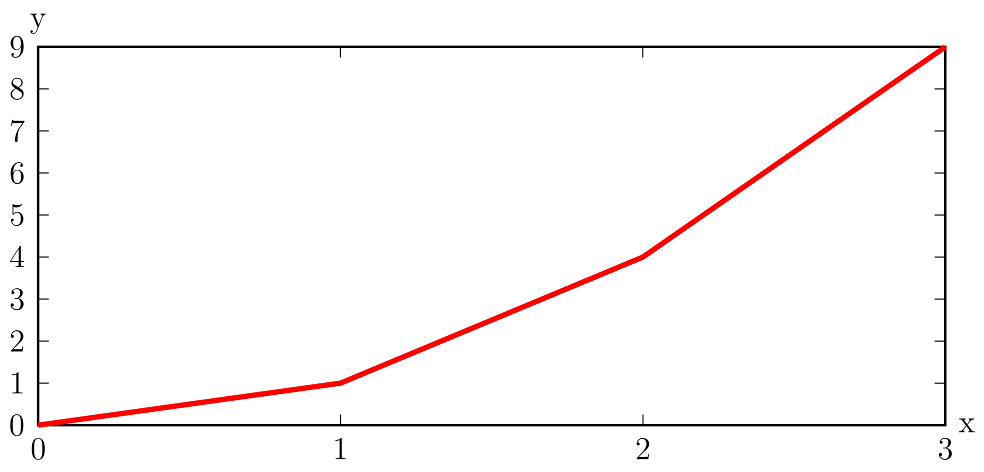


```
\readdata{\data}{demo2.dat}%
\readdata{\dataII}{demo3.dat}%
\pstScalePoints(1,1){1989 sub}{}
\psset{llx=-0.5cm,lly=-1cm, xAxisLabel=Year,yAxisLabel=Whatever,%
     xAxisLabelPos={2in,-0.4in},yAxisLabelPos={-0.4in,1in}}
\psgraph[axesstyle=frame,Dx=2,Ox=1989,subticks=2](0,0)(12,6){4in}{2in}%
  \listplot[linecolor=red,linewidth=2pt]{\data}
  \listplot[linecolor=blue,linewidth=2pt]{\dataII}
  \listplot[linecolor=cyan,linewidth=2pt,yunit=0.5]{\dataII}
\endpsgraph
```



```
\psset{llx=-0.5cm,lly=-0.75cm}
\pstScalePoints(1,1){1989 sub}{2 sub}
\begin{psgraph}[axesstyle=frame,Dx=2,Ox=1989,Oy=2,subticks=2](0,0)(12,4){6in}{3in}%
  \listplot[linecolor=red,linewidth=2pt]{\data}
  \listplot[linecolor=blue,linewidth=2pt]{\dataII}
  \listplot[linecolor=cyan,linewidth=2pt,yunit=0.5]{\dataII}
\end{psgraph}
```

An example with ticks on every side of the frame:

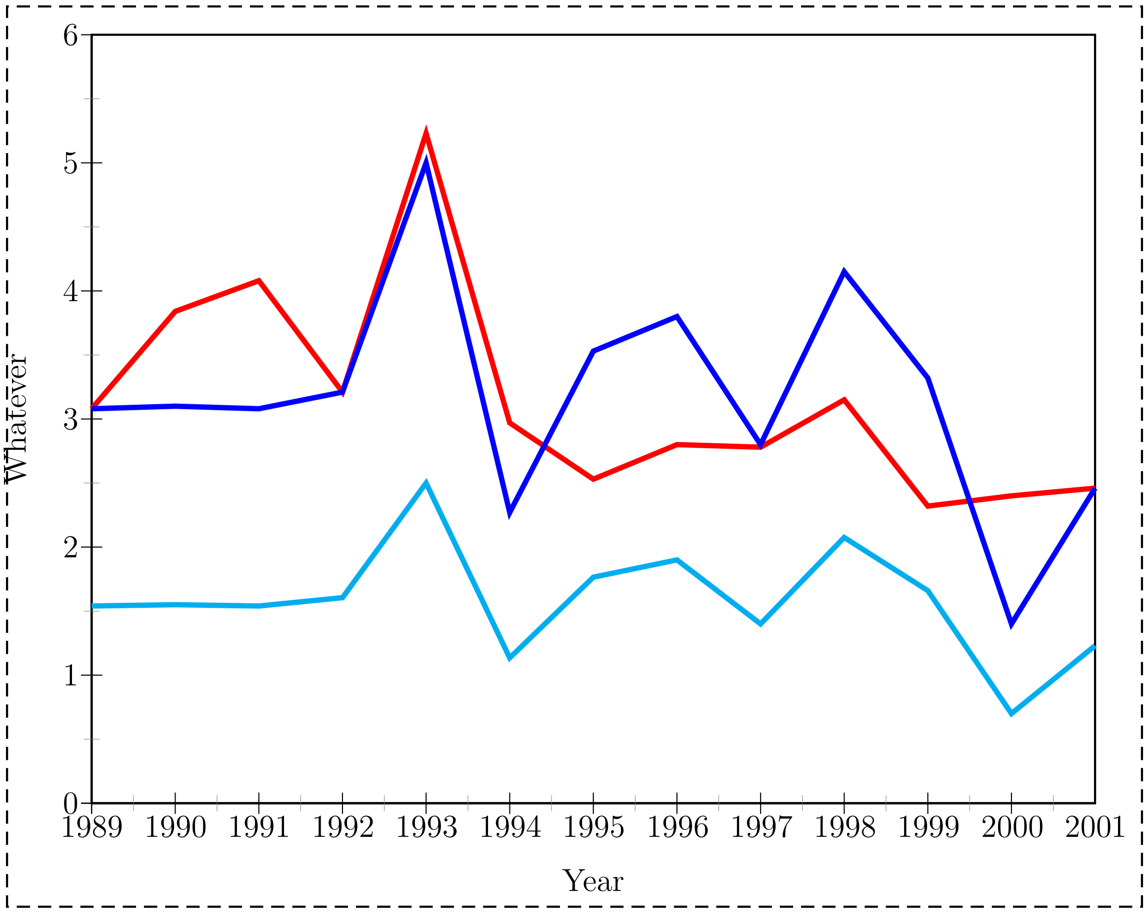


```
\def\data{0 0 1 1 2 4 3 9}
\psset{lly=-0.5cm}
\begin{psgraph}[axesstyle=frame,ticksize=0 4pt](0,0)(3.0,9.0){12cm}{5cm}
  \psaxes[axesstyle=frame,labels=none,ticksize=-4pt 0](3,9)(0,0)(3,9)
  \listplot[linecolor=red,linewidth=2pt]{\data}
\end{psgraph}
```

25.2.1 The new options

name	default	meaning
`xAxisLabel`	x	label for the x-axis
`yAxisLabel`	y	label for the y-axis
`xAxisLabelPos`	`{}`	where to put the x-label
`yAxisLabelPos`	`{}`	where to put the y-label
`llx`	`0pt`	trim for the lower left x
`lly`	`0pt`	trim for the lower left y
`urx`	`0pt`	trim for the upper right x
`ury`	`0pt`	trim for the upper right y

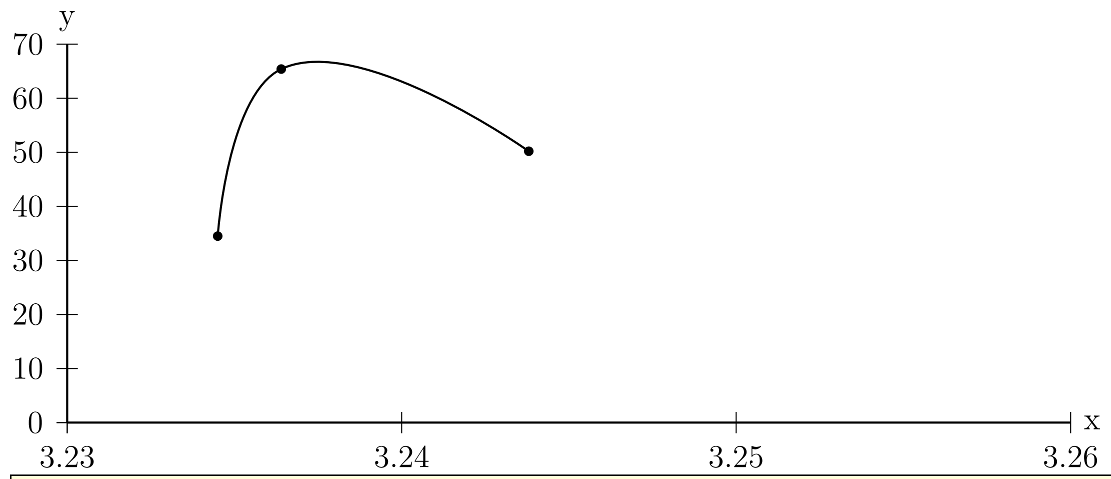
There is one restriction in using the trim parameters, they must been set **before** `psgraph` is called. They are senseless, when using as parameters of `psgraph` itself.



```
\psset{llx=-1cm,lly=-1.25cm,urx=0.5cm,ury=0.1in,xAxisLabel=Year,%
   yAxisLabel=Whatever,xAxisLabelPos={.4\linewidth,-0.4in},%
   yAxisLabelPos={-0.4in,2in}}
\pstScalePoints(1,1){1989 sub}{}
\psframebox[linestyle=dashed,boxsep=0pt]{%
\begin{psgraph}[axesstyle=frame,Ox=1989,subticks=2](0,0)(12,6){0.8\linewidth}{4in}%
  \listplot[linecolor=red,linewidth=2pt]{\data}%
  \listplot[linecolor=blue,linewidth=2pt]{\dataII}%
  \listplot[linecolor=cyan,linewidth=2pt,yunit=0.5]{\dataII}%
\end{psgraph}%
}
```

25.2.2 Problems

Floating point operations in TEX are a real mess, which causes a lot of problems when there are very small oder very big units. With the options of `\pst-plot` it is possible to choose normal units (whatever this may be ...), but plotting the data as usual.



```
\begin{filecontents*}{test.dat}
3.2345 34.5
3.2364 65.4
3.2438 50.2
\end{filecontents*}

\psset{lly=-0.5cm,llx=-1cm}
\readdata{\data}{test.dat}
\pstScalePoints(1,1){3.23 sub 100 mul}{}
\begin{psgraph}[Ox=3.23,Dx=0.01,dx=\psxunit,Dy=10](0,0)(3,70){0.8\linewidth}{5cm}%
  \listplot[showpoints=true,plotstyle=curve]{\data}
\end{psgraph}
```

This example shows some important facts:

- `3.23 sub 100 mul`: the x values are now $0.45; 0.64; 1.38$
- `Ox=3.23`: the origin of the x axis is set to 3.23
- `Dx=0.01`: the increment of the labels
- `dx=\psxunit`: uses the calculated unit value to get every unit a label
- `Dy=10`: increase the y labels by 10

Using the internal `\psxunit` one can have dynamical x-units, depending to the linewidth od the document.

25.3 \psStep

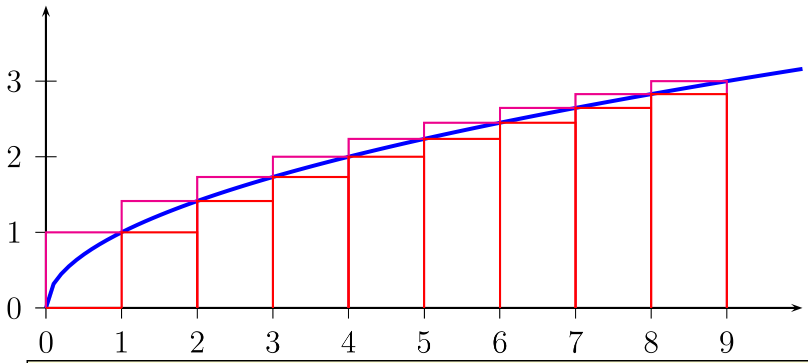
`\psStep` caclulates a step function for the upper or lower sum of a given function. The available option is

`StepType=lower|upper`

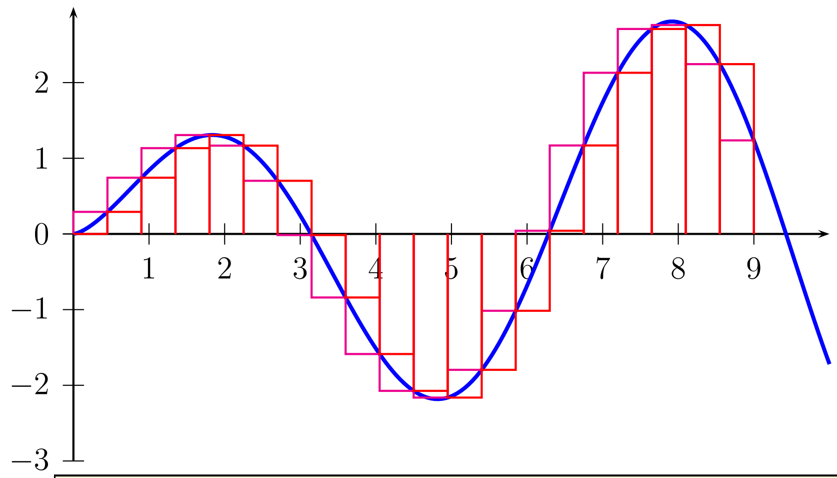
with `lower` as the default setting. The syntax of the function is

`\psStep[options](x1,x2){n}{function}`

(x1,x2) is the given Intervall for the step wise caculated function, n is the number of the rectabgles and `function` is the mathematical function in postfix notation.



```
\begin{pspicture}(-0.5,-0.5)(10,4) \psaxes{->}(10,4)
  \psplot[plotpoints=100,linewidth=1.5pt,linecolor=blue,algebraic]{0}{10}{sqrt(x)}
  \psStep[algebraic,linecolor=magenta,StepType=upper](0,9){9}{sqrt(x)}
  \psStep[linecolor=red,linestyle=dashed](0,9){9}{x sqrt }
\end{pspicture}
```



```
\begin{pspicture}[plotpoints=200](-0.5,-3)(10,3) \psaxes{->}(0,0)(0,-3)(10,3)
  \psplot[linewidth=1.5pt,linecolor=blue,algebraic]{0}{10}{sqrt(x)*sin(x)}
  \psStep[algebraic,linecolor=magenta,StepType=upper](0,9){20}{sqrt(x)*sin(x)}
  \psStep[linecolor=red,linestyle=dashed](0,9){20}{x sqrt x RadtoDeg sin mul}
\end{pspicture}
```

25.4 `\psplotTangent`[2]

There is an additional option, named `Derive` vor an alternative function (see following example) to calculate the slope of the tangent. This will be in general the first derivation, but can also be any other function. If this option is different to to the default value `Derive=default`, then this function is taken to calculate the slope. For the other cases, `pstricks-add` builds a secant with -0.00005<x<0.00005, calculates the slope and takes this for the tangent. This maybe problematic in some cases of special functions or x values, then it may be appropriate to use the Derivate option.

The macro expects three parameters:

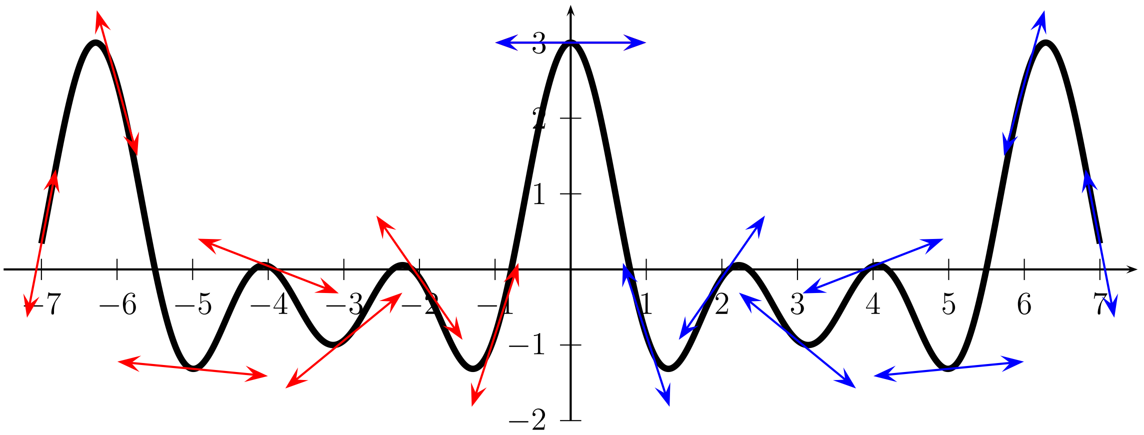
x : the x value of the function for which the tangent should be calculated

dx : the dx to both sides of the x value

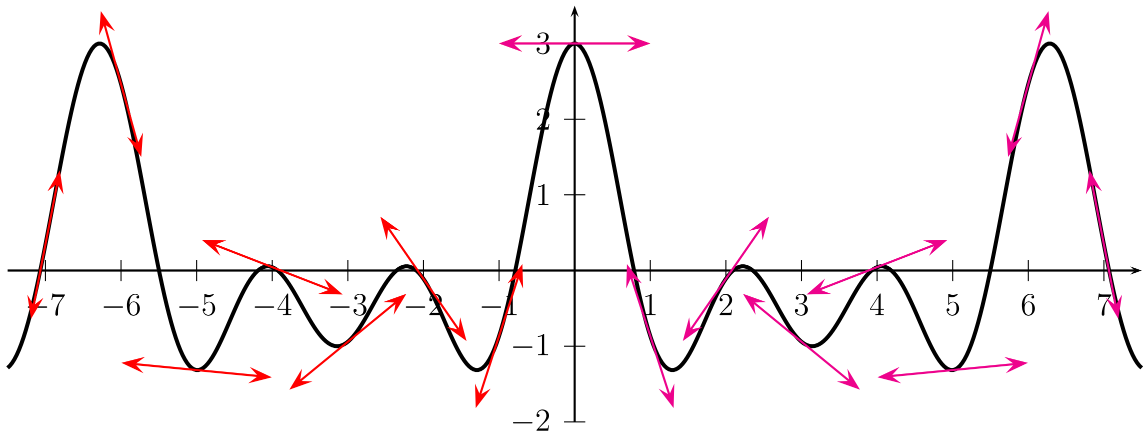
$f(x)$: the function in infix (with option `algebraic`) or the default postfix (PostScript) notation

[2]This part is adapted from the package `pst-eqdf`, written by Dominique Rodriguez.

The following examples show the use of the algebraic option together with the Derive option. Remember that using the `algebraic` option implies that the angles have to be in the radian unit!

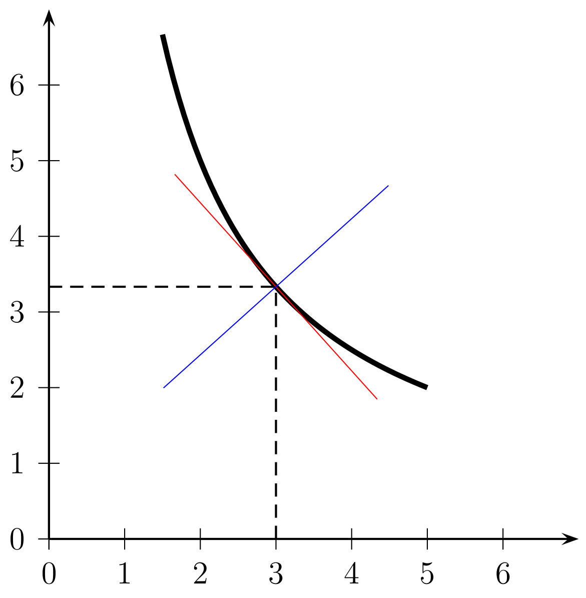


```
\def\F{x RadtoDeg dup dup cos exch 2 mul cos add exch 3 mul cos add}
\def\Fp{x RadtoDeg dup dup sin exch 2 mul sin 2 mul add exch 3 mul sin 3
   mul add neg}
\psset{plotpoints=1001}
\begin{pspicture}(-7.5,-2.5)(7.5,4)%X\psgrid
  \psaxes{->}(0,0)(-7.5,-2)(7.5,3.5)
  \psplot[linewidth=3\pslinewidth]{-7}{7}{\F}
  \psset{linecolor=red, arrows=<->, arrowscale=2}
  \multido{\n=-7+1}{8}{\psplotTangent{\n}{1}{\F}}
  \psset{linecolor=magenta, arrows=<->, arrowscale=2}%
  \multido{\n=0+1}{8}{\psplotTangent[linecolor=blue, Derive=\Fp]{\n
    }{1}{\F}}
\end{pspicture}
```



```
\def\Falg{cos(x)+cos(2*x)+cos(3*x)}   \def\Fpalg{-sin(x)-2*sin(2*x)-3*
  sin(3*x)}
\begin{pspicture}(-7.5,-2.5)(7.5,4)%\psgrid
  \psaxes{->}(0,0)(-7.5,-2)(7.5,3.5)
  \psplot[linewidth=1.5pt,algebraic,plotpoints=500]{-7.5}{7.5}{\Falg}
  \multido{\n=-7+1}{8}{\psplotTangent[linecolor=red,arrows=<->,
    arrowscale=2,algebraic]{\n}{1}{\Falg}}
  \multido{\n=0+1}{8}{\psplotTangent[linecolor=magenta,%
      arrows=<->,arrowscale=2,algebraic,Derive={\Fpalg}]{\n}{1}{\Falg}}
\end{pspicture}
```

The next example shows the use of `Derive` option to draw the perpendicular line of the tangent.



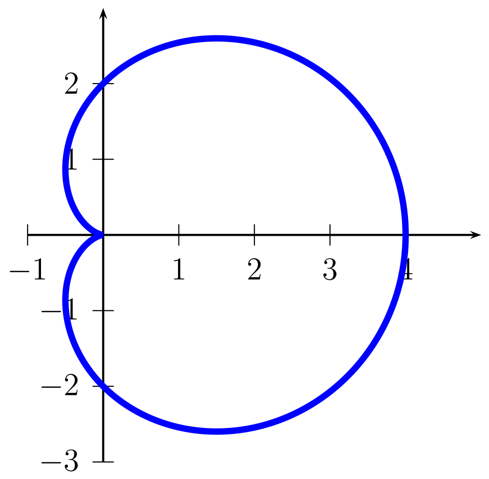
```
\begin{pspicture}(-0.5,-0.5)(7.25,7.25)
  \def\Func{10 x div}
  \psaxes[arrowscale=1.5]{->}(7,7)
  \psplot[linewidth=2pt,algebraic]{1.5}{5}{10/x}
  \psplotTangent[linewidth=.5\pslinewidth,linecolor=
    red,algebraic]{3}{2}{10/x}
  \psplotTangent[linewidth=.5\pslinewidth,linecolor=
    blue,algebraic,Derive=(x*x)/10]{3}{2}{10/x}
  \psline[linestyle=dashed](!0 /x 3 def \Func)(!3 /x
     3 def \Func)(3,0)
\end{pspicture}
```

25.4.1 A `polarplot` example

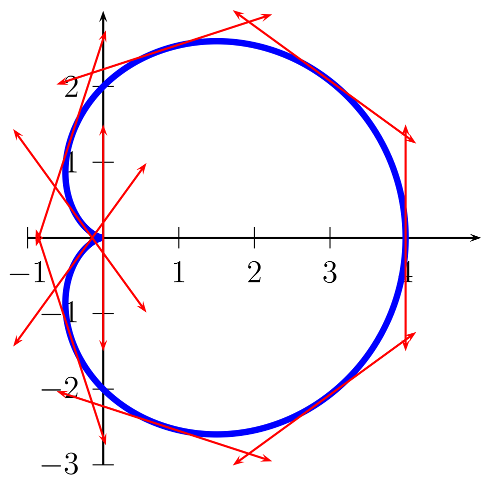
Let's work with the classical cardioid : $\rho = 2(1+\cos(\theta))$ and $\dfrac{d\rho}{d\theta} = -2\sin(\theta)$. The Derive option always expects the $\frac{d\rho}{d\theta}$ value and uses internally the equation for the derivation of implicit defined functions:

$$\frac{dy}{dx} = \frac{\rho\prime \cdot \sin\theta + x}{\rho\prime \cdot \cos\theta - y}$$

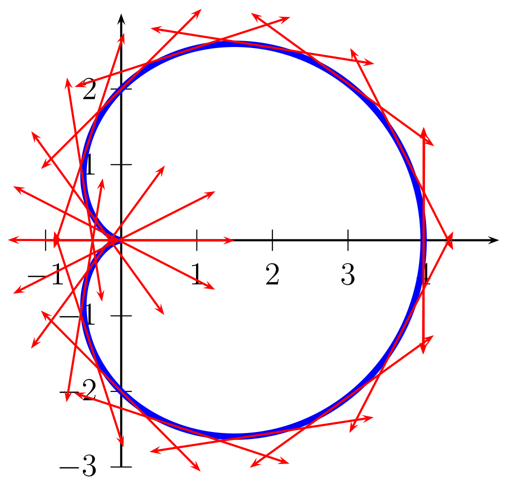
where $x = r \cdot \cos\theta$ and $y = r \cdot \sin\theta$



```
\begin{pspicture}(-1,-3)(5,3)%\psgrid[subgridcolor=lightgray]
  \psaxes{->}(0,0)(-1,-3)(5,3)
  \psplot[polarplot,linewidth=3\pslinewidth,linecolor=blue,%
     plotpoints=500]{0}{360}{1 x cos add 2 mul}
\end{pspicture}
```



```
\begin{pspicture}(-1,-3)(5,3)%\psgrid[subgridcolor=lightgray]
  \psaxes{->}(0,0)(-1,-3)(5,3)
  \psplot[polarplot,linewidth=3\pslinewidth,linecolor=blue,
    plotpoints=500]{0}{360}{1 x cos add 2 mul}
  \multido{\n=0+36}{10}{%
     \psplotTangent[polarplot,linecolor=red,arrows=<->]{\n
       }{1.5}{1 x cos add 2 mul} }
\end{pspicture}
```

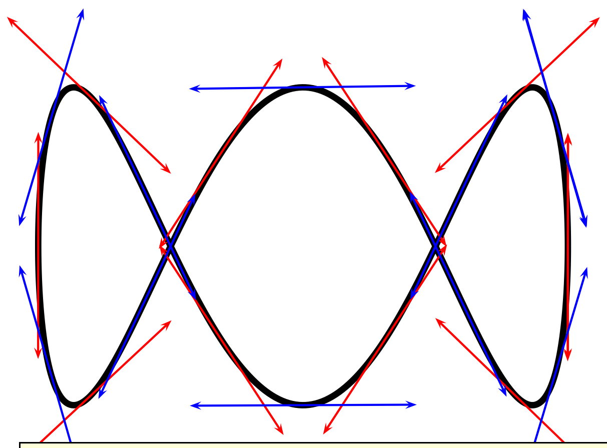


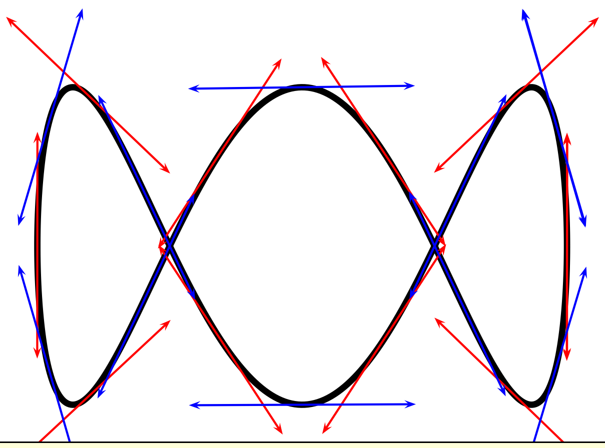
```
\begin{pspicture}(-1,-3)(5,3)%\psgrid[subgridcolor=lightgray]
  \psaxes{->}(0,0)(-1,-3)(5,3)
  \psplot[polarplot,linewidth=3\pslinewidth,linecolor=blue,
    algebraic,plotpoints=500]{0}{6.289}{2*(1+cos(x))}
  \multido{\r=0.000+0.314}{21}{%
     \psplotTangent[polarplot,Derive=-2*sin(x),algebraic,
       linecolor=red,arrows=<->]{\r}{1.5}{2*(1+cos(x))} }
\end{pspicture}
```

25.4.2 A `\parametricplot` example

Let's work with a Lissajou curve : $\begin{cases} x = 3.5\cos(2t) \\ y = 3.5\sin(6t) \end{cases}$ whose derivative is : $\begin{cases} x = -7\sin(2t) \\ y = 21\cos(6t) \end{cases}$

The parameter must be the letter t instead of x and when using the `algebraic` option divide the two equations by a | (see example).





```
\def\Lissa{t dup 2 RadtoDeg mul cos 3.5 mul exch 6 mul RadtoDeg sin 3.5 mul}%
\psset{yunit=0.6}
\begin{pspicture}(-4,-4)(4,6)
  \parametricplot[plotpoints=500,linewidth=3\pslinewidth]{0}{3.141592}{\Lissa}
  \multido{\r=0.000+0.314}{11}{%
    \psplotTangent[linecolor=red,arrows=<->]{\r}{1.5}{\Lissa} }
  \multido{\r=0.157+0.314}{11}{%
    \psplotTangent[linecolor=blue,arrows=<->]{\r}{1.5}{\Lissa} }
\end{pspicture}\hfill%
\def\LissaAlg{3.5*cos(2*t)|3.5*sin(6*t)} \def\LissaAlgDer{-7*sin(2*t)|21*cos(6*t)}%
\begin{pspicture}(-4,-4)(4,6)
  \parametricplot[algebraic,plotpoints=500,linewidth=3\pslinewidth]{0}{3.141592}{\LissaAlg}
  \multido{\r=0.000+0.314}{11}{%
    \psplotTangent[algebraic,linecolor=red,arrows=<->]{\r}{1.5}{\LissaAlg} }
  \multido{\r=0.157+0.314}{11}{%
    \psplotTangent[algebraic,linecolor=blue,arrows=<->,%
      Derive=\LissaAlgDer]{\r}{1.5}{\LissaAlg} }
\end{pspicture}
```

25.5 `\psplotDiffEqn` – solving diffential equations[3]

A differential euqation of first order is like

$$y' = f(x, y) \tag{1}$$

where y is a function of x. We define some vectors $Y = [y, y', \cdots, y^{(n-1)}]$ und $Y' = [y', y'', \cdots, y^n]$, depending to the order n. The syntax of the macro is

```
\psplotDiffEqn[options]{x0}{x1}{y0}{f(x,y)}
```

- `options`: the `\psplotDiffEqn` specific options and all other of PSTricks, which make sense;
- x_0: the start value;
- x_1: the end value of the definition interval;
- y_0: the initial values for $y(x_0)$ $y'(x_0)$ $\ldots$;
- $f(x, y, y', ...)$: the differential equation, depending to the number of initial values, e.g.: `{0 1}` for y_0 are two initial values, so that we have a differential equation of second order $f(x, y, y')$ and the macro leaves y y' on the stack.

The new options are:

- `method`: integration method (`euler` for order 1 euler method, `rk4` for 4th order Runge-Kutta method);
- `whichabs`: select the abscissa for plotting the graph, by default it is x, but you can specify a number which represent a position in the vector y;
- `whichord`: same as precedent for the ordinate, by default $y(0)$;
- `plotfuncx`: describe a ps function for the abscissa, parameter `whichabs` becomes useless;
- `plotfuncy`: idem for ordoinate;
- `buildvector`: boolean parameter for specifying the input-output of the f description:

  `true` (default): y is put on the stack element by element, y' must be given in the same way;

  `false` : y is put on the stack as a vector, y' must be returned in the same way;

[3]This part is adapted from the package `pst-eqdf`, written by Dominique Rodriguez.

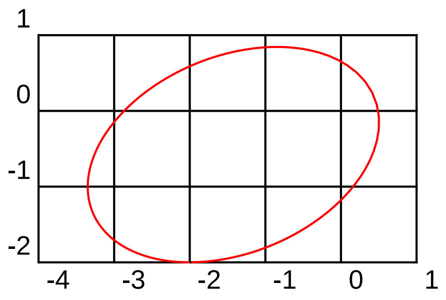
- `algebraic`: algebraic description for f, `buildvector` parameter is useless when activating this option.

The variable t (time) is represented by x in the `\psplotDiffEqn`, x and y (position) are represented respectively by $y[0]$ and $y[1]$ For `funcx` and `funcy` there is some examples at the end.

```
\def\Grav{%
  /yp2 exch def /xp2 exch def /ay2 exch def /ax2 exch def
  /yp1 exch def /xp1 exch def /ay1 exch def /ax1 exch def
  /ro2 ax2 ax1 sub dup [21~mul ay2 ay1 sub dup mul add def
  xp1 yp1
  ax2 ax1 sub ro2 sqrt div ro2 div
  ay2 ay1 sub ro2 sqrt div ro2 div
  xp2 yp2
  3 index -20 mul 3 index -20 mul}
             %%  0   1   2    3   4   5    6    7
             %% x1  y1  x'1  y'1  x2  y2  x'2  y'2
\def\InitCond{ 1   1   .1   0  -1  -1   -2    0}
```

25.5.1 plotfuncx and plotfuncy

```
\begin{pspicture}(-4,-2.5)(1,1.1)\psgrid[subgriddiv=1]
  \psplotDiffEqn[linecolor=red, method=rk4, plotpoints=200,%
     plotfuncx=y dup 4 get exch 0 get sub,%
     plotfuncy=dup 5 get exch 1 get sub ]{0}{3.9}{\InitCond}{\Grav}
\end{pspicture}
```

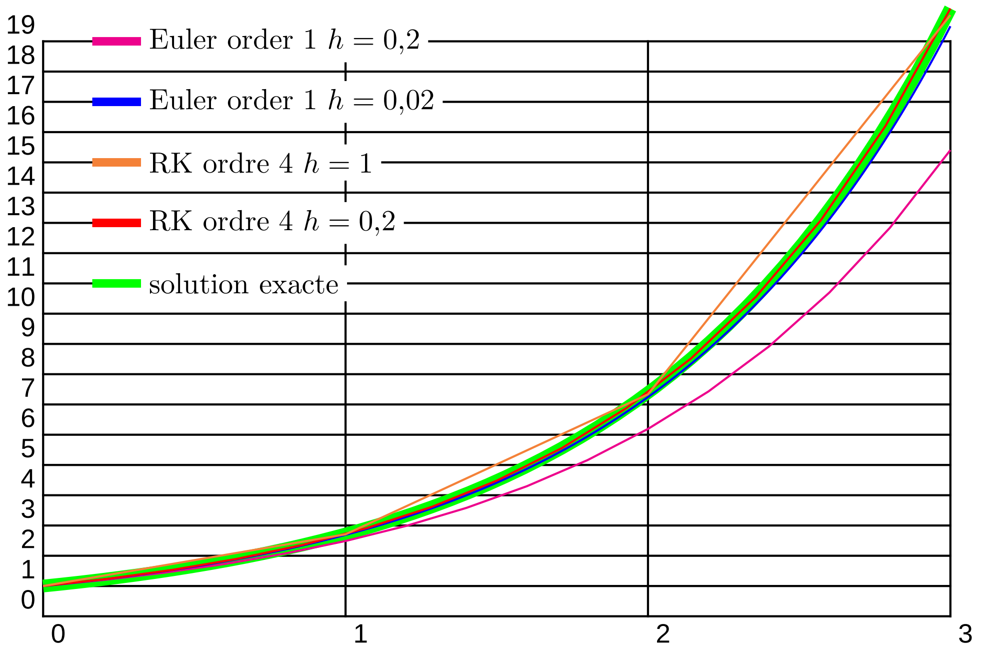


The center of the landmark is set to $y[0]$ and $y[1]$ There is also a drawing of the speed (vitesse in french) of the stars which uses these parameters.

25.5.2 Simple equation of first order$y' = y$

For the initial value $y(0) = 1$ we have the solution $y(x) = e^x$. y is always on the stack, so we have to do nothing. Using the `algebraic` option, we write it as `y[0]`. The following example shows different solutions depending to the number of plotpoints with $y_0 = 1$:

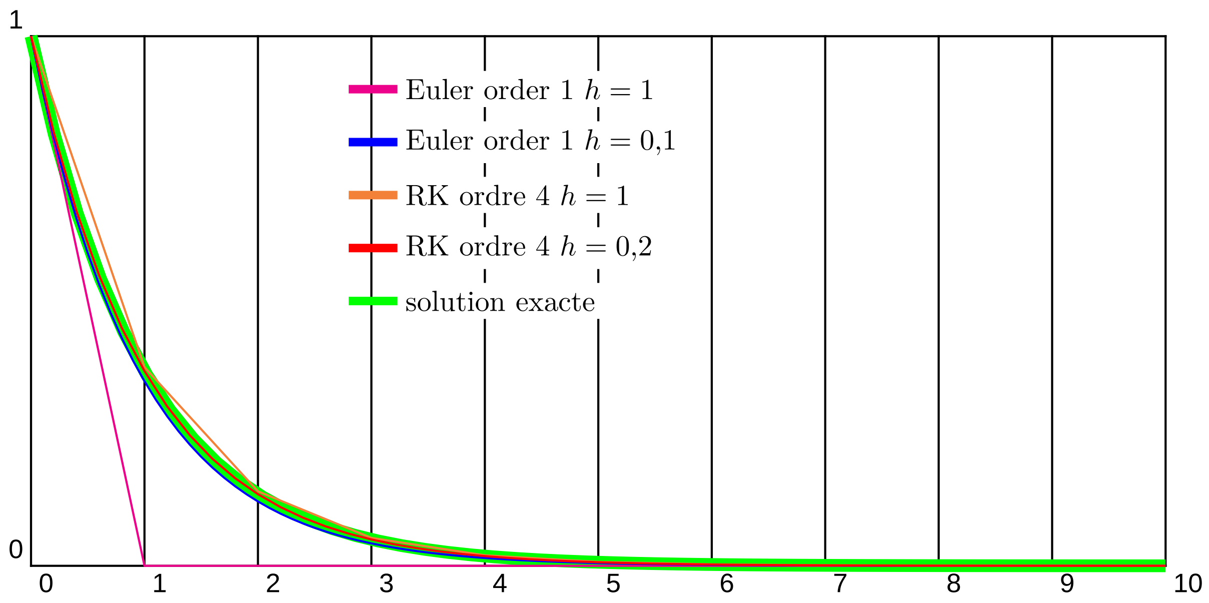
```
\psset{xunit=4, yunit=.4}
\begin{pspicture}(3,19)\psgrid[subgriddiv=1]
  \psplot[linewidth=6\pslinewidth, linecolor=green]{0}{3}{Euler x exp}
  \psplotDiffEqn[linecolor=magenta,plotpoints=16,algebraic=true
    ]{0}{3}{1}{y[0]}
  \psplotDiffEqn[linecolor=blue,plotpoints=151]{0}{3}{1}{}
  \psplotDiffEqn[linecolor=red,method=rk4,plotpoints=15]{0}{3}{1}{}
  \psplotDiffEqn[linecolor=Orange,method=rk4,plotpoints=4]{0}{3}{1}{}
  \psset{linewidth=4\pslinewidth}
  \rput*(0.35,19){\psline[linecolor=magenta](-.75cm,0)}
  \rput*[l](0.35,19){\small Euler order 1 $h=0{,}2$}
  \rput*(0.35,17){\psline[linecolor=blue](-.75cm,0)}
  \rput*[l](0.35,17){\small Euler order 1 $h=0{,}02$}
  \rput*(0.35,15){\psline[linecolor=Orange](-.75cm,0)}
  \rput*[l](0.35,15){\small RK ordre 4 $h=1$}
  \rput*(0.35,13){\psline[linecolor=red](-.75cm,0)}
  \rput*[l](0.35,13){\small RK ordre 4 $h=0{,}2$}
  \rput*(0.35,11){\psline[linecolor=green](-.75cm,0)}
  \rput*[l](0.35,11){\small solution exacte}
\end{pspicture}
```



25.5.3 $y' = -y$

For the initial value $y(0) = 1$ we get the solution $y(x) = e^{-x}$, which is seen in the following example with $y_0 = 1$:

```
\def\Funct{neg}\def\FunctAlg{-y[0]}
\psset{xunit=1.5, yunit=7}
\begin{pspicture}(0,0)(10,1)\psgrid[subgriddiv=1]
  \psplot[linewidth=6\pslinewidth,linecolor=green]{0}{10}{Euler x neg exp}
  \psplotDiffEqn[linecolor=magenta,plotpoints=11]{0}{10}{1}{\Funct}
  \psplotDiffEqn[linecolor=blue,plotpoints=101,algebraic=true]{0}{10}{1}{\
    FunctAlg}
  \psplotDiffEqn[linecolor=Orange,method=rk4,plotpoints=11]{0}{10}{1}{\Funct}
  \psplotDiffEqn[linecolor=red,method=rk4,plotpoints=51]{0}{10}{1}{\Funct}
  \psset{linewidth=4\pslinewidth}
  \rput*(3.3,.9){\psline[linecolor=magenta](-.75cm,0)}
  \rput*[l](3.3,.9){\small Euler order 1 $h=1$}
  \rput*(3.3,.8){\psline[linecolor=blue](-.75cm,0)}
  \rput*[l](3.3,.8){\small Euler order 1 $h=0{,}1$}
  \rput*(3.3,.7){\psline[linecolor=Orange](-.75cm,0)}
  \rput*[l](3.3,.7){\small RK ordre 4 $h=1$}
  \rput*(3.3,.6){\psline[linecolor=red](-.75cm,0)}
  \rput*[l](3.3,.6){\small RK ordre 4 $h=0{,}2$}
  \rput*(3.3,.5){\psline[linecolor=green](-.75cm,0)}
  \rput*[l](3.3,.5){\small solution exacte}
\end{pspicture}
```



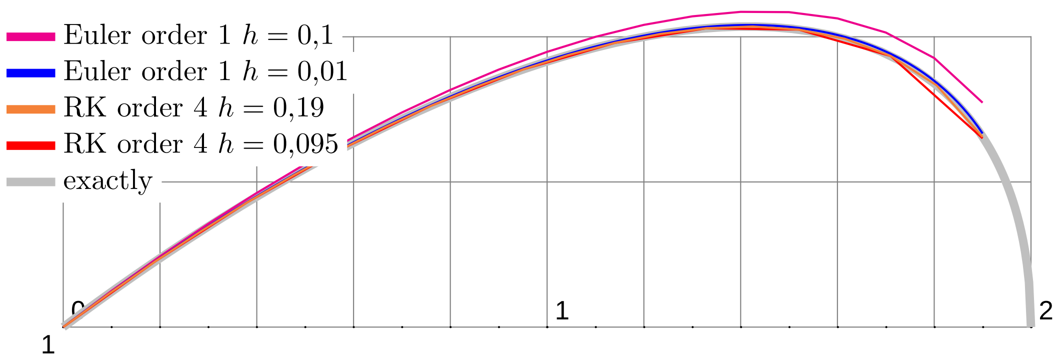
25.5.4 $y' = \dfrac{2 - ty}{4 - t^2}$

For the initial value $y(0) = 1$ the exact solution is $y(x) = \dfrac{t + \sqrt{4 - t^2}}{2}$. The function f described in PostScript code is like (y ist still on the stack):

```
x               %% y x
mul             %% x*y
2 exch sub      %% 2-x*y
4 x dup mul     %% 2-x*y 4 x^2
sub             %% 2-x*y 4-x^2
div             %% (2-x*y)/(4-x^2)
```

The following example uses $y_0 = 1$.

```
\newcommand{\InitCond}{1}
\newcommand{\Func}{x mul 2 exch sub 4 x dup mul sub div}
\newcommand{\FuncAlg}{(2-x*y[0])/(4-x^2)}
```

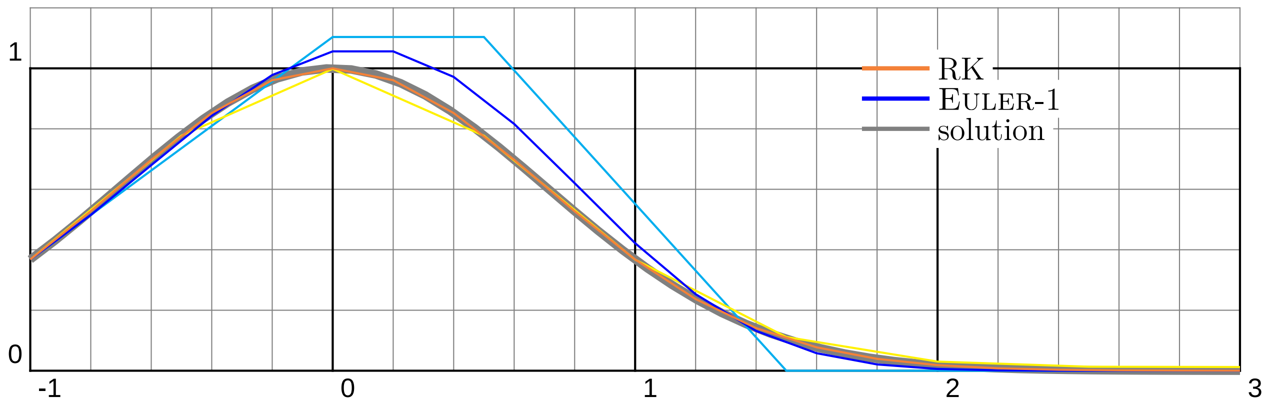


```
\psset{xunit=6.4, yunit=9.6, showpoints=false}
\begin{pspicture}(0,1)(2,1.7)  \psgrid[subgriddiv=5]
  { \psset{linewidth=4\pslinewidth,linecolor=lightgray}
  \psplot{0}{1.8}{x dup dup mul 4 exch sub sqrt add 2 div}
  \psplot{1.8}{2}{x dup dup mul 4 exch sub sqrt add 2 div} }
  \def\InitCond{1}
  \def\Func{x mul 2 exch sub 4 x dup mul sub div}
  \psplotDiffEqn[linecolor=magenta, plotpoints=20]{0}{1.9}{\InitCond}{\Func}
  \psplotDiffEqn[linecolor=blue, plotpoints=191]{0}{1.9}{\InitCond}{\Func}
  \psplotDiffEqn[linecolor=red, method=rk4, plotpoints=11,%
```

```
    algebraic=true]{0}{1.9}{\InitCond}{(2-x*y[0])/(4-x^2)}
  \psplotDiffEqn[linecolor=Orange, method=rk4, plotpoints=21,%
    algebraic=true]{0}{1.9}{\InitCond}{(2-x*y[0])/(4-x^2)}
  \psset{linewidth=4\pslinewidth}
  \rput*(0.3,1.6){\psline[linecolor=magenta](-.75cm,0)}\rput*[l](0.3,1.6){\small
   Euler order 1 $h=0{,}1$}
  \rput*(0.3,1.55){\psline[linecolor=blue](-.75cm,0)}\rput*[l](0.3,1.55){\small
   Euler order 1 $h=0{,}01$}
  \rput*(0.3,1.5){\psline[linecolor=Orange](-.75cm,0)}\rput*[l](0.3,1.5){\small RK
   order 4 $h=0{,}19$}
  \rput*(0.3,1.45){\psline[linecolor=red](-.75cm,0)}\rput*[l](0.3,1.45){\small RK
   order 4 $h=0{,}095$}
  \rput*(0.3,1.4){\psline[linecolor=lightgray](-.75cm,0)}\rput*[l](0.3,1.4){\small
   exactly}
\end{pspicture}
```

25.5.5 $y' = -2xy$

For $y(-1) = \frac{1}{e}$ we get $y(x) = e^{-x^2}$.



```
\psset{unit=4}
\begin{pspicture}(-1,0)(3,1.1)\psgrid
  \psplot[linewidth=4\pslinewidth,linecolor=gray]{-1}{3}{Euler x dup mul
     neg exp}
  \psset{plotpoints=9}
  \psplotDiffEqn[linecolor=cyan]{-1}{3}{1 Euler div}{x -2 mul mul}
  \psplotDiffEqn[linecolor=yellow, method=rk4]{-1}{3}{1 Euler div}{x -2
   mul mul}
  \psset{plotpoints=21}
  \psplotDiffEqn[linecolor=blue]{-1}{3}{1 Euler div}{x -2 mul mul}
  \psplotDiffEqn[linecolor=Orange, method=rk4]{-1}{3}{1 Euler div}{x -2
    mul mul}
  \psset{linewidth=2\pslinewidth}
  \rput*(2,1){\psline[linecolor=Orange](-0.25,0)}
  \rput*[l](2,1){RK}
  \rput*(2,.9){\psline[linecolor=blue](-0.25,0)}
  \rput*[l](2,.9){\textsc{Euler}-1}
  \rput*(2,.8){\psline[linecolor=gray](-0.25,0)}
  \rput*[l](2,.8){solution}
\end{pspicture}
```

25.5.6 Spirale of Cornu

The integrals of Fresnel :

$$x = \int_0^t \cos\frac{\pi t^2}{2}\mathrm{d}t \tag{2}$$

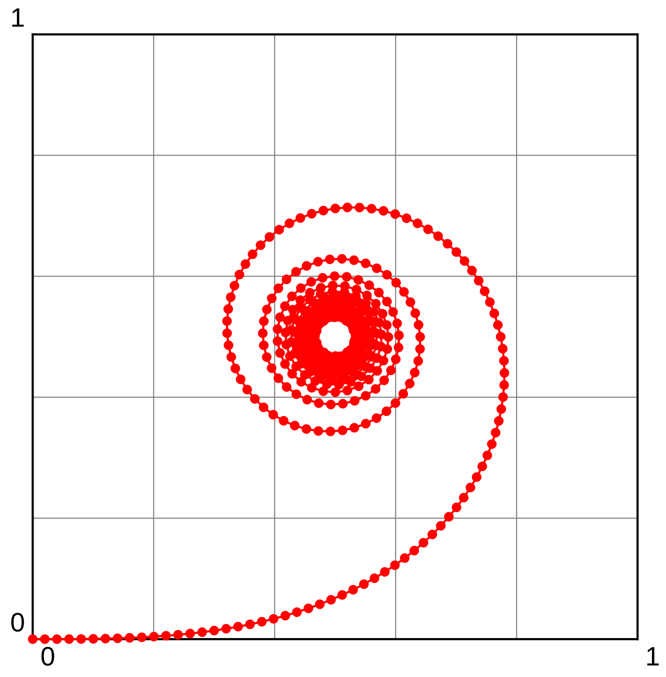
$$y = \int_0^t \sin\frac{\pi t^2}{2}\mathrm{d}t \tag{3}$$

with

$$\dot{x} = \cos\frac{\pi t^2}{2} \tag{4}$$

$$\dot{y} = \sin\frac{\pi t^2}{2} \tag{5}$$

```
\psset{unit=8}
\begin{pspicture}(1,1)\psgrid[subgriddiv=5]
  \psplotDiffEqn[whichabs=0,whichord=1,linecolor=red,method=rk4,
    algebraic,%
      plotpoints=500,showpoints=true]{0}{10}{0 0}{cos(Pi*x^2/2)|sin(Pi*x
        ^2/2)}
\end{pspicture}
```



25.5.7 Lotka-Volterra

The Lotka-Volterra model describes interactions between two species in an ecosystem, a predator and a prey. This represents our first multi-species model. Since we are considering two species, the model will involve two equations, one which describes how the prey population changes and the second which describes how the predator population changes.

For concreteness let us assume that the prey in our model are rabbits, and that the predators are foxes. If we let $R(t)$ and $F(t)$ represent the number of rabbits and foxes, respectively, that are alive at time t, then the Lotka-Volterra model is:

$$\dot{R} = a \cdot R - b \cdot R \cdot F \tag{6}$$
$$\dot{F} = e \cdot b \cdot R \cdot F - c \cdot F \tag{7}$$

where the parameters are defined by:

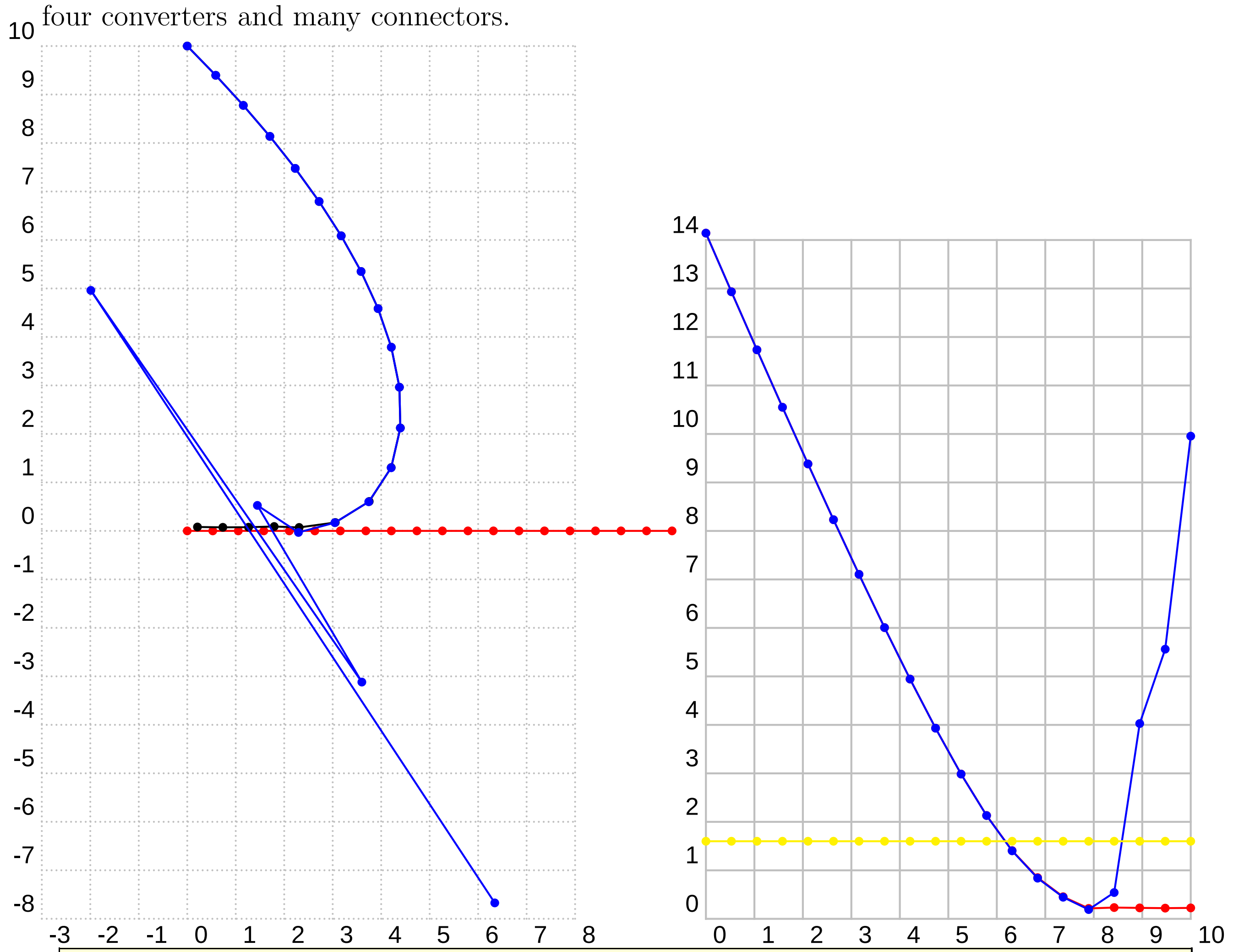
a is the natural growth rate of rabbits in the absence of predation,

c is the natural death rate of foxes in the absence of food (rabbits),

b is the death rate per encounter of rabbits due to predation,

e is the efficiency of turning predated rabbits into foxes.

The Stella model representing the Lotka-Volterra model will be slightly more complex than the single species models we've dealt with before. The main difference is that our model will have two stocks (reservoirs), one for each species. Each species will have its own birth and death rates. In addition, the Lotka-Volterra model involves four parameters rather than two. All told, the Stella representation of the Lotka-Volterra model will use two stocks, four flows, four converters and many connectors.

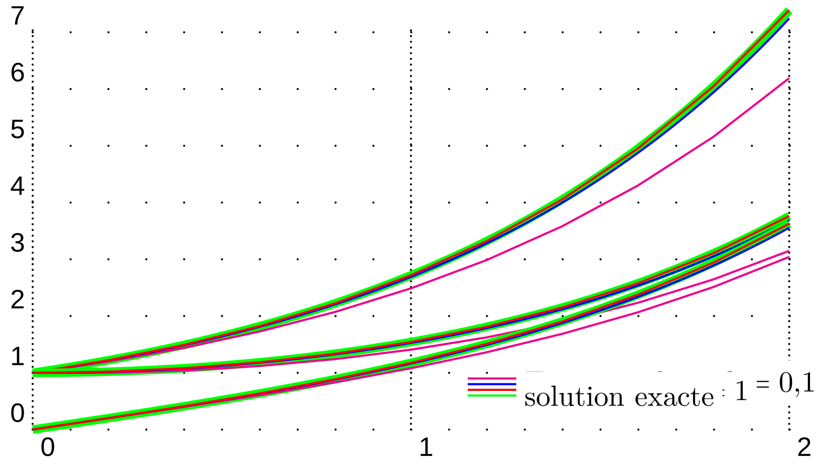


```
\def\InitCond{ 0 10 10}%% xa ya xl
\def\Faiglelapin{\Vaigle*(y[2]-y[0])/sqrt(y[1]^2+(y[2]-y[0])^2)|%
                  -\Vaigle*y[1]/sqrt(y[1]^2+(y[2]-y[0])^2)|%
                  -\Vlapin}
\def\Vlapin{1}  \def\Vaigle{1.6}
```

```
\psset{unit=.7,subgriddiv=0,gridcolor=lightgray,method=adams,algebraic,%
   plotpoints=20,showpoints=true}
\begin{pspicture}(-3,-8)(5,10)\psgrid[griddots=10]
 \psplotDiffEqn[plotfuncy=pop 0,whichabs=2,linecolor=red]{0}{10}{\
   InitCond}{\Faiglelapin}
 \psplotDiffEqn[whichabs=0,whichord=1,linecolor=black,method=rk
   4]{0}{10}{\InitCond}{\Faiglelapin}
  \psplotDiffEqn[whichabs=0,whichord=1,linecolor=blue]{0}{10}{\InitCond
    }{\Faiglelapin}
\end{pspicture}\hfill
\begin{pspicture}(10,12)\psgrid
 \psplotDiffEqn[plotfuncy=dup 1 get dup mul exch dup 0 get exch 2 get
   sub dup
     mul add sqrt,linecolor=red,method=rk4]{0}{10}{\InitCond}{\
       Faiglelapin}
 \psplotDiffEqn[plotfuncy=dup 1 get dup mul exch dup 0 get exch 2 get
   sub dup
     mul add sqrt,linecolor=blue]{0}{10}{\InitCond}{\Faiglelapin}
 \psplotDiffEqn[plotfuncy=pop Func aload pop pop dup mul exch dup mul
   add sqrt,
     linecolor=yellow]{0}{10}{\InitCond}{\Faiglelapin}
\end{pspicture}
```

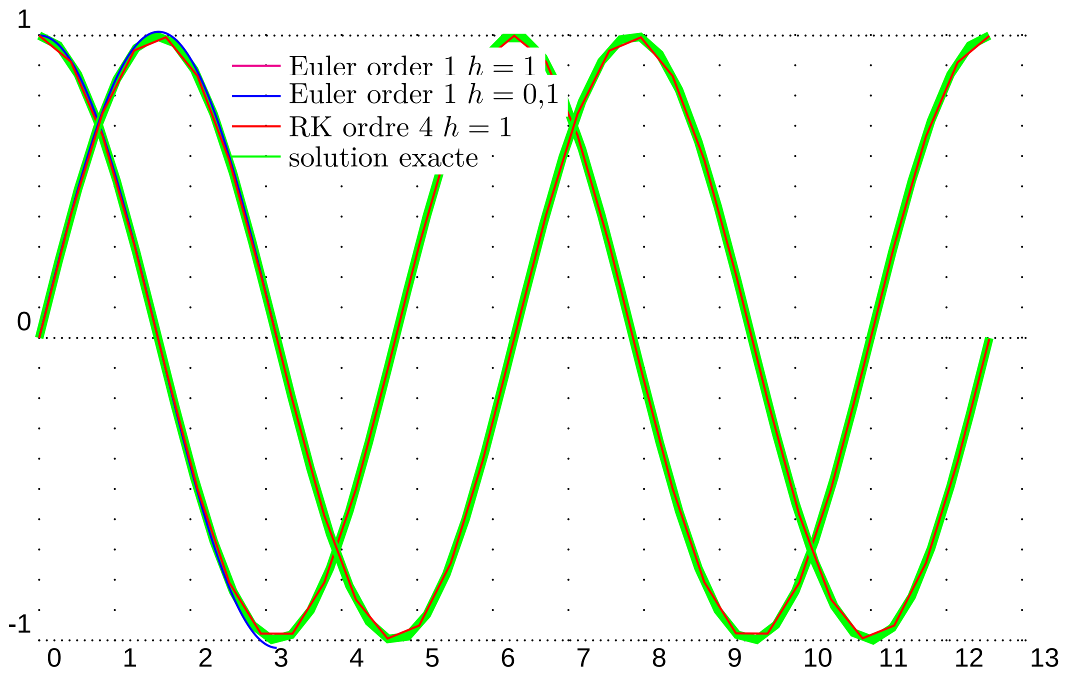
25.5.8 $y'' = y$

Beginning with the initial equation $y(x) = Ae^x + Be^{-x}$ we get the hyperbolic trigonometrical functions.



```latex
\def\Funct{exch}   \psset{xunit=5cm, yunit=0.75cm}
\begin{pspicture}(0,-0.25)(2,7)\psgrid[subgriddiv=1,griddots=10]
 \psplot[linewidth=4\pslinewidth, linecolor=green]{0}{2}{Euler x exp}  %%e^x
 \psplotDiffEqn[linecolor=magenta, plotpoints=11]{0}{2}{1 1}{\Funct}
 \psplotDiffEqn[linecolor=blue, plotpoints=101]{0}{2}{1 1}{\Funct}
 \psplotDiffEqn[linecolor=red, method=rk4, plotpoints=11]{0}{2}{1 1}{\Funct}
 \psplot[linewidth=4\pslinewidth, linecolor=green]{0}{2}{Euler dup x exp   %%ch(x)
    exch x neg exp add 2 div}
 \psplotDiffEqn[linecolor=magenta, plotpoints=11]{0}{2}{1 0}{\Funct}
 \psplotDiffEqn[linecolor=blue, plotpoints=101]{0}{2}{1 0}{\Funct}
 \psplotDiffEqn[linecolor=red, method=rk4, plotpoints=11]{0}{2}{1 0}{\Funct}
 \psplot[linewidth=4\pslinewidth, linecolor=green]{0}{2}{Euler dup x exp
     exch x neg exp sub 2 div}  %%sh(x)
 \psplotDiffEqn[linecolor=magenta, plotpoints=11]{0}{2}{0 1}{\Funct}
 \psplotDiffEqn[linecolor=blue, plotpoints=101]{0}{2}{0 1}{\Funct}
 \psplotDiffEqn[linecolor=red, method=rk4, plotpoints=11]{0}{2}{0 1}{\Funct}
 \rput*(1.3,.9){\psline[linecolor=magenta](-.75cm,0)}\rput*[l](1.3,.9){\small\textsc{Euler} ordre 1 $h=1$}
 \rput*(1.3,.8){\psline[linecolor=blue](-.75cm,0)}\rput*[l](1.3,.8){\small\textsc{Euler} ordre 1 $h=0{,}1$}
 \rput*(1.3,.7){\psline[linecolor=red](-.75cm,0)}\rput*[l](1.3,.7){\small RK ordre 4 $h=1$}
 \rput*(1.3,.6){\psline[linecolor=green](-.75cm,0)}\rput*[l](1.3,.6){\small solution exacte}
\end{pspicture}
```

25.5.9 $y'' = -y$



```
\def\Funct{exch neg}
\psset{xunit=1, yunit=4}
\def\quatrepi{12.5663706144}%%4pi=12.5663706144
\begin{pspicture}(0,-1.25)(\quatrepi,1.25)\psgrid[subgriddiv=1,griddots
  =10]
 \psplot[linewidth=4\pslinewidth,linecolor=green]{0}{\quatrepi}{x
   RadtoDeg cos}%%cos(x)
 \psplotDiffEqn[linecolor=blue, plotpoints=201]{0}{3.1415926}{1 0}{\
   Funct}
 \psplotDiffEqn[linecolor=red, method=rk4, plotpoints=31]{0}{\quatrepi
   }{1 0}{\Funct}
 \psplot[linewidth=4\pslinewidth,linecolor=green]{0}{\quatrepi}{x
   RadtoDeg sin}  %%sin(x)
 \psplotDiffEqn[linecolor=blue,plotpoints=201]{0}{3.1415926}{0 1}{\Funct
   }
 \psplotDiffEqn[linecolor=red,method=rk4, plotpoints=31]{0}{\quatrepi}{0
    1}{\Funct}
 \rput*(3.3,.9){\psline[linecolor=magenta](-.75cm,0)}\rput*[l](3.3,.9){\
   small Euler order 1 $h=1$}
```

```
\rput*(3.3,.8){\psline[linecolor=blue](-.75cm,0)}\rput*[l](3.3,.8){\
  small Euler order 1 $h=0{,}1$}
\rput*(3.3,.7){\psline[linecolor=red](-.75cm,0)}\rput*[l](3.3,.7){\
  small RK ordre 4 $h=1$}
\rput*(3.3,.6){\psline[linecolor=green](-.75cm,0)}\rput*[l](3.3,.6){\
  small solution exacte}
\end{pspicture}
```

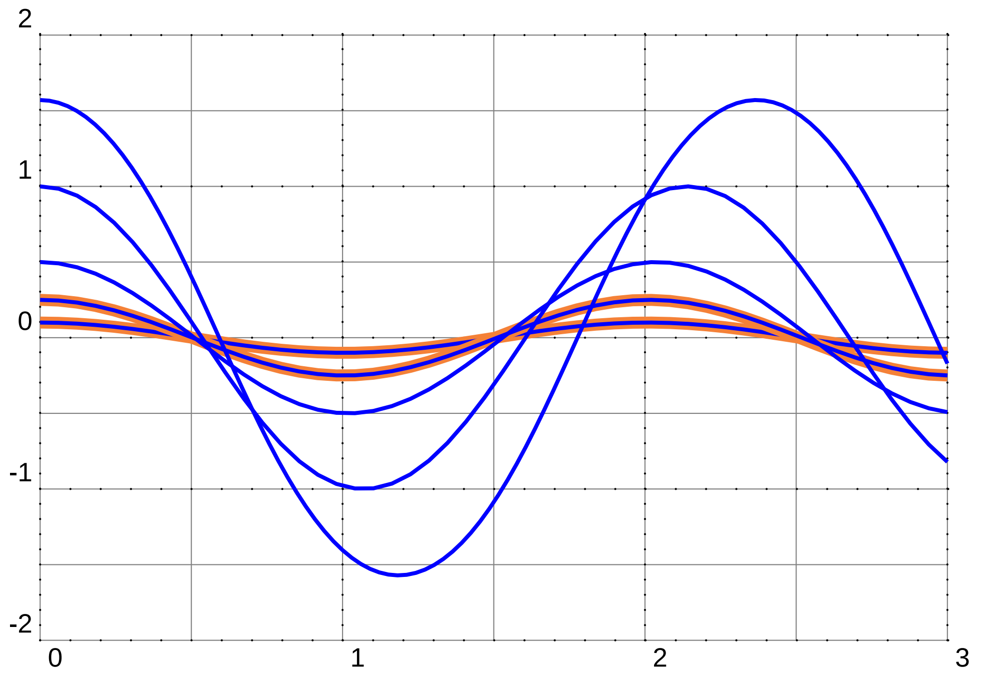
25.5.10 The mechanical pendulum: $y'' = -\frac{g}{l}\sin(y)$

Pour des faibles oscillations $\sin(y) \simeq y$:

$$y(x) = y_0 \cos\left(\sqrt{\frac{g}{l}}x\right)$$

The function f is writen in PostScript code:

```
exch RadtoDeg sin -9.8 mul %% y' -gsin(y)
```

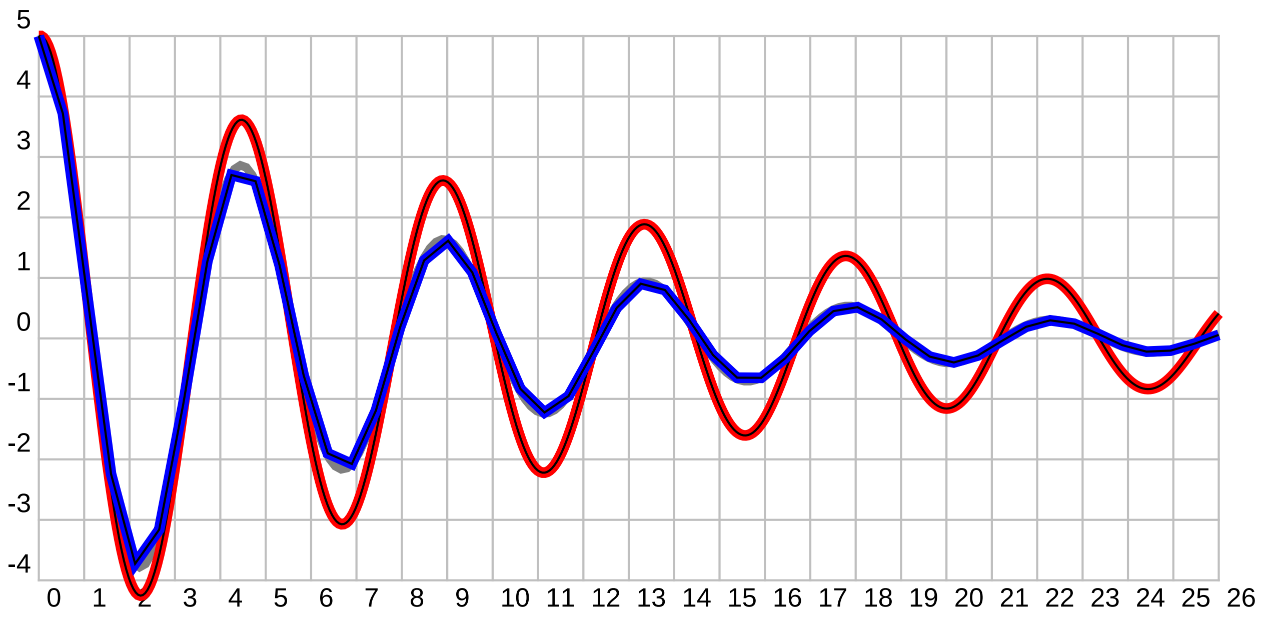


```
\def\Func{y[1]|-9.8*sin(y[0])}
\psset{yunit=2,xunit=4,algebraic=true,linewidth=1.5pt}
\begin{pspicture}(0,-2.25)(3,2.25)\psgrid[subgriddiv=2,griddots=10]
  \psplot[linewidth=3\pslinewidth, linecolor=Orange]{0}{3}{.1*cos(sqrt
    (9.8)*x)}
  \psset{method=rk4,plotpoints=50,linecolor=blue}
  \psplotDiffEqn{0}{3}{.1 0}{\Func}
  \psplot[linewidth=3\pslinewidth,linecolor=Orange]{0}{3}{.25*cos(sqrt
    (9.8)*x)}
  \psplotDiffEqn{0}{3}{.25 0}{\Func}
  \psplotDiffEqn{0}{3}{.5 0}{\Func}
  \psplotDiffEqn{0}{3}{1 0}{\Func}
  \psplotDiffEqn[plotpoints=100]{0}{3}{Pi 2 div 0}{\Func}
\end{pspicture}
```

25.5.11 $y'' = -\frac{y'}{4} - 2y$

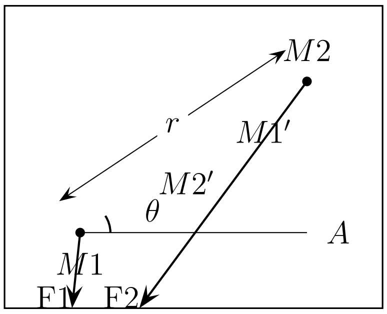
Pour $y_0 = 5$ et $y'_0 = 0$ la solution est :

$$5e^{-\frac{x}{8}}\left(\cos\left(\omega x\right) + \frac{\sin(\omega x)}{8\omega}\right) \quad \text{avec } \omega = \frac{\sqrt{127}}{8}$$



```
\psset{xunit=.6,yunit=0.8,plotpoints=500}
\begin{pspicture}(0,-4.25)(26,5.25)
  \psgrid[subgriddiv=0,gridcolor=lightgray,linewidth=1.5pt]
  \psplot[plotpoints=200,linewidth=4\pslinewidth,linecolor=gray]{0}{26}{%
     Euler x -8 div exp x 127 sqrt 8 div mul RadtoDeg dup cos 5 mul exch
        sin 127 sqrt div 5 mul add mul}
  \psplotDiffEqn[linecolor=red,linewidth=5\pslinewidth]{0}{26}{5 0}
     {dup 3 1 roll -4 div exch 2 mul sub}
  \psplotDiffEqn[linecolor=black,algebraic]{0}{26}{5 0} {y[1]|-y[1]/4-2*
    y[0]}
  \psset{method=rk4, plotpoints=50}
  \psplotDiffEqn[linecolor=blue,linewidth=5\pslinewidth]{0}{26}{5 0}{%
      dup 3 1 roll -4 div exch 2 mul sub}
  \psplotDiffEqn[linecolor=black,algebraic=true]{0}{26}{5 0}{y[1]|-y
    [1]/4-2*y[0]}
\end{pspicture}
```

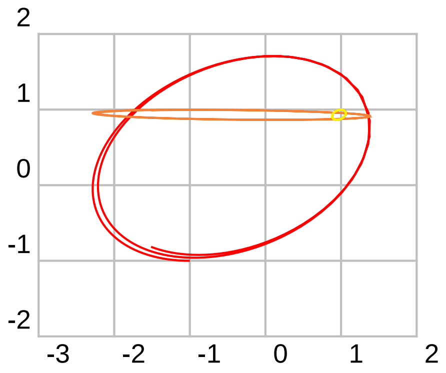
25.5.12 Gravitation example of second order



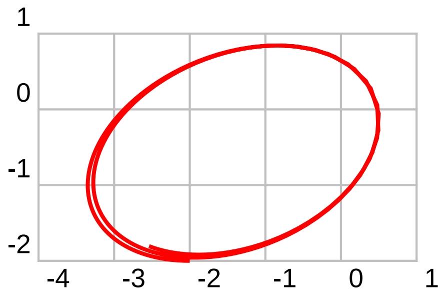
$$\begin{cases} x_1'' = \dfrac{M_2}{r^2}\cos(\theta) \\ y_1'' = \dfrac{M_2}{r^2}\sin(\theta) \\ x_2'' = \dfrac{M_1}{r^2}\cos(\theta) \\ y_2'' = \dfrac{M_1}{r^2}\sin(\theta) \end{cases} \text{avec} \begin{cases} r^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2 \\ \cos(\theta) = \dfrac{(x_1 - x_2)}{r} \\ \sin(\theta) = \dfrac{(y_1 - y_2)}{r} \end{cases}$$

Table 3: PostScript code for the gravitation examples

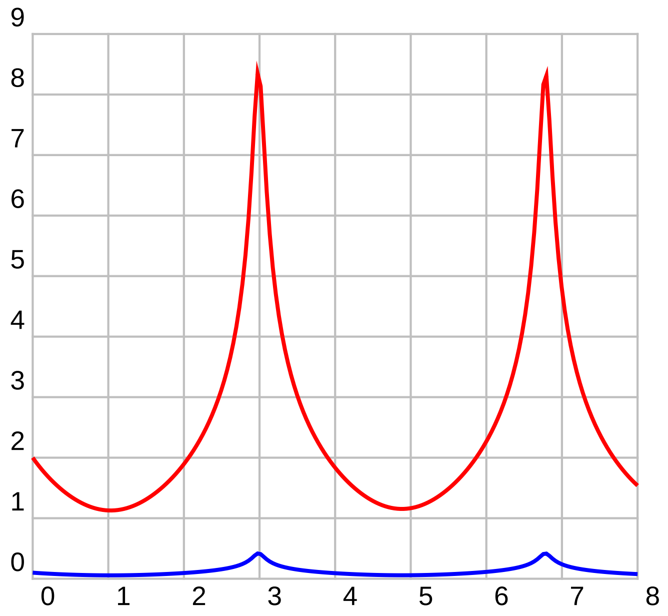
	`x1 y1 x'1 y'1 x2 y2 x'2 y'2`
`/yp2 exch def /xp2 exch def /ay2 exch def /ax2 exch def`	`mise en variables`
`/yp1 exch def /xp1 exch def /ay1 exch def /ax1 exch def`	`mise en variables`
`/ro2 ax2 ax1 sub dup mul ay2 ay1 sub dup mul add def`	`calcul de r*r`
`xp1 yp1`	
`ax2 ax1 sub ro2 sqrt div ro2 div`	`calcul de x''1`
`ay2 ay1 sub ro2 sqrt div ro2 div`	`calcul de y''1`
`xp2 yp2`	
`3 index -20 mul`	`calcul de x''2=-20x''1`
`3 index -20 mul`	`calcul de y''2=-20y''1`



```latex
\begin{pspicture}(-3,-2.5)(2,2.25)
 \psgrid[subgriddiv=0,gridcolor=lightgray,linewidth=1.5pt]
 \psset{method=rk4,plotpoints=200,whichord=1}
 \psplotDiffEqn[whichabs=0,linecolor=blue]{0}{8}{\InitCond}{\Grav}
 \psplotDiffEqn[whichabs=4,whichord=5,linecolor=red]{0}{8}{\InitCond}{\Grav}
 \psplotDiffEqn[whichabs=4,linecolor=Orange,algebraic]{0}{8}{\InitCond}{\GravAlg}
 \psplotDiffEqn[whichabs=0,whichord=1,linecolor=yellow,algebraic]{0}{8}{\InitCond}{\
   GravAlg}
\end{pspicture}
```



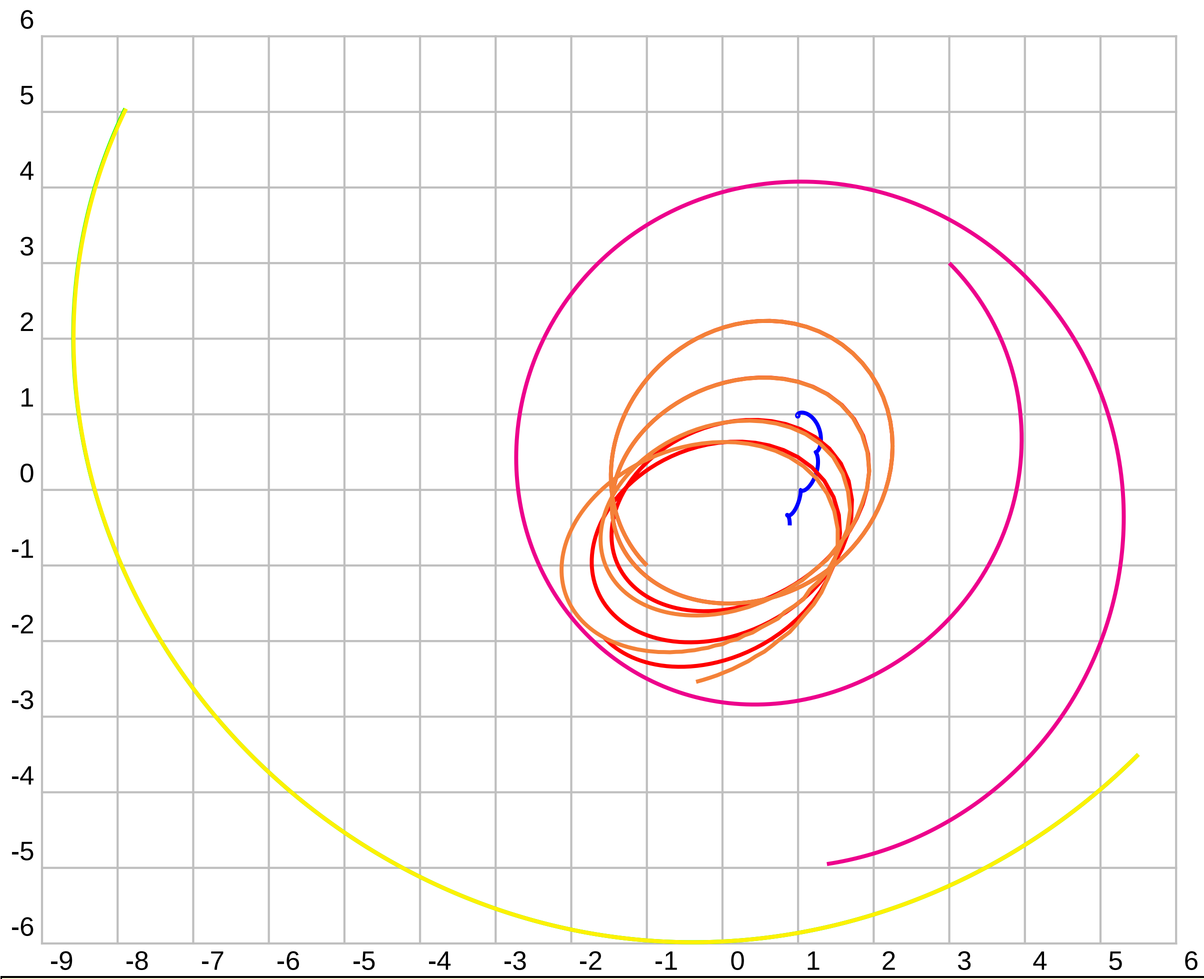
```
\begin{pspicture}(-4,-2.5)(1,1.25)
 \psgrid[subgriddiv=0,gridcolor=lightgray]
 \psplotDiffEqn[linecolor=red,method=rk4,plotpoints=200,linewidth=1.5pt,%
   plotfuncx=y dup 4 get exch 0 get sub,
   plotfuncy=dup 5 get exch 1 get sub ]{0}{8}{\InitCond}{\Grav}
\end{pspicture}
```



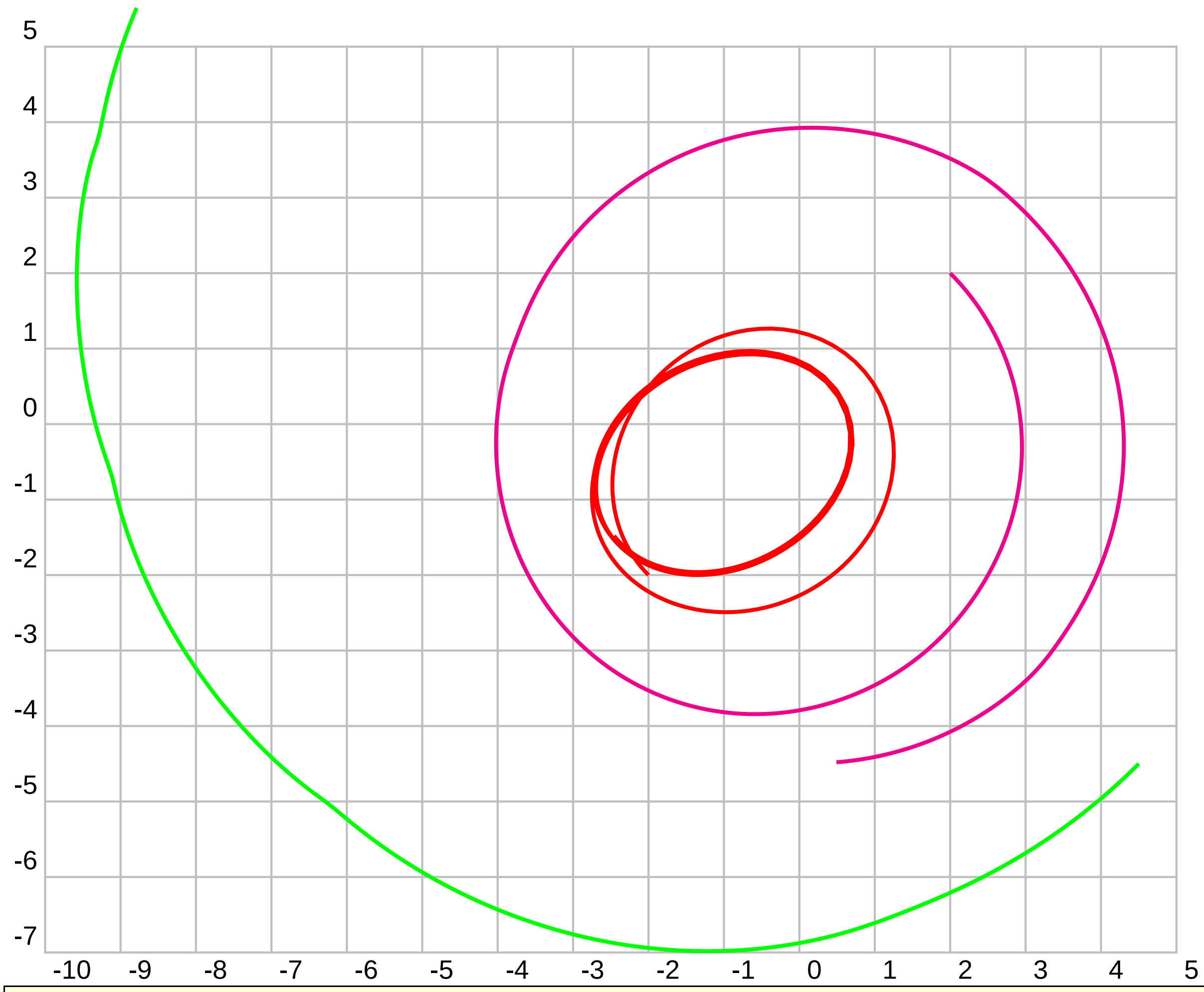
```
\begin{pspicture}(0,-0.5)(8,8)
  \psset{yunit=0.8,method=rk4,plotpoints=200,linewidth=1.5pt}
  \psgrid[subgriddiv=0,gridcolor=lightgray](8,9)
  \psplotDiffEqn[linecolor=red,plotfuncy=dup 6 get dup mul exch 7 get dup mul add sqrt
    ]{0}{8}{\InitCond}{\Grav}
  \psplotDiffEqn[linecolor=blue,plotfuncy=dup 2 get dup mul exch 3 get dup mul add sqrt
    ]{0}{8}{\InitCond}{\Grav}
\end{pspicture}
```

25.5.13 Gravitation: two stars and more

x1 y1 x'1 y'1 x2 y2 x'2 y'2 x3 y3 x'3 y'3 x4 y4 x'4 y'4



```
\begin{pspicture}(-9,-6.25)(6,6)
 \psgrid[subgriddiv=0,gridcolor=lightgray](-9,-6)(6,6)
 \psset{buildvector=true,method=rk4,linewidth=1.5pt}
 \psplotDiffEqn[whichabs=0,whichord=1,linecolor=blue]{0}{\tMAX}{\InitCond}{\Test}
 \psplotDiffEqn[whichabs=4,whichord=5,linecolor=red]{0}{\tMAX}{\InitCond}{\Test}
 \psplotDiffEqn[whichabs=4,whichord=5,linecolor=Orange,method=adams]{0}{\tMAX}{\
   InitCond}{\Test}
 \psplotDiffEqn[whichabs=8,whichord=9,linecolor=magenta]{0}{\tMAX}{\InitCond}{\Test}
 \psplotDiffEqn[whichabs=12,whichord=13,linecolor=green]{0}{\tMAX}{\InitCond}{\Test}
 \psplotDiffEqn[whichabs=12,whichord=13,linecolor=yellow,method=adams]{0}{\tMAX}{\
   InitCond}{\Test}
\end{pspicture}
```



```
\begin{pspicture}(-10,-7.25)(5,5)
  \psgrid[subgriddiv=0,gridcolor=lightgray](-10,-7)(5,5)
  \psset{buildvector=true,method=rk4,linewidth=1.5pt}
  \psplotDiffEqn[linecolor=red,plotfuncx=y dup 4 get exch 0 get sub ,
     plotfuncy=dup 5 get exch 1 get sub ]{0}{\tMAX}{\InitCond}{\Test}
  \psplotDiffEqn[linecolor=magenta,plotfuncx=y dup 8 get exch 0 get sub ,
     plotfuncy=dup 9 get exch 1 get sub ]{0}{\tMAX}{\InitCond}{\Test}
  \psplotDiffEqn[linecolor=green,plotfuncx=y dup 12 get exch 0 get sub ,
     plotfuncy=dup 13 get exch 1 get sub ]{0}{\tMAX}{\InitCond}{\Test}
\end{pspicture}
```

25.6 \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`.

26 Credits

Hendri Adriaens | Ulrich Dirr | Hubert Gäßlein | Denis Girou | Peter Hutnick | Christophe Jorssen | Manuel Luque | Jens-Uwe Morawski | Tobias Nähring | Rolf Niepraschk | Dominique Rodriguez | Arnaud Schmittbuhl | Timothy Van Zandt

References

[1] Hendri Adriaens. xkeyval package. `CTAN:/macros/latex/contrib/xkeyval`, 2004.

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

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

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

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

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

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

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

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

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

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

27 Change log

See file Changes