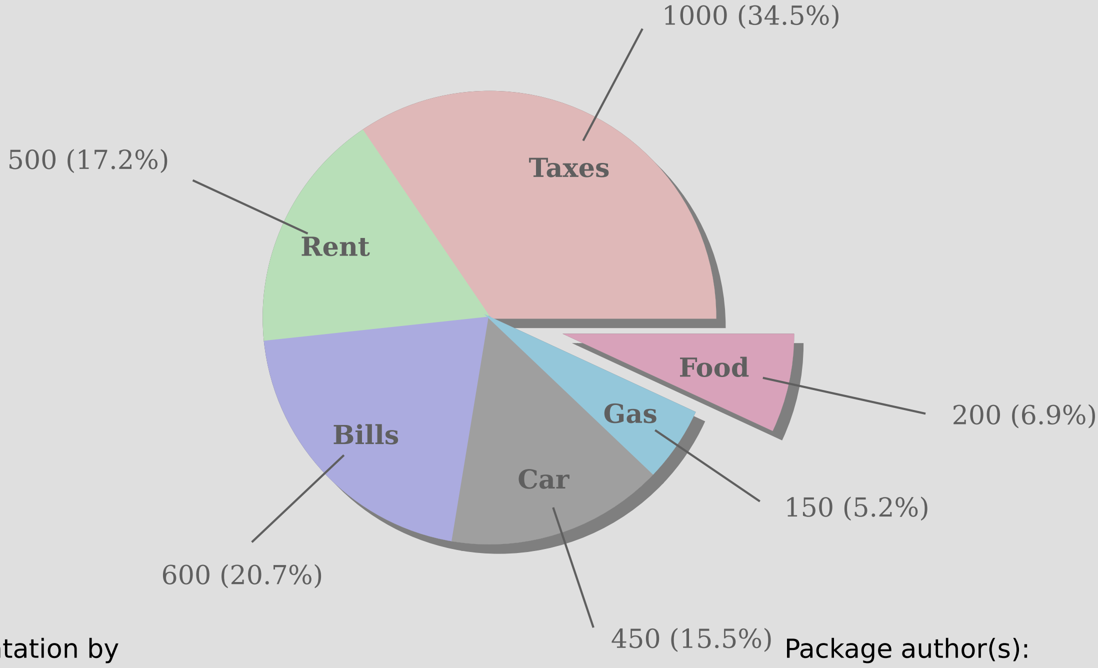
PSTricks

pstricks-add
additionals Macros for pstricks
v.3.31

April 29, 2009



Documentation by
Herbert Voß

Package author(s):
Dominique Rodriguez
Herbert Voß

This version of `pstricks-add` needs `pstricks.tex` version >1.04 from June 2004, otherwise the additional macros may not work as expected. 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 the **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 use the old keyval interface are loaded **before** the `xkeyval`.[1]
- the option `tickstyle` from `pst-plot` is no longer supported; use `ticksize` instead.
- the option `xyLabel` is no longer supported; use the option `labelFontSize` instead.
- if `pstricks-add` is loaded together with the package `pst-func` then `InsideArrow` of the `\psbezier` macro doesn't work!

Thanks to: Hendri Adriaens; Stefano Baroni; Martin Chicoine; Gerry Coombes; Ulrich Dirr; Christophe Fourey; Hubert Gäßlein; Jürgen Gilg; Denis Girou; Peter Hutnick; Christophe Jorssen; Uwe Kern; Manuel Luque; Jens-Uwe Morawski; Tobias Nähring; Rolf Niepraschk; Alan Ristow; Christine Römer; Arnaud Schmittbuhl; Timothy Van Zandt

Contents

Part I. pstricks

1. Numeric functions

All macros have a @ in their name, because they are only for internal use, but it is no problem to use them like other macros. One can define another name without a @:

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

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

1.1. `\pst@divide`

`pstricks` itself has its own divide macro, called `\pst@divide`, which can divide two lengths and save 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 function in documents requires a setting inside the `makeatletter` and `makeatother` environment. It makes some sense to define a new macroname in the preamble and use it throughout, 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 a 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@mindim

\pst@mindim{*dimension*}{*dimension*}{*result as dimension register*}

967.39369pt
123.0pt

```
\newdimen\minDim
\makeatletter
\pst@mindim{34cm}{1234pt}\minDim \the\minDim\\
\pst@mindim{34cm}{123pt}\minDim \the\minDim
\makeatother
```

1.6. \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
```

1.7. \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
```

1.8. \pst@int

\pst@int{*number*}{*result as a truncated integer*}

-34
234

```
\makeatletter
\pst@int{-34.0}\\
\pst@int{234.123}
\makeatother
```

1.9. \pstFPDiv

\pstFPDiv{*result as a truncated integer*}{*number*}{*number*}

-145
-0
7

```
\makeatletter
\pstFPDiv\Result{-3.405}{0.02345} \Result\\
\pstFPDiv\Result{0.02345}{-3.405} \Result\\
\pstFPDiv\Result{234.123}{33} \Result
\makeatother
```

1.10. \psGetSlope

\psGetSlope(x_1,y_1)(x_2,y_2)\⟨*macro*⟩

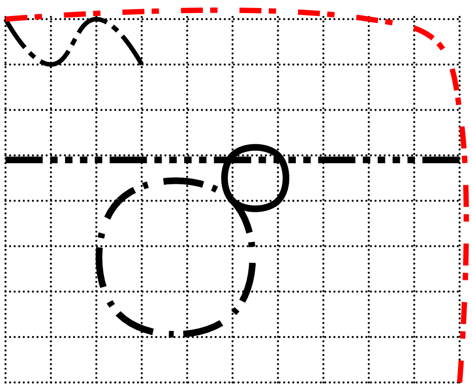
0.0
2.0
-0.2
0.00615

```
\psGetSlope(-2,1)(3,1)\SlopeVal \SlopeVal\\
\psGetSlope(-2,1)(-3,-1)\SlopeVal \SlopeVal\\
\psGetSlope(-2,0)(3,-1)\SlopeVal \SlopeVal\\
\psGetSlope(-2111,-12)(3,1)\SlopeVal \SlopeVal
```

2. Dashed Lines

Tobias Nähring has implemented an enhanced feature for dashed lines. The number of arguments is no longer limited.

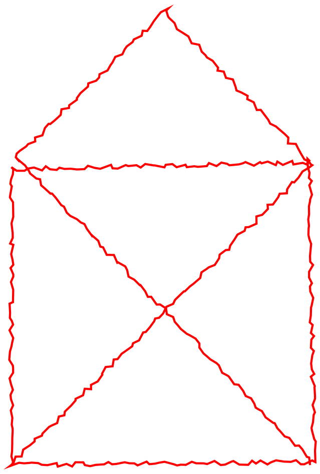
dash=value1 `unit` value2 `unit` ...



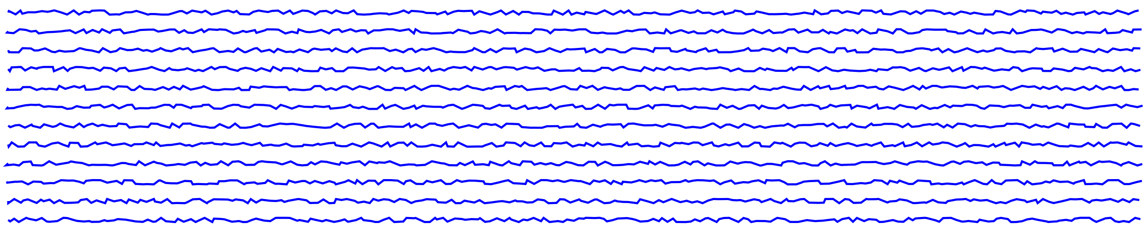
```
\psset{linewidth=2.5pt,unit=0.6}
\begin{pspicture}(-5,-4)(5,4)
 \psgrid[subgriddiv=0,griddots=10,
   gridlabels=0pt]
  \psset{linestyle=dashed}
  \pscurve[dash=5mm 1mm 1mm 1mm,linewidth
    =0.1](-5,4)(-4,3)(-3,4)(-2,3)
  \psline[dash=5mm 1mm 1mm 1mm 1mm 1mm 1mm
    1mm 1mm 1mm](-5,0.9)(5,0.9)
  \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. "‘Handmade"’ lines :-)

\pslineByHand [Options] $(()(x_1,y_1))(()(x_2,y_2))(()(x_3,y_3))\ldots$



```
\begin{pspicture}(4,6)
\psset{unit=2cm}
 \pslineByHand[linecolor=red](0,0)(0,2)
   (2,2)(2,0)(0,0)(2,2)(1,3)(0,2)(2,0)
\end{pspicture}
```

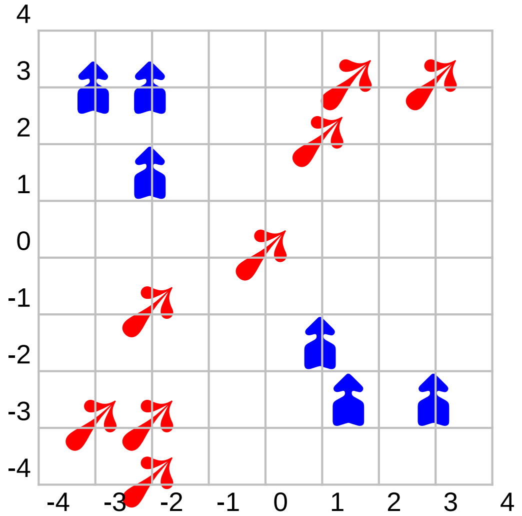


```
\begin{pspicture}(\linewidth,3)
\multido{\rA=0.00+0.25}{12}{\pslineByHand[linecolor=blue](0,\rA)(\linewidth
  ,\rA)}
\end{pspicture}
```

4. \rmultiput: a multiple \rput

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

\rmultiput* [Options] {*any material*}$(x_1,y_1)(x_2,y_2)\ldots(x_n,y_n)$

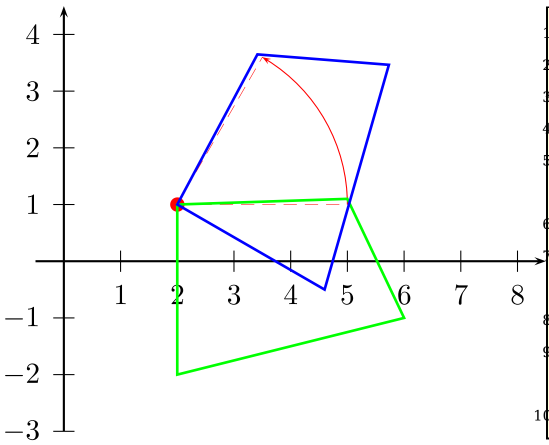



```
\psset{unit=0.75}
\begin{pspicture}(-4,-4)(4,4)
\rmultiput[rot=45]{\red\psscalebox{3}{\
  ding{250}}}%
   (-2,-4)(-2,-3)(-3,-3)(-2,-1)(0,0)(1,2)
      (1.5,3)(3,3)
\rmultiput[rot=90,ref=lC]{\blue\
  psscalebox{2}{\ding{253}}}%
   (-2,2.5)(-2,2.5)(-3,2.5)(-2,1)(1,-2)
      (1.5,-3)(3,-3)
\psgrid[subgriddiv=0,gridcolor=lightgray]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

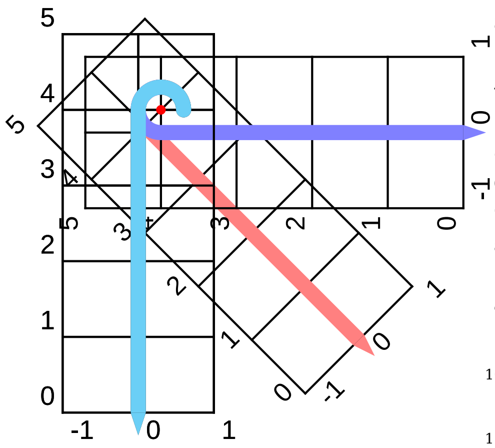
5. \psrotate: Rotating objects

`\rput` also has an optional argument for rotating objects, but it always depends on 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.

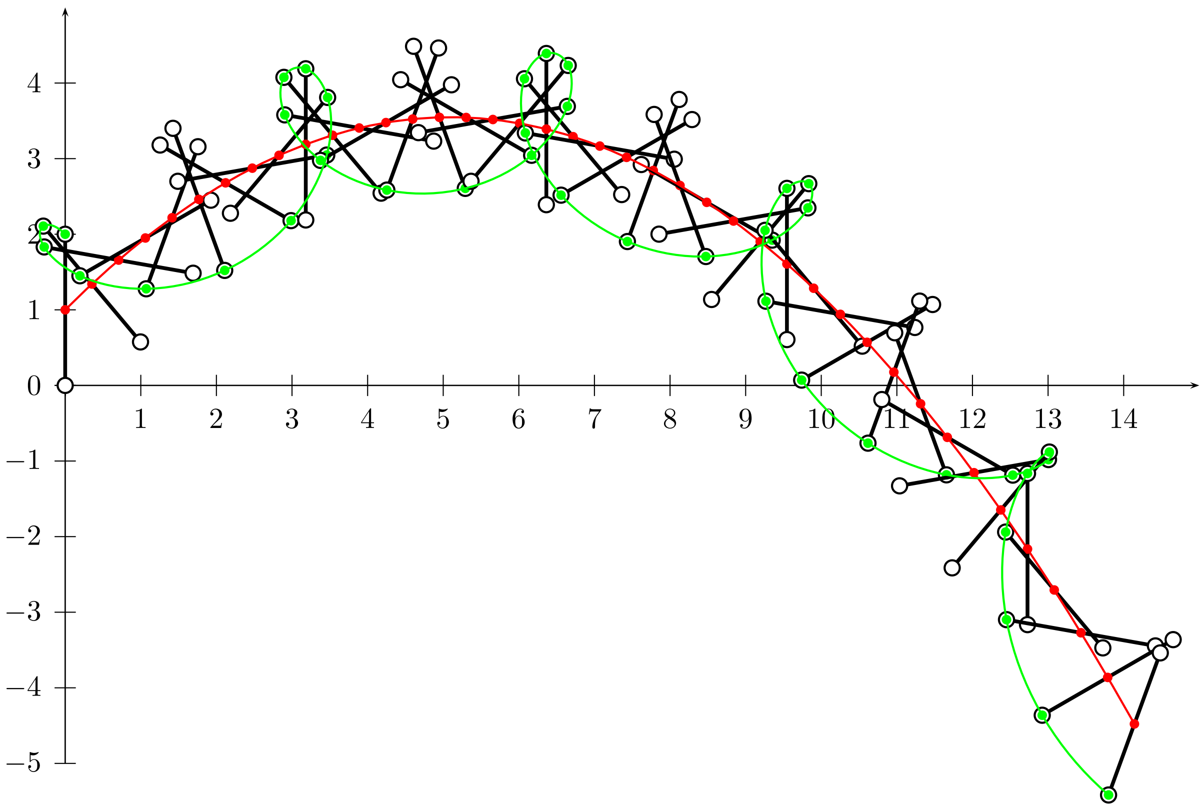
`\psrotate [Options] (`x,y`){`*rot angle*`}{`*object*`}`




```
\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,linewidth=1pt
    ](2,1)(5,1.1)(6,-1)(2,-2)
  \psrotate(2,1){60}{%
    \pspolygon[linecolor=blue,linewidth=1pt
      ](2,1)(5,1.1)(6,-1)(2,-2)}
\end{pspicture}
```



```
\def\canne{% Idea by Manuel Luque
  \psgrid[subgriddiv=0](-1,0)(1,5)
  \pscustom[linewidth=2mm]{\psline(0,4)\
    psarcn(0.3,4){0.3}{180}{360}}%
  \pscircle*(0.6,4){0.1}\pstriangle*(0,0)
    (0.2,-0.3)}
\def\Object{}
\begin{pspicture}(-1,-1)(3,6)
  \canne
  \psrotate(0.3,4){45}{\psset{linecolor=red
    !50}\canne}
  \psrotate(0.3,4){90}{\psset{linecolor=
    blue!50}\canne}
  \psrotate(0.3,4){360}{\psset{linecolor=
    cyan!50}\canne}
  \psdot[linecolor=red](0.3,4)
\end{pspicture}
```

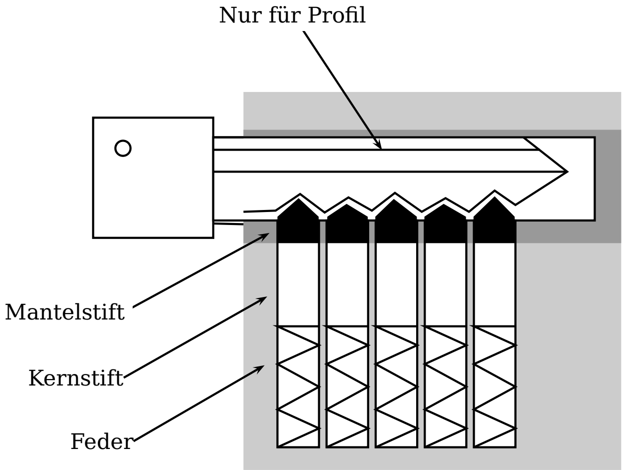



```
\def\majorette{\psline[linewidth=0.5mm](0,2)% Idea by Manuel Luque
           \pscircle[fillstyle=solid]{0.1}
           \pscircle[fillstyle=solid](0,2){0.1}}
\begin{pspicture}(0,-6)(15,5)
 \psaxes[linewidth=0.5pt]{->}(0,0)(0,-5)(15,5)
 \pstVerb{/V0 10 def /Alpha 45 def}% vitesse initiale, angle de lancement
 \multido{\nT=0.0+0.05,\iA=0+40}{41}{%
   \pstVerb{/nT \nT\space def}%
   \rput(!V0 Alpha cos mul nT mul -9.81 2 div nT dup mul mul V0 Alpha sin
     mul nT mul add){%
     \psrotate(0,1){\iA}{\majorette\psdot[linecolor=red](0,1)\psdot[
       linecolor=green](0,2)}}}
 \parametricplot[linecolor=red]{0}{2}{% trajectoire du milieu
    V0 Alpha cos mul t mul -9.81 2 div t dup mul mul V0 Alpha sin mul t mul
      add 1 add}
 \parametricplot[linecolor=green,plotpoints=360]{0}{2}{% d'une extremite
    V0 Alpha cos mul t mul 800 t mul sin sub % x(t)
    -9.81 2 div t dup mul mul V0 Alpha sin mul t mul add 1 add 800 t mul cos
      add }%y(t)
\end{pspicture}
```

6. \psComment: comments to a graphic

`\psComment` `*` `[Options]` `{arrows}` $(x_0,y_0)(x_1,y_1)$`{Text}` `[line macro]`

By default the macro uses the `\ncline` macro to draw a line from the first to the second point. With the second additional argument one can use another macro for the line.




```
\SpecialCoor\newpsstyle{weiss}{fillstyle=solid,fillcolor=white}
\footnotesize\psset{unit=0.5cm,dimen=middle}
\begin{pspicture}(-12,-4)(6,10)
\psframe*[linecolor=black!20](-5,-3)(5,7) \psframe*[linecolor=black!40](-5,3)(5,6)
\pscircle(-8.19,5.51){0.2}
\psframe[fillcolor=white,fillstyle=solid](-5.8,3.6)(4.3,5.8)
\psframe(-8.98,3.14)(-5.8,6.32)
\multido{\rA=-4.1+1.3}{5}{\rput(\rA,-2.4){\psframe[style=weiss](1.1,6)
  \psline(0,0)(1.1,0.5)(0,1)(1.1,1.6)(0,2.2)(1.1,2.7)(0,3.2)(1.1,3.2)}}
\pspolygon*(-4.1,3.7)(-4.1,3)(-3,3)(-3.01,3.7)(-3.54,4.19)
\pspolygon*(1.09,3.7)(1.1,3)(2.2,3)(2.18,3.7)(1.65,4.24)
\pspolygon*(-2.78,3.7)(-2.8,3)(-1.7,3)(-1.71,3.7)(-2.27,4.04)
\pspolygon*(-1.51,3.7)(-1.5,3)(-0.4,3)(-0.41,3.7)(-1.02,4.17)
\pspolygon*(-0.21,3.7)(-0.2,3)(0.9,3)(0.89,3.7)(0.3,4.04)
\psline(-5,3.83)(-4.15,3.86)(-3.5,4.3)(-2.85,3.81)(-2.22,4.21)(-1.6,3.86)(-0.99,4.33)
      (-0.28,3.83)(0.35,4.19)(0.97,3.83)(1.65,4.39)(2.2,4.01)(3.57,4.89)(2.41,5.8)
  \psline(-5,5.8)(-5.78,5.8) \psline(-5.78,5.47)(2.85,5.47)
  \psline(-5.8,3.52)(-5,3.5) \psline(3.57,4.89)(-5.8,4.89)
  \psComment*[ref=r]{->}(-8.14,1.19)(-4.31,3.27){Mantelstift}
  \psComment*[ref=r]{->}(-8.17,-0.56)(-4.37,1.59){Kernstift}[\rput]
  \psComment*[ref=r]{->}(-7.91,-2.24)(-4.44,-0.23){Feder}[\rput]
  \psComment[npos=-0.1]{->}(-3.48,8.72)(-1.33,5.46){Nur f\"ur Profil}
\end{pspicture}
```

7. \psChart: a pie chart

`\psChart [Options] {`*comma separated value list*`}{`*comma separated value list*`}{`*radius*`}`

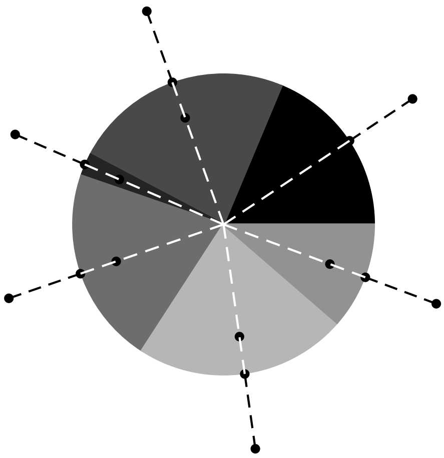
The special optional arguments for the `\psChart` macro are as follows:

name	*description*	*default*
`chartSep`	distance from the pie chart center to an outraged pie piece	`10pt`
`chartColor`	gray or colored pie (values are: `gray` or `color`)	`gray`
`userColor`	a comma separated list of user defined colors for the pie	`{}`

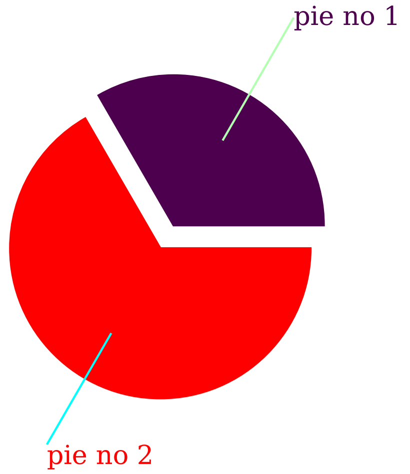
The first mandatory argument is the list of the values and may not be empty. The second one is a list of outraged pieces, numbered consecutively from 1 to up the total number of values. The list of user defined colors must be enclosed in braces!

The macro `\psChart` defines for every value three nodes at the half angle and in distances from 0.75, 1, and 1.25 times of the radius from the origin. The nodes are named as `psChartI?`, `psChart?`, and `psChartO?`, where ? is the number of the pie. The letter I leads to the inner node and the letter O to the outer node. The distance can be changed with the optional arguments `chartNodeI` and `chartNodeO` in the usual way with `\psset{chartNodeI=...,chartNodeO=...}`.

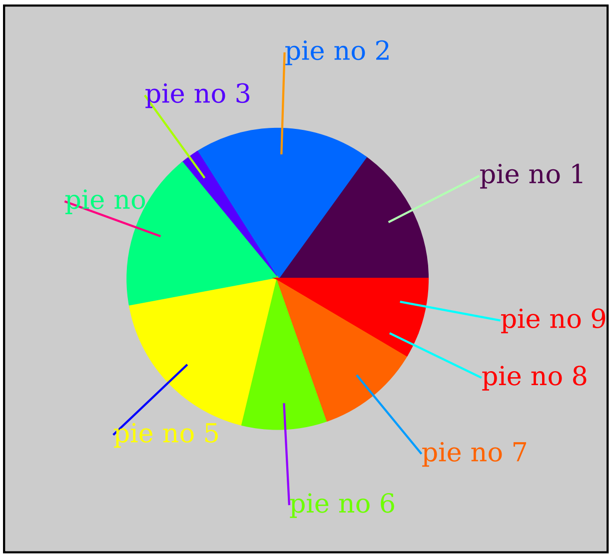
The other one is the node on the circle line. The origin is by default `(0,0)`. Moving the pie to another position can be done as usual with the `\rput`-macro. The used colors are named internally as `chartFillColor?` and can be used by the user for coloring lines or text.



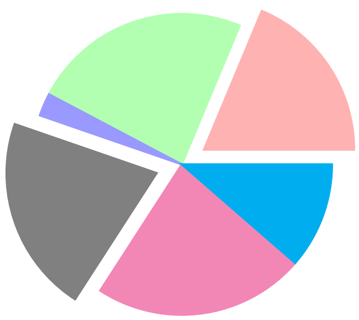
```
\begin{pspicture}(-3,-3)(3,3)
\psChart{ 23, 29, 3, 26, 28, 14 }{}{2}
\multido{\iA=1+1}{6}{%
  \psdot(psChart\iA)\psdot(psChartI\iA)\
    psdot(psChartO\iA)%
  \psline[linestyle=dashed,linecolor=white
    ](psChart\iA)
  \psline[linestyle=dashed](psChart\iA)(
    psChartO\iA)}
\end{pspicture}
```



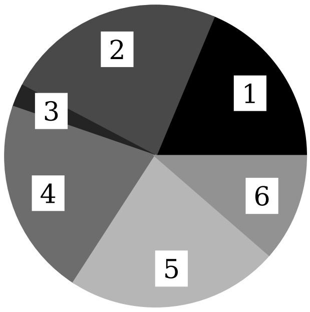

```
\begin{pspicture}(-3,-3)(3,3)
\psChart[chartColor=color]{ 45, 90 }{ 1 }{2}
\ncline[linecolor=-chartFillColor1,
  nodesepB=-20pt]{psChartO1}{psChart1}
\rput[l](psChartO1){%
  \textcolor{chartFillColor1}{pie no 1}}
\ncline[linecolor=-chartFillColor2,
  nodesepB=-20pt]{psChartO2}{psChart2}
\rput[lt](psChartO2){%
  \textcolor{chartFillColor2}{pie no 2}}
\end{pspicture}
```



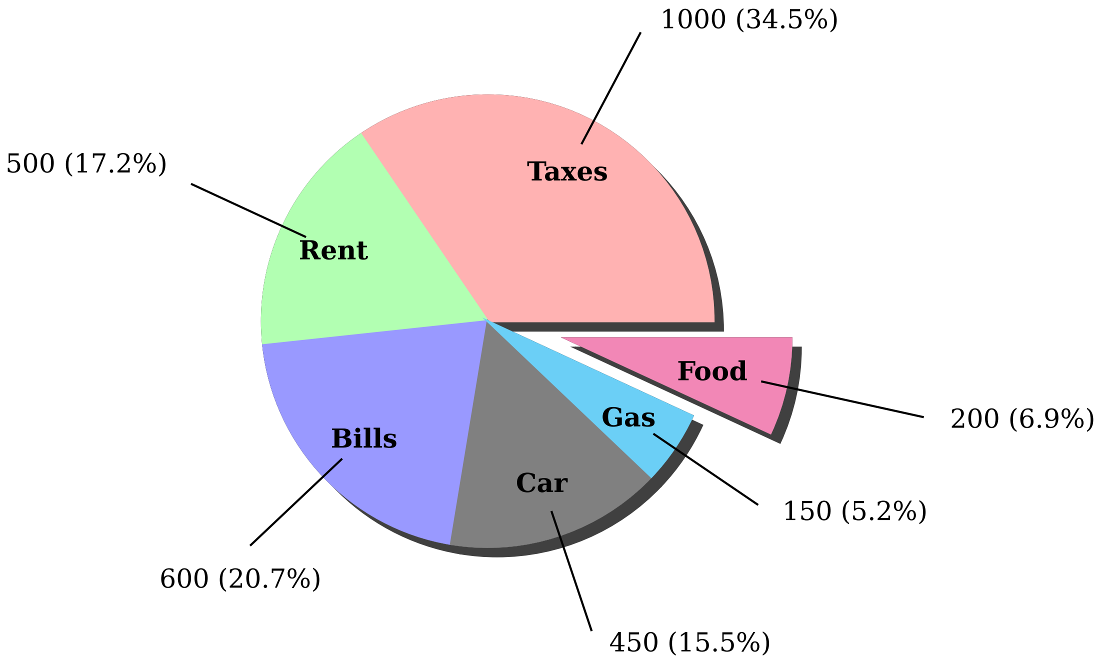

```
\psframebox[fillcolor=black!20,
  fillstyle=solid]{%
\begin{pspicture}(-3.5,-3.5)(4.25,3.5)
\psChart[chartColor=color]%
  {23, 29, 3, 26, 28, 14, 17, 4, 9}{}{2}
\multido{\iA=1+1}{9}{%
  \ncline[linecolor=-chartFillColor\iA,
    nodesepB=-10pt]{psChartO\iA}{psChart\iA}
  \rput[l](psChartO\iA){%
    \textcolor{chartFillColor\iA}{pie no \iA}}}
\end{pspicture}}
```



```
\begin{pspicture}(-3,-3)(3,3)
\psChart[userColor={red!30,green!30,
   blue!40,gray,magenta!60,cyan}]%
     { 23, 29, 3, 26, 28, 14 }{1,4}{2}
\end{pspicture}
```



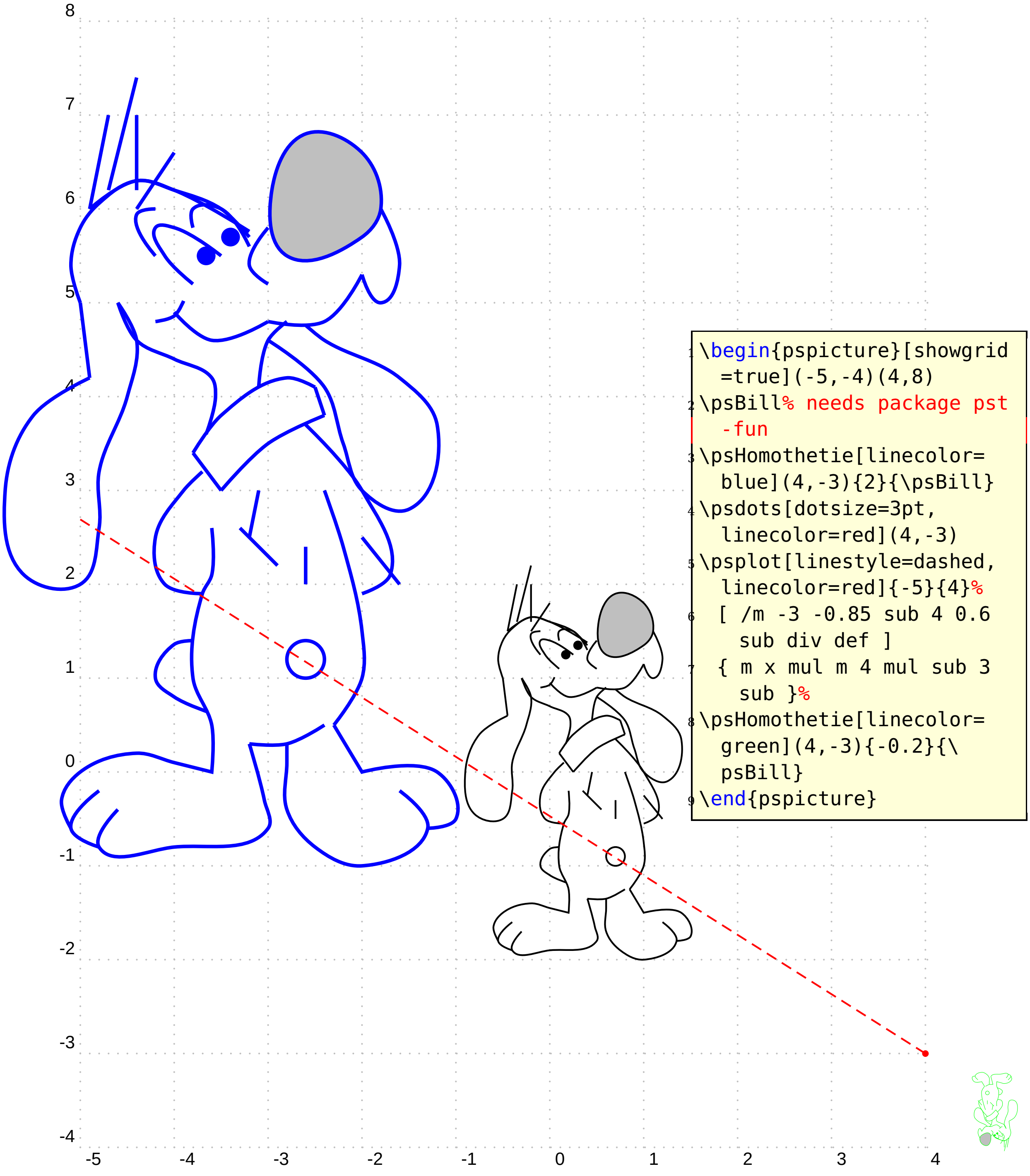

```
\begin{pspicture}(-3,-2.5)(3,2.5)
\psChart{ 23, 29, 3, 26, 28, 14 }{}{2}
\multido{\iA=1+1}{6}{\rput*(psChartI\iA){\iA}}
\end{pspicture}
```




```
\psset{unit=1.5}
\begin{pspicture}(-3,-3)(3,3)
\psChart[userColor={red!30,green!30,blue!40,gray,cyan!50,
   magenta!60,cyan},chartSep=30pt,shadow=true,shadowsize=5pt
     ]{34.5,17.2,20.7,15.5,5.2,6.9}{6}{2}
\psset{nodesepA=5pt,nodesepB=-10pt}
\ncline{psChart01}{psChart1}\nput{0}{psChart01}{1000 (34.5\%)}
\ncline{psChart02}{psChart2}\nput{150}{psChart02}{500 (17.2\%)}
\ncline{psChart03}{psChart3}\nput{-90}{psChart03}{600 (20.7\%)}
\ncline{psChart04}{psChart4}\nput{0}{psChart04}{450 (15.5\%)}
\ncline{psChart05}{psChart5}\nput{0}{psChart05}{150 (5.2\%)}
\ncline{psChart06}{psChart6}\nput{0}{psChart06}{200 (6.9\%)}
\bfseries%
\rput(psChartI1){Taxes}\rput(psChartI2){Rent}\rput(psChartI3){Bills}
\rput(psChartI4){Car}\rput(psChartI5){Gas}\rput(psChartI6){Food}
\end{pspicture}
```

8. \psHomothetie: central dilatation

\psHomothetie [Options] (*center*){*factor*}{*object*}

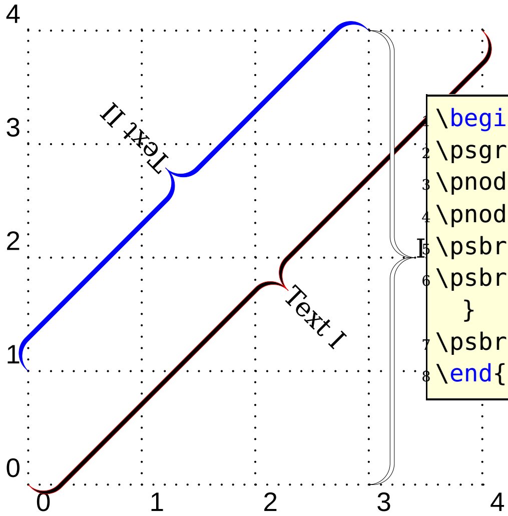



```
\begin{pspicture}[showgrid
  =true](-5,-4)(4,8)
\psBill% needs package pst
  -fun
\psHomothetie[linecolor=
  blue](4,-3){2}{\psBill}
\psdots[dotsize=3pt,
  linecolor=red](4,-3)
\psplot[linestyle=dashed,
  linecolor=red]{-5}{4}%
 [ /m -3 -0.85 sub 4 0.6
   sub div def ]
 { m x mul m 4 mul sub 3
   sub }%
\psHomothetie[linecolor=
  green](4,-3){-0.2}{\
  psBill}
\end{pspicture}
```

9. \psbrace

9.1. Syntax

`\psbrace* [Options] (A)(B){text}`

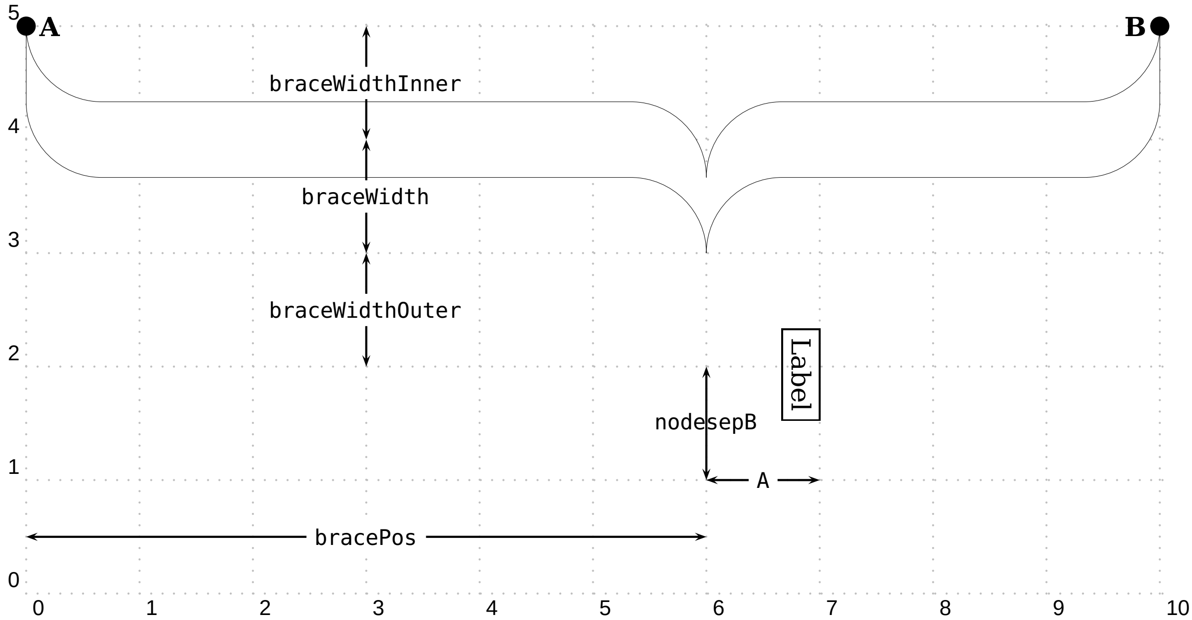



```
\begin{pspicture}(4,4)
\psgrid[subgriddiv=0,griddots=10]
\pnode(0,0){A}
\pnode(4,4){B}
\psbrace[linecolor=red,ref=lC](A)(B){Text I}
\psbrace*[linecolor=blue,ref=lC](3,4)(0,1){Text II
  }
\psbrace[fillcolor=white](3,0)(3,4){III}
\end{pspicture}
```

The option `\specialCoor` is enabled, so that all types of coordinates are possible, (nodename), (x, y), $(nodeA|nodeB)$, ... The star version fills the inner of the brace with the current linecolor. With the fillcolor `white` or any other background color the brace can be "'unfilled"'.

9.2. Options

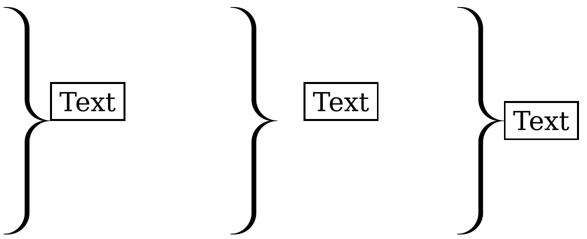
Additional to all other available options from `pstricks` or the other related packages, there are two new option, named `braceWidth` and `bracePos`. All important ones are shown in the following graphics and table.



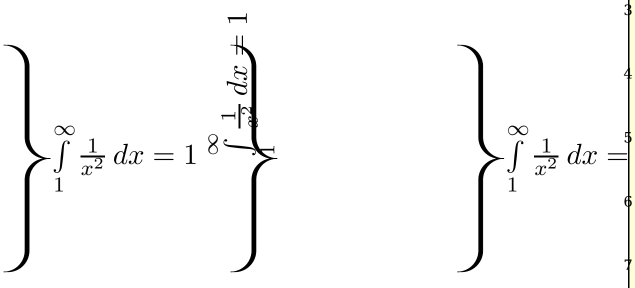
A positive value for nodesepA and nodesepB shifts the label to the upper right and a negative value to the lower left. This does not depends on the value for the rotating of the label!

name	meaning
braceWidth	default is \pslinewidth
braceWidthInner	default is 10\pslinewidth
braceWidthOuter	default is 10\pslinewidth
bracePos	relative position (default is 0.5)
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)
fillcolor	default is black

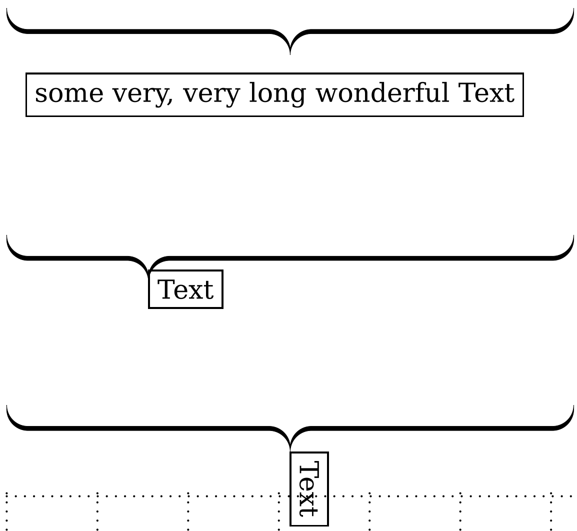
By default the text is written perpendicular to the brace line and can be changed with the pstricks option rot=... The text parameter can take any object and may also be empty. The reference point can be any value of the combination of l (left) or r (right) and b (bottom) or B (Baseline) or C (center)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



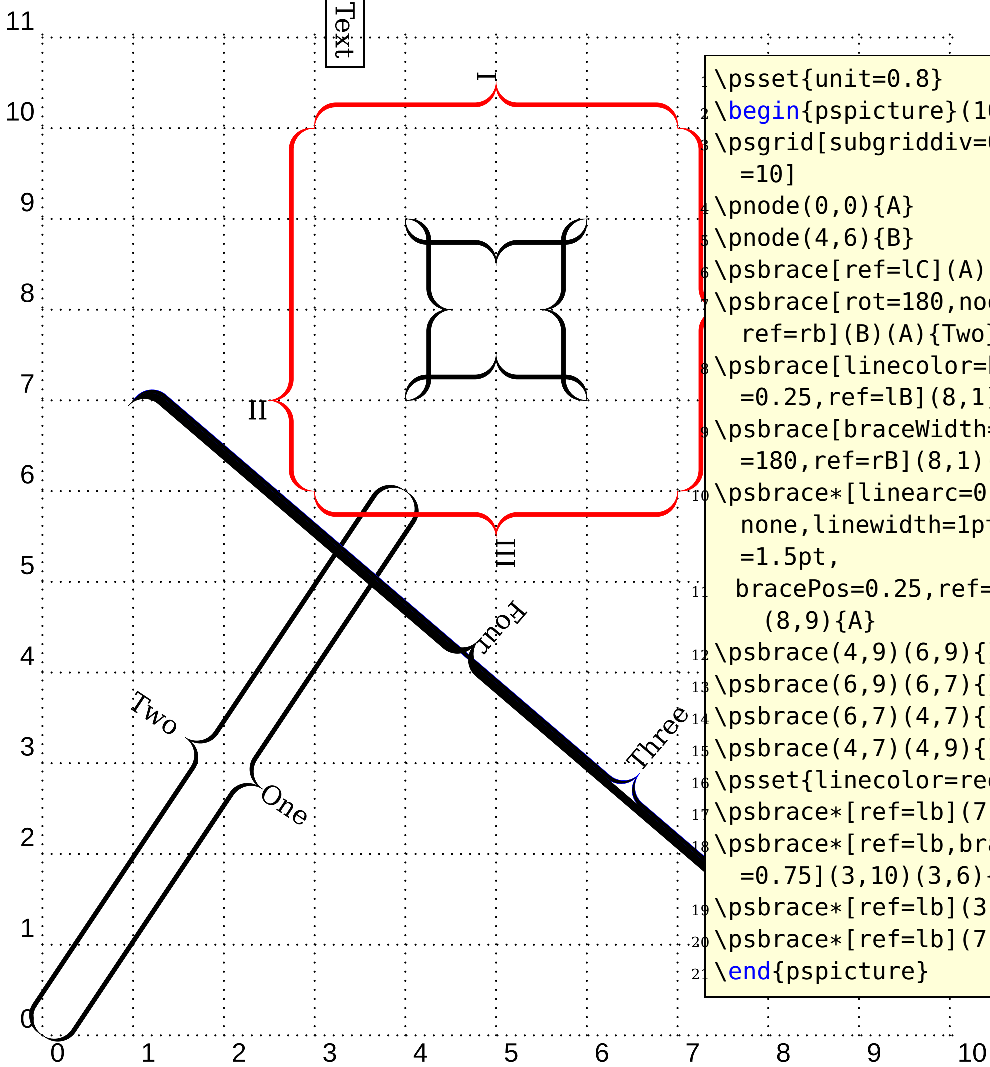

```
\begin{pspicture}(8,2.5)
\psbrace(0,0)(0,2){\fbox{Text}}%
\psbrace[nodesepA=10pt](2,0)(2,2)
  {\fbox{Text}}
\psbrace[ref=lC](4,0)(4,2){\fbox{
  Text}}
\psbrace[ref=lt,rot=90,nodesepB
  =-15pt](6,0)(6,2){\fbox{Text}}
\psbrace[ref=lt,rot=90,nodesepA=-5
  pt,nodesepB=15pt](8,2)(8,0){\
  fbox{Text}}
\end{pspicture}
```



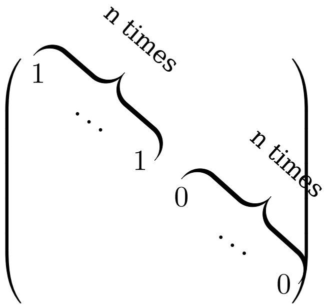
```
\def\someMath{$\int\limits_1^{\
  infty}\frac{1}{x^2}\,dx=1$}
\begin{pspicture}(8,2.5)
\psbrace[ref=lC](0,0)(0,2){\
  someMath}%
\psbrace[rot=90](2,0)(2,2){\
  someMath}
\psbrace[ref=lC](4,0)(4,2){\
  someMath}
\psbrace[ref=lt,rot=90,nodesepB
  =-30pt](6,0)(6,2){\someMath}
\psbrace[ref=lt,rot=90,nodesepB=30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




```
\psset{unit=0.8}
\begin{pspicture}(10,11)
\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,ref=lB](8,1)(1,7){Three}
\psbrace[braceWidth=-1mm,rot=180,ref=rB](8,1)(1,7){Four}
\psbrace*[linearc=0.5,fillstyle=none,linewidth=1pt,braceWidth=1.5pt,
 bracePos=0.25,ref=lC](8,1)(8,9){A}
\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[rot=-90,nodesepB=-0.5,nodesepA=-0.2](B)(A){\small n times}
\psbrace[rot=-90,nodesepB=-0.5,nodesepA=-0.2](D)(C){\small n times}
```

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

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

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

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

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

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

\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}
```

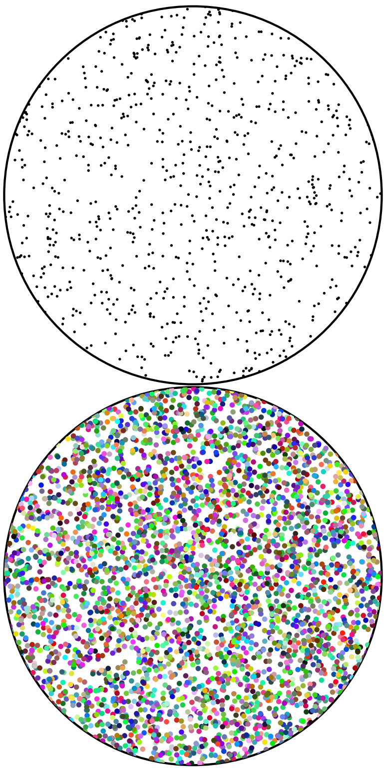
10. Random dots

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

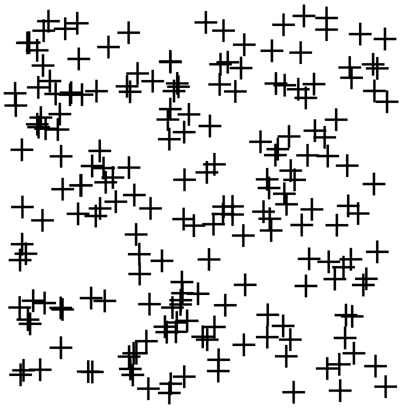
`\psRandom[Options]{}`
`\psRandom[Options]`(x_{Min}, y_{Min}) (x_{Max}, y_{Max}) `{`*clip path*`}`

If there is no area for the dots defined, then `(0,0)(1,1)` in the current scale setting is used for placing the dots. If there is only one (x_{Max}, y_{Max}) defined, then `(0,0)` is used for the other point. 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:

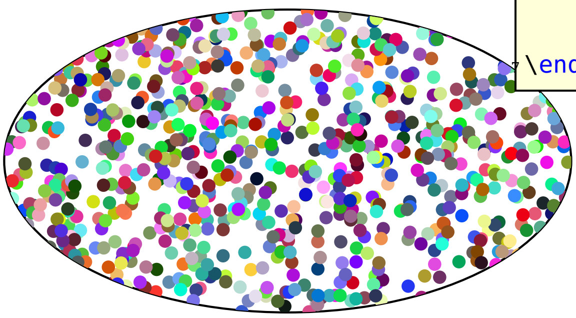
name	default	
`randomPoints`	`1000`	number of random dots
`color`	`false`	random color

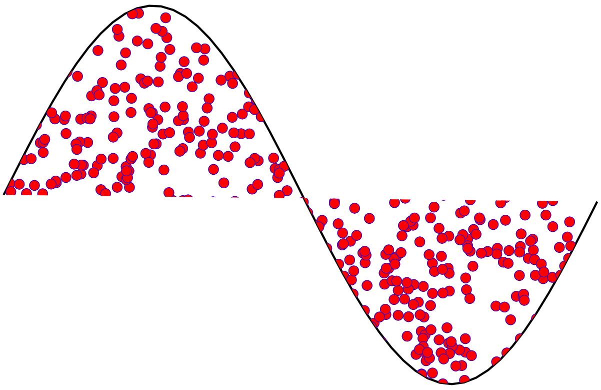


```
\psset{unit=5cm}
\begin{pspicture}(1,1)
  \psRandom[dotsize=1pt,fillstyle=solid](1,1){\
    pscircle(0.5,0.5){0.5}}
\end{pspicture}
\begin{pspicture}(1,1)
  \psRandom[dotsize=2pt,randomPoints=5000,color,%
      fillstyle=solid](1,1){\pscircle(0.5,0.5){0.5}}
\end{pspicture}
```



```
\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}
```

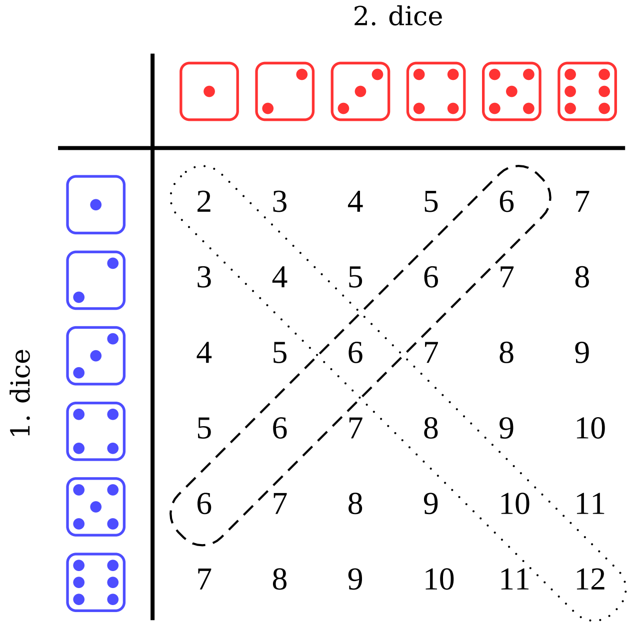




```
\psset{unit=2.5cm}
\begin{pspicture}(0,-1)(3,1)
 \psRandom[dotsize=4pt,dotstyle=o,
   linecolor=blue,fillcolor=red,%
    fillstyle=solid,randomPoints
      =1000]%
     (0,-1)(3,1){\psplot{0}{3.14}{
       x 114 mul sin }}
\end{pspicture}
```

11. Dice

\psdice creates the view of a dice. The number on the dice is the only parameter. The optional parameters, like the color can be used as usual. The macro is a box of dimension zero and is placed at the current point. Use the \rput macro to place it anywhere. The optional argument unit can be used to scale the dice. the default size of the dice $1\mathrm{cm} \times 1\mathrm{cm}$.




```
\begin{pspicture}(-1,-1)(8,8)
\multido{\iA=1+1}{6}{%
  \rput(\iA,7.5){\Huge\psdice[unit=0.75,linecolor=red!80]{\iA}}
  \rput(! -0.5 7 \iA\space sub){\Huge\psdice[unit=0.75,linecolor=blue!70]{\
    iA}}%
  \multido{\iB=1+1}{6}{%
    \rput(! \iA\space 7 \iB\space sub){%
      \rnode[c]{p\iA\iB}{\makebox[1em][l]{\strut\psPrintValue[fontscale=12]{\
        iA\space \iB\space add}}}%
}}}
\ncbox[linearc=0.35,nodesep=0.2,linestyle=dotted]{p11}{p66}
\ncbox[linearc=0.35,nodesep=0.2,linestyle=dashed]{p15}{p51}
\rput{90}(-1.5,3.5){1. dice}
\rput{0}(3.5,8.5){2. dice}
\psline[linewidth=1.5pt](0.25,0.5)(0.25,8)
\psline[linewidth=1.5pt](-1,6.75)(6.5,6.75)
\end{pspicture}
```

12. Arrows

12.1. Definition

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

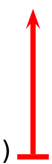
Value	Example	Name
`-`		None
`<->`		Arrowheads.
`>-<`		Reverse arrowheads.
`<<->>`		Double arrowheads.
`>>-<<`		Double reverse arrowheads.
`|-|`		T-bars, flush to endpoints.
`|*-|*`		T-bars, centered on endpoints.
`[-]`		Square brackets.
`]-[`		Reversed square brackets.
`(-)`		Rounded brackets.
`)-(`		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.
`v-v|`		left/right inside vee arrows.
`V-V|`		left/right outside vee arrows.
`f-f|`		left/right inside filled arrows.
`F-F|`		left/right outside filled arrows.
`t-t|`		left/right inside slash arrows.
`T-T|`		left/right outside slash 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)
```



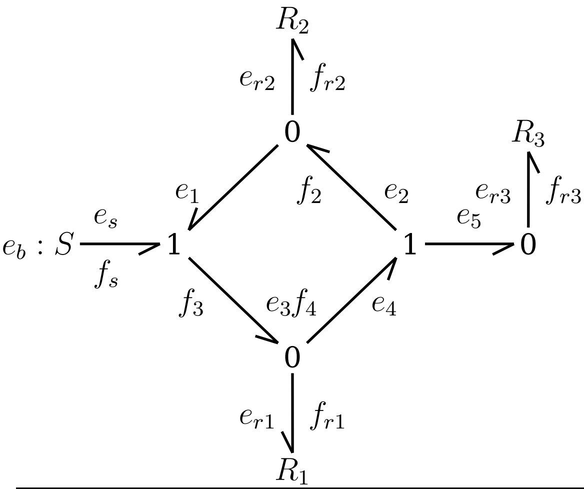
12.2. Multiple arrows

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)`	

12.3. hookarrow




```
\psset{arrowsize=8pt,arrowlength=1,linewidth=1pt,nodesep=2pt,shortput=tablr}
\large
\begin{psmatrix}[colsep=12mm,rowsep=10mm]
      &  & $R_2$        \\
      &  &  0  &  & $R_3$\\
```

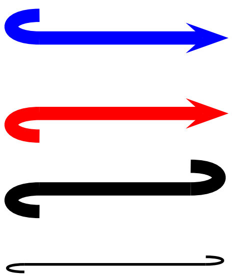
```
$e_b:S$ & 1 &  & 1 & 0 \\
      &  &  0         \\
      &  &  $R_1$     \\
\end{psmatrix}
\ncline{h-}{1,3}{2,3}<{$e_{r2}$}>{$f_{r2}$}\ncline{-h}{2,3}{3,2}<{$e_1$}
\ncline{-h}{3,1}{3,2}^{$e_s$}_{$f_{s}$} \ncline{-h}{3,2}{4,3}>{$e_3$}<{$f
  _3$}
\ncline{-h}{4,3}{3,4}>{$e_4$}<{$f_4$} \ncline{-h}{3,4}{2,3}>{$e_2$}<{$f_2$}
\ncline{-h}{3,4}{3,5}^{$e_5$}
\ncline{-h}{3,5}{2,5}<{$e_{r3}$}>{$f_{r3}$}
\ncline{-h}{4,3}{5,3}<{$e_{r1}$}>{$f_{r1}$}
```

12.4. hookrightarrow and hookleftarrow

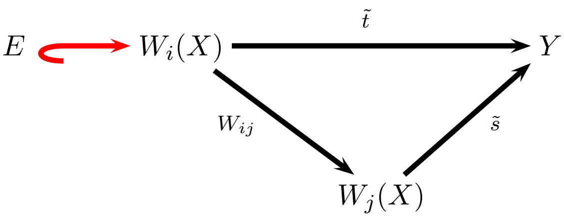
This is another type of arrow and is abbreviated with H. The length and width of the hook is set by the new options hooklength and hookwidth, which are by default set to

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

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



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




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

12.5. ArrowInside Option

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

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

Without the default arrow definition there is only the one inside the line, defined by the type and the position. The position is relative to the length of the whole line. 0.25 means at 25% of the line length. The peak of the arrow gets the coordinates which are calculated by the macro. If you want arrows with an absolute 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.

> The ArrowInside takes only arrow definitions like -> into account. Arrows from right to left (<-) are not possible and ignored. If you need such arrows, change the order of the pairs of coordinates for the line or curve macro.

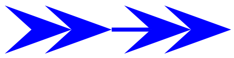
12.6. ArrowFill Option

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

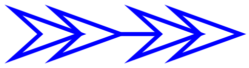
```
\psset{arrowscale=2.5}
\psline[linecolor=red,arrowinset=0]{<->}(-1,0)(2,0)
```

```
\psset{arrowscale=2.5}
\psline[linecolor=red,arrowinset=0,ArrowFill=false
  ]{<->}(-1,0)(2,0)
```

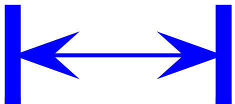
```
\psset{arrowscale=2.5}
\psline[linecolor=red,arrowinset=0,arrowsize=0.2,
  ArrowFill=false]{<->}(-1,0)(2,0)
```



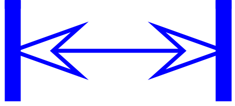
```
\psline[linecolor=blue,arrowscale=4,
  ArrowFill]{>>->>}(-1,0)(2,0)
```



```
\psline[linecolor=blue,arrowscale=4,
  ArrowFill=false]{>>->>}(-1,0)(2,0)
\rule{3cm}{0pt}\\[30pt]
```



```
\psline[linecolor=blue,arrowscale=4,
  ArrowFill]{>|->|}(-1,0)(2,0)
```

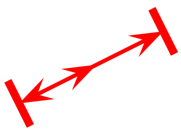


```
\psline[linecolor=blue,arrowscale=4,
  ArrowFill=false]{>|->|}(-1,0)(2,0)%
```

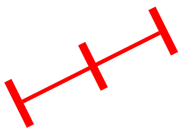
12.7. Examples

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

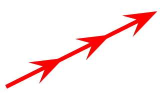
\psline



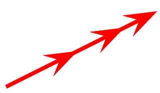
```
\begin{pspicture}(2,2)
\psset{arrowscale=2,ArrowFill=true}
\psline[ArrowInside=->]{|<->|}(2,1)
\end{pspicture}
```



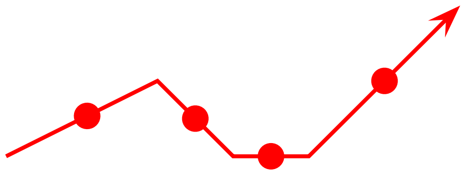
```
\begin{pspicture}(2,2)
\psset{arrowscale=2,ArrowFill=true}
\psline[ArrowInside=-|]{|-|}(2,1)
\end{pspicture}
```



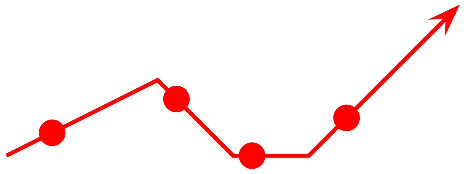
```
\begin{pspicture}(2,2)
\psset{arrowscale=2,ArrowFill=true}
\psline[ArrowInside=->,ArrowInsideNo=2]{->}(2,1)
\end{pspicture}
```



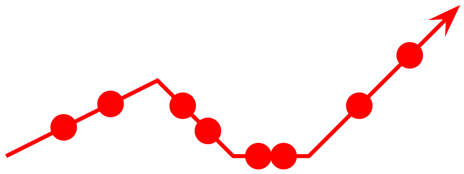
```
\begin{pspicture}(2,2)
\psset{arrowscale=2,ArrowFill=true}
\psline[ArrowInside=->,ArrowInsideNo=2,ArrowInsideOffset
  =0.1]{->}(2,1)
\end{pspicture}
```



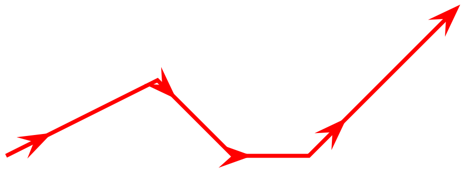
```
\begin{pspicture}(6,2)
\psset{arrowscale=2,ArrowFill=true}
\psline[ArrowInside=-*]{->}(0,0)(2,1)
  (3,0)(4,0)(6,2)
\end{pspicture}
```



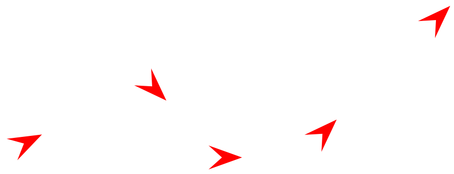
```
\begin{pspicture}(6,2)
\psset{arrowscale=2,ArrowFill=true}
\psline[ArrowInside=-*,ArrowInsidePos
  =0.25]{->}(0,0)(2,1)(3,0)(4,0)(6,2)
\end{pspicture}
```



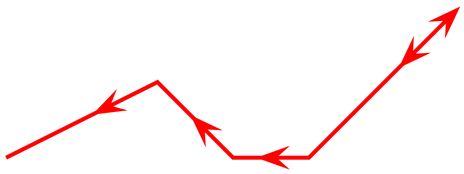
```
\begin{pspicture}(6,2)
\psset{arrowscale=2,ArrowFill=true}
\psline[ArrowInside=-*,ArrowInsidePos
  =0.25,ArrowInsideNo=2]{->}%
   (0,0)(2,1)(3,0)(4,0)(6,2)
\end{pspicture}
```



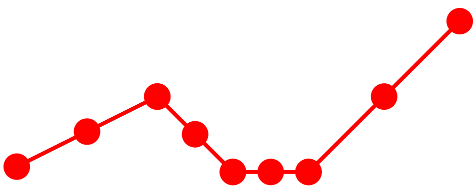
```
\begin{pspicture}(6,2)
\psset{arrowscale=2,ArrowFill=true}
\psline[ArrowInside=->, ArrowInsidePos
  =0.25]{->}%
       (0,0)(2,1)(3,0)(4,0)(6,2)
\end{pspicture}
```



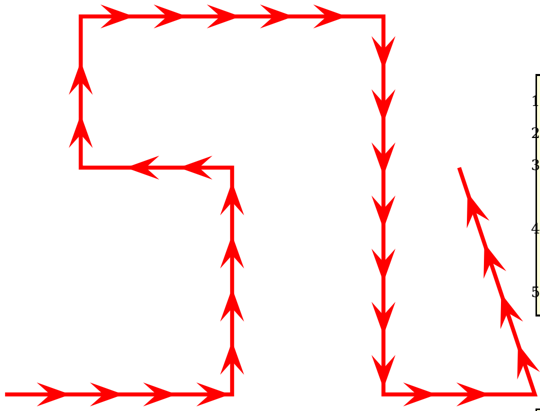
```
\begin{pspicture}(6,2)
\psset{arrowscale=2,ArrowFill=true}
\psline[linestyle=none,ArrowInside=->,
  ArrowInsidePos=0.25]{->}%
       (0,0)(2,1)(3,0)(4,0)(6,2)
\end{pspicture}
```



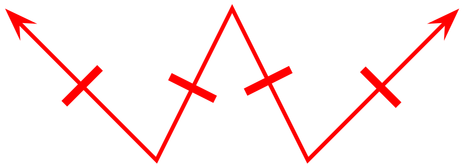
```
\begin{pspicture}(6,2)
\psset{arrowscale=2,ArrowFill=true}
\psline[ArrowInside=-<, ArrowInsidePos
  =0.75]{->}%
    (0,0)(2,1)(3,0)(4,0)(6,2)
\end{pspicture}
```



```
\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}
```

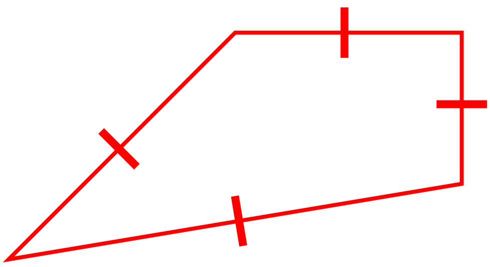


```
\begin{pspicture}(6,5)
\psset{arrowscale=2,ArrowFill=true}
\psline[ArrowInside=->,ArrowInsidePos=20](0,0)(3,0)%
      (3,3)(1,3)(1,5)(5,5)(5,0)(7,0)(6,3)
\end{pspicture}
```

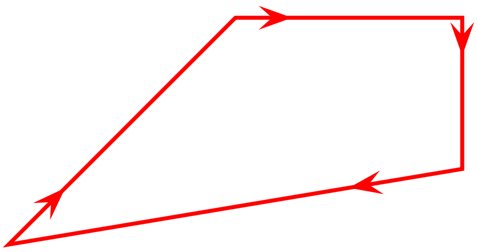


```
\begin{pspicture}(6,2)
\psset{arrowscale=2,ArrowFill=true}
\psline[ArrowInside=-|]{<->}(0,2)(2,0)(3,2)(4,0)(6,2)
\end{pspicture}
```

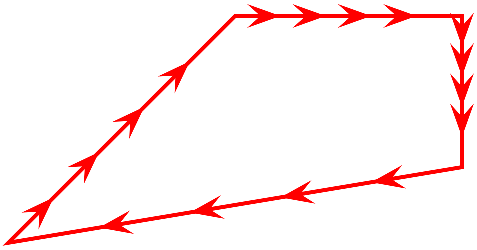
\pspolygon



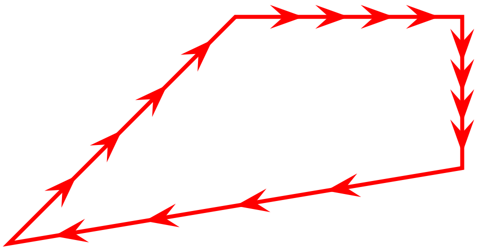
```
\begin{pspicture}(6,3)
\psset{arrowscale=2}
\pspolygon[ArrowInside=-|](0,0)(3,3)(6,3)(6,1)
\end{pspicture}
```



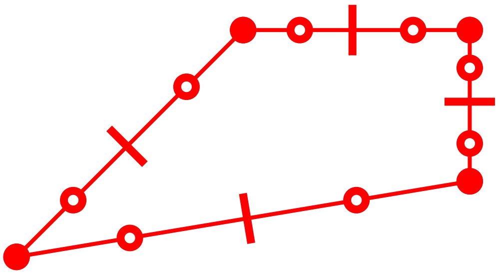
```
\begin{pspicture}(6,3)
\psset{arrowscale=2}
\pspolygon[ArrowInside=->,ArrowInsidePos=0.25]%
    (0,0)(3,3)(6,3)(6,1)
\end{pspicture}
```



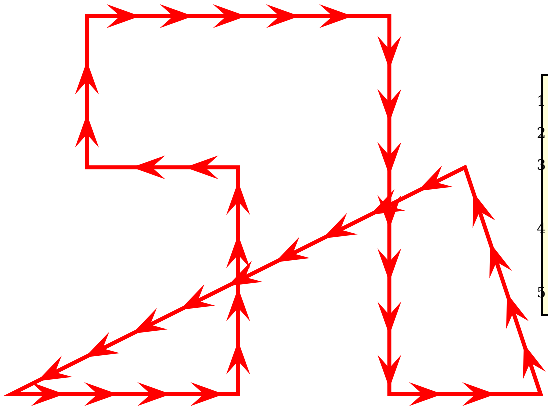
```
\begin{pspicture}(6,3)
\psset{arrowscale=2}
\pspolygon[ArrowInside=->,ArrowInsideNo=4]%
      (0,0)(3,3)(6,3)(6,1)
\end{pspicture}
```



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

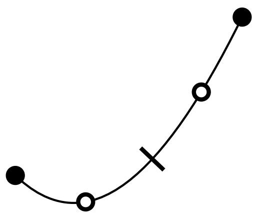


```
\begin{pspicture}(6,3)
\psset{arrowscale=2}
 \pspolygon[ArrowInside=-|](0,0)(3,3)
   (6,3)(6,1)
 \psset{linestyle=none,ArrowInside=-*}
 \pspolygon[ArrowInsidePos=0](0,0)(3,3)
   (6,3)(6,1)
 \pspolygon[ArrowInsidePos=1](0,0)(3,3)
   (6,3)(6,1)
 \psset{ArrowInside=-o}
 \pspolygon[ArrowInsidePos=0.25](0,0)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

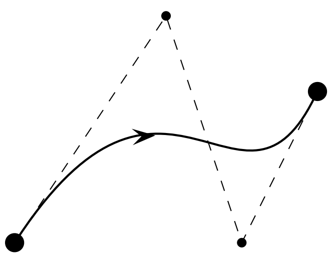


```
\begin{pspicture}(6,5)
\psset{arrowscale=2}
  \pspolygon[ArrowInside=->,
    ArrowInsidePos=20]%
    (0,0)(3,0)(3,3)(1,3)(1,5)(5,5)(5,0)
      (7,0)(6,3)
\end{pspicture}
```

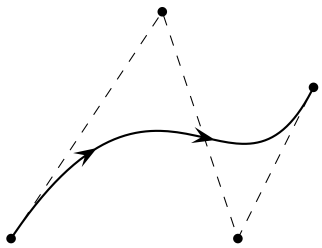
\psbezier



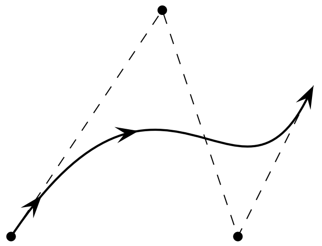
```
\begin{pspicture}(3,3)
\psset{arrowscale=2}
  \psbezier[ArrowInside=-|](0,1)(1,0)(2,1)(3,3)
  \psset{linestyle=none,ArrowInside=-o}
  \psbezier[ArrowInsidePos=0.25](0,1)(1,0)(2,1)(3,3)
  \psbezier[ArrowInsidePos=0.75](0,1)(1,0)(2,1)(3,3)
  \psset{linestyle=none,ArrowInside=-*}
  \psbezier[ArrowInsidePos=0](0,1)(1,0)(2,1)(3,3)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



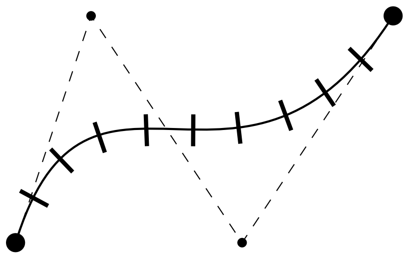
```
\begin{pspicture}(4,3)
\psset{arrowscale=2}
  \psbezier[ArrowInside=->,showpoints=true]%
     {*-*}(0,0)(2,3)(3,0)(4,2)
\end{pspicture}
```



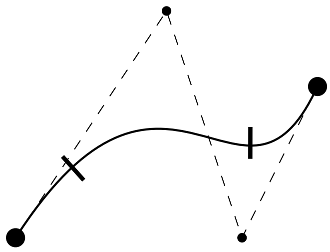
```
\begin{pspicture}(4,3)
\psset{arrowscale=2}
 \psbezier[ArrowInside=->,showpoints=true,
     ArrowInsideNo=2](0,0)(2,3)(3,0)(4,2)
\end{pspicture}
```



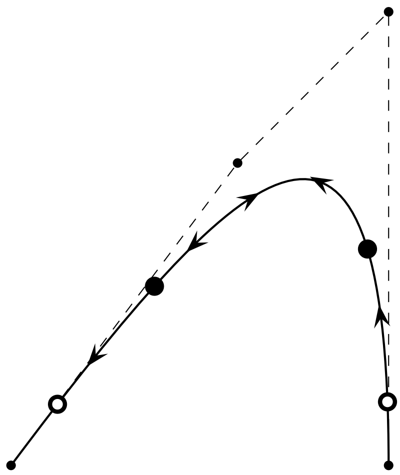
```
\begin{pspicture}(4,3)
\psset{arrowscale=2}
 \psbezier[ArrowInside=->,showpoints=true,
    ArrowInsideNo=2,ArrowInsideOffset=-0.2]%
     {->}(0,0)(2,3)(3,0)(4,2)
\end{pspicture}
```



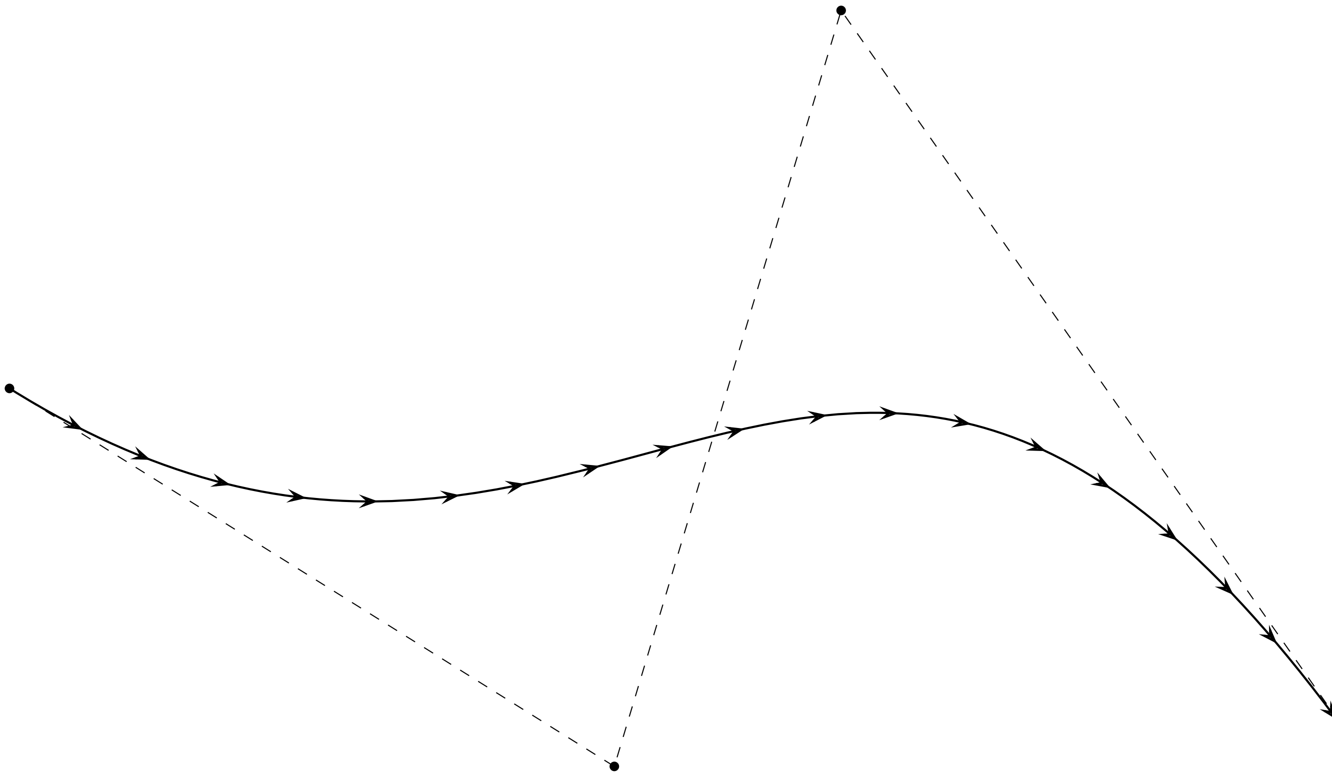
```
\begin{pspicture}(5,3)
\psset{arrowscale=2}
 \psbezier[ArrowInsideNo=9,ArrowInside=-|,%
   showpoints=true]{*-*}(0,0)(1,3)(3,0)(5,3)
\end{pspicture}
```



```
\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](0,0)(2,3)(3,0)
   (4,2)
\end{pspicture}
```



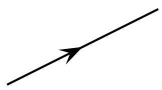
```
\begin{pspicture}(5,6)
\psset{arrowscale=2}
 \pnode(3,4){A}\pnode(5,6){B}\pnode(5,0){C}
 \psbezier[ArrowInside=->,%
    showpoints=true](A)(B)(C)
 \psset{linestyle=none,ArrowInside=-<}
 \psbezier[ArrowInsideNo=4](0,0)(A)(B)(C)
 \psset{ArrowInside=-o}
 \psbezier[ArrowInsidePos=0.1](0,0)(A)(B)(C)
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



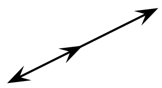
```
\begin{pspicture}(-3,-5)(15,5)
  \psbezier[ArrowInsideNo=19,%
     ArrowInside=->,ArrowFill=false,%
     showpoints=true]{->}(-3,0)(5,-5)(8,5)(15,-5)
\end{pspicture}
```

\pcline

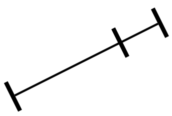
These examples need the package `pst-node`.



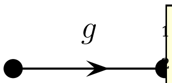
```
\begin{pspicture}(2,1)
\psset{arrowscale=2}
\pcline[ArrowInside=->](0,0)(2,1)
\end{pspicture}
```



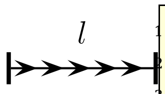
```
\begin{pspicture}(2,1)
\psset{arrowscale=2}
\pcline[ArrowInside=->]{<->}(0,0)(2,1)
\end{pspicture}
```



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



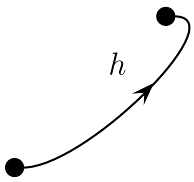

```
\psset{arrowscale=2}
\pcline[ArrowInside=->,ArrowInsidePos=0.65]{*-*}(0,0)(2,0)
\naput[labelsep=0.3]{\large$g$}
```



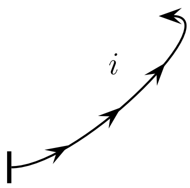

```
\psset{arrowscale=2}
\pcline[ArrowInside=->,ArrowInsidePos=10]{|-|}(0,0)(2,0)
\naput[labelsep=0.3]{\large$l$}
```

\pccurve

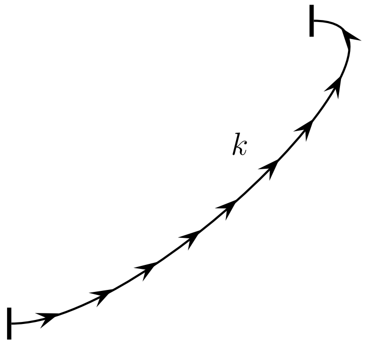
These examples also need the package `pst-node`.




```
\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}
```

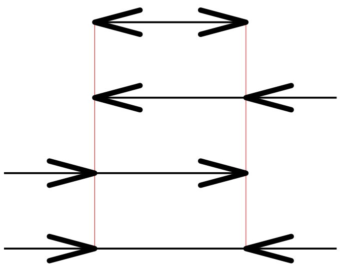



```
\begin{pspicture}(4,4)
\psset{arrowscale=2}
\pccurve[ArrowInside=->,ArrowInsidePos
  =20]{|-|}(0,0)(4,4)
\naput[labelsep=0.3]{\large$k$}
\end{pspicture}
```

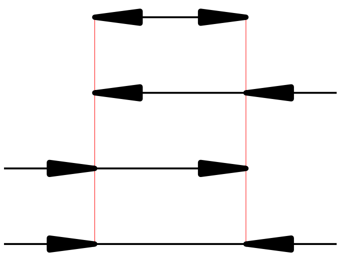
12.8. Special arrows v-V,t-T, and f-F

Possible optional arguments are

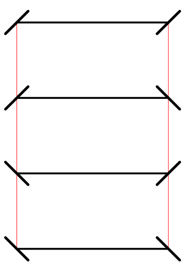
name	meaning
veearrowlength	default is 3mm
veearrowangle	default is 30
veearrowlinewidth	default is 0.35mm
filledveearrowlength	default is 3mm
filledveearrowangle	default is 15
filledveearrowlinewidth	default is 0.35mm
tickarrowlength	default is 1.5mm
tickarrowlinewidth	default is 0.35mm



```
\psset{unit=5mm}
\begin{pspicture}(4,6)
 \psset{dimen=middle,arrows=c-c,
   arrowscale=2,linewidth=.25mm}
 \psline[linecolor=red,linewidth=.05mm](0,0)(0,6)
 \psline[linecolor=red,linewidth=.05mm](4,0)(4,6)
 \psline{v-v}(0,6)(4,6)
 \psline{v-V}(0,4)(4,4)
 \psline{V-v}(0,2)(4,2)
 \psline{V-V}(0,0)(4,0)
\end{pspicture}
```



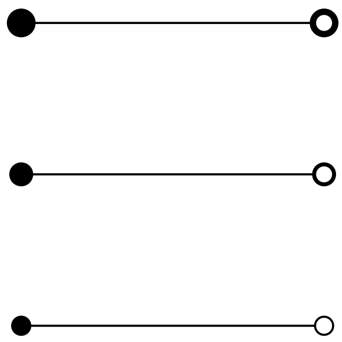
```
\psset{unit=5mm}
\begin{pspicture}(4,6)
 \psset{dimen=middle,arrows=c-c,
   arrowscale=2,linewidth=.25mm}
 \psline[linecolor=red,linewidth=.05mm](0,0)(0,6)
 \psline[linecolor=red,linewidth=.05mm](4,0)(4,6)
 \psline{f-f}(0,6)(4,6)
 \psline{f-F}(0,4)(4,4)
 \psline{F-f}(0,2)(4,2)
 \psline{F-F}(0,0)(4,0)
\end{pspicture}
```



```
\psset{unit=5mm}
\begin{pspicture}(4,6)
 \psset{dimen=middle,arrows=c-c,linewidth=.25mm}
 \psline[linecolor=red,linewidth=.05mm](0,0)(0,6)
 \psline[linecolor=red,linewidth=.05mm](4,0)(4,6)
 \psline{t-t}(0,6)(4,6)
 \psline{t-T}(0,4)(4,4)
 \psline{T-t}(0,2)(4,2)
 \psline{T-T}(0,0)(4,0)
\end{pspicture}
```

12.9. Special arrow option arrowLW

Only for the arrowtype o and ∗ it is possible to set the arrowlinewidth with the optional keyword arrowLW. When scaling an arrow by the keyword arrowscale the width of the borderline is also scaled. With the optional argument arrowLW the line width can be set separately and is not taken into account by the scaling value.



```
\begin{pspicture}(4,6)
\psline[arrowscale=3,arrows=*-o](0,5)(4,5)
\psline[arrowscale=3,arrows=*-o,
  arrowLW=0.5pt](0,3)(4,3)
\psline[arrowscale=3,arrows=*-o,
  arrowLW=0.3333\pslinewidth](0,1)(4,1)
\end{pspicture}
```

13. \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.

14. Color

14.1. Transparent colors

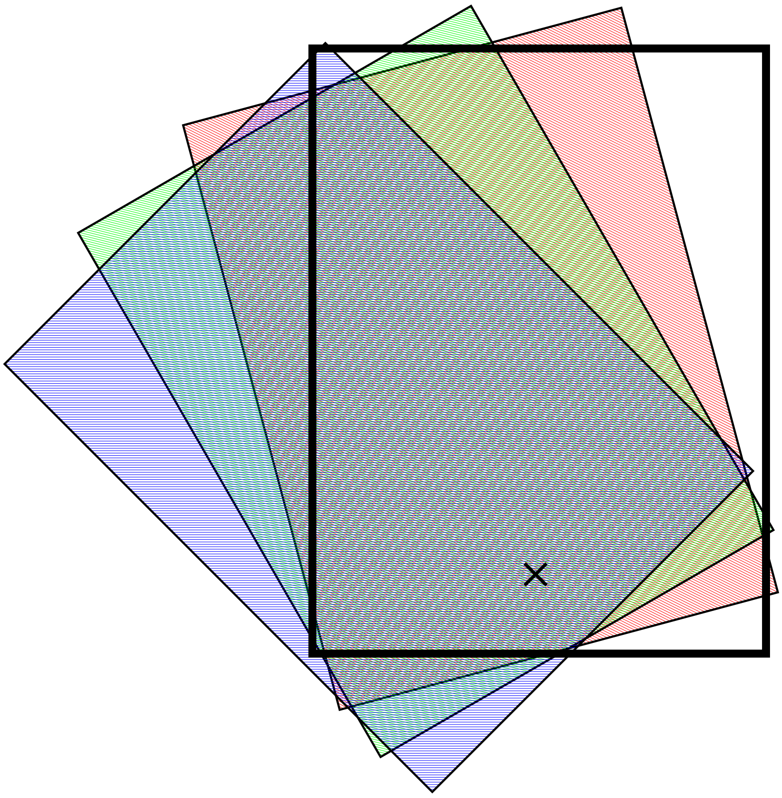
Transparency is now part of the main `pstricks` package. But pay attention, the names and syntax have changed and you need to run `ps2pdf` with the option `-dCompatibilityLevel=1.4`.

14.2. „Manipulating transparent colors"

`pstricks-add` supports real transparency and a simulated one 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:



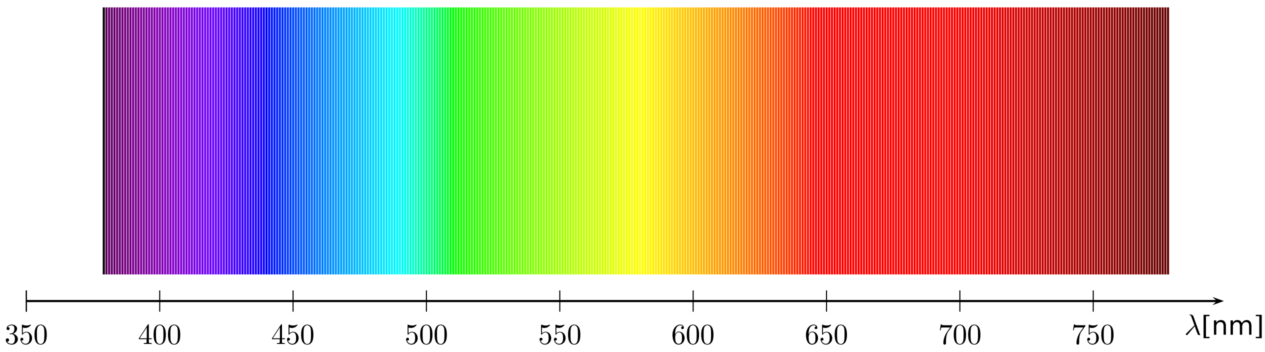
```
\begin{pspicture}(-3,-5)(5,5)
\psframe(-1,-3)(5,5) % objet de base
\psrotate(2,-2){15}{%
  \psframe[style=TRed](-1,-3)(5,5)}
\psrotate(2,-2){30}{%
  \psframe[style=TGreen](-1,-3)(5,5)}
\psrotate(2,-2){45}{%
  \psframe[style=TBlue](-1,-3)(5,5)}
\psframe[linewidth=3pt](-1,-3)(5,5)
\psdots[dotstyle=+,dotangle=45,dotscale=3](2,-2)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

14.3. Calculated colors

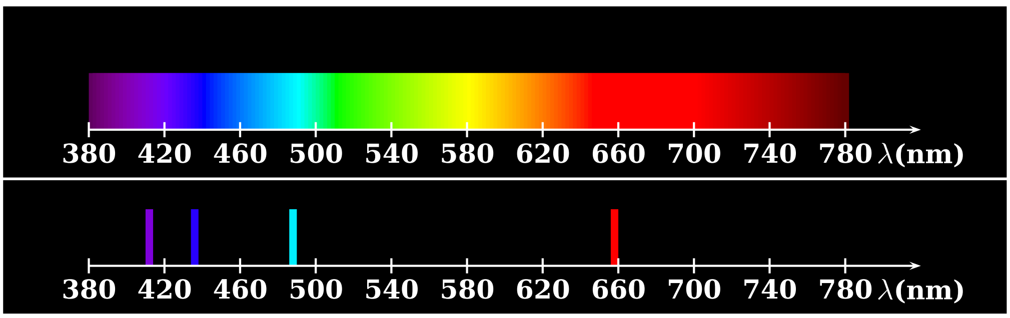
The `xcolor` package (version 2.6) has a new feature for defining colors:

```
\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 the `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)
```

14.4. Gouraud shading

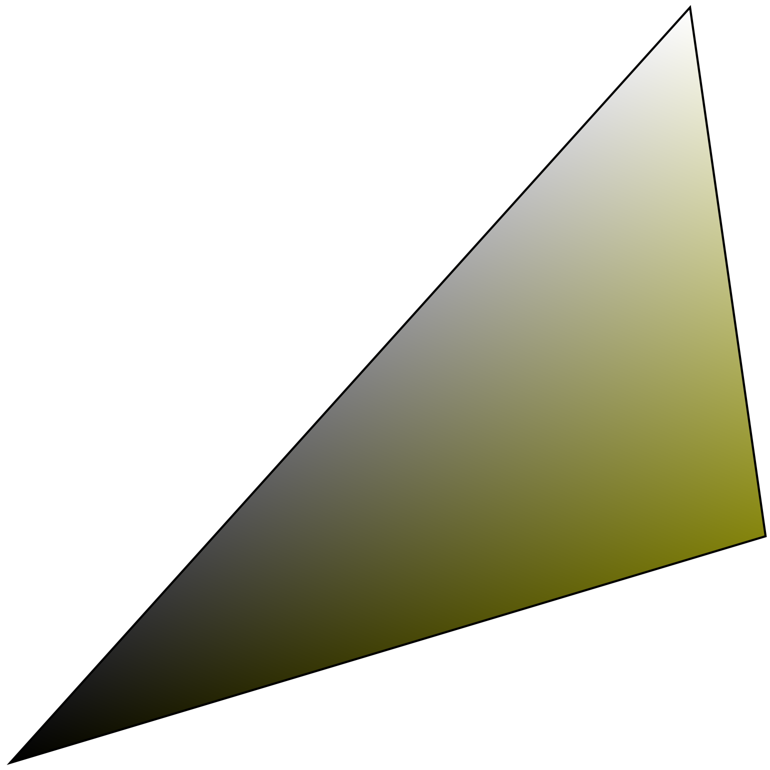
Gouraud shading is a method used in computer graphics to simulate the differing effects of light and colour across the surface of an object. In practice, Gouraud shading is used to achieve smooth lighting on low-polygon surfaces without the heavy computational requirements of calculating lighting for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

http://www.wikipedia.org

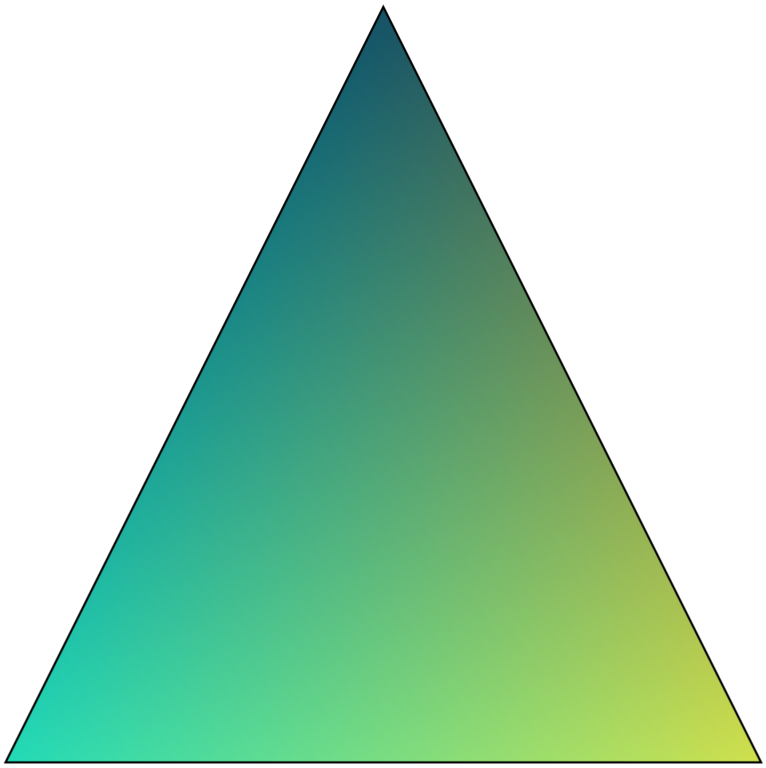
PostScript level 3 supports this kind of shading and it can only be seen with Acroread 7 or later. The syntax is easy:

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

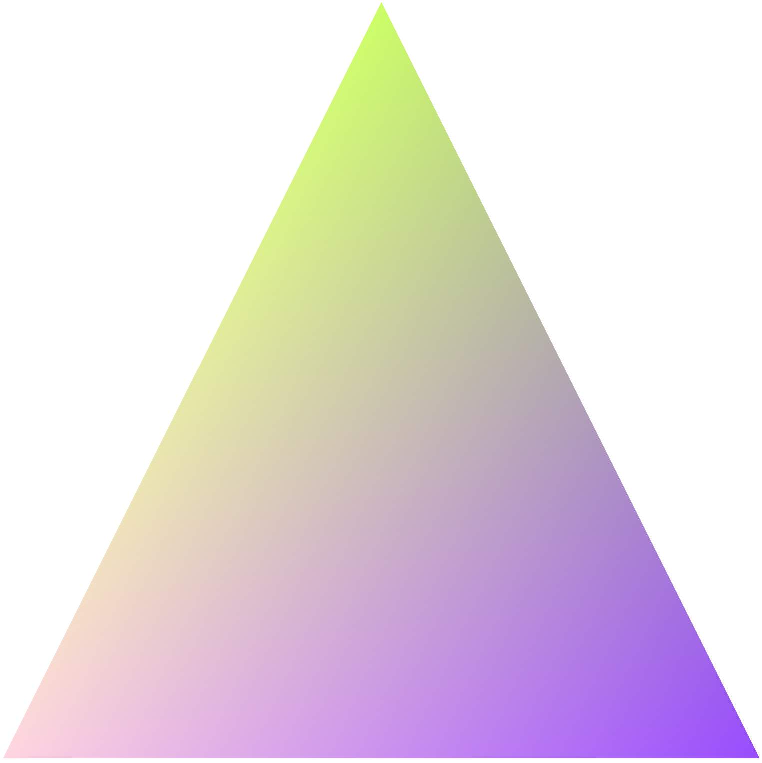
```
\begin{pspicture}(0,-.25)(10,10)
  \psGTriangle(0,0)(5,10)(10,0){red}{green}{blue}
\end{pspicture}
```



```
\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}
```

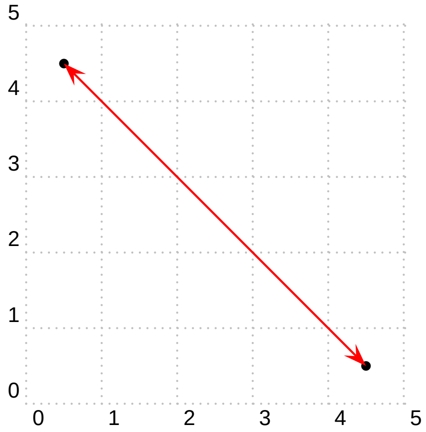


```
\definecolor{rose}{rgb}{1.00, 0.84, 0.88}
\definecolor{vertpommepasmure}{rgb}{0.80, 1.0, 0.40}
\definecolor{fushia}{rgb}{0.60, 0.30, 1.0}
\begin{pspicture}(0,-.25)(10,10)
  \psGTriangle(0,0)(5,10)(10,0){rose}{vertpommepasmure}{fushia}
\end{pspicture}
```

Part II.
pst-node

15. Relative nodes with \psGetNodeCenter

The command `\psGetNodeCenter{`*node*`}` makes sense only at the PostScript level. It defines the two variables *node.x* and *node.y* which can be used to define relative nodes. The following example defines the node `MyNode` and a second one relative to the first one, with 4 units left and 4 units up. *node* must be an existing node name.



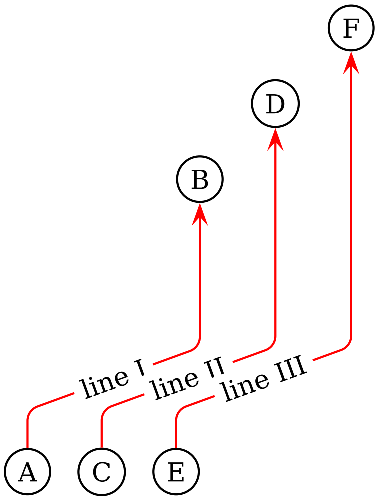

```
\begin{pspicture}[showgrid=true,arrowscale
  =2](5,5)
\pnode(4.5,0.5){MyNode}
\psdot(MyNode)
\pnode(! \psGetNodeCenter{MyNode}
   MyNode.x 4 sub MyNode.y 4 add){MySecondNode}
\psdot(MySecondNode)
\ncline[linecolor=red]{<->}{MyNode}{
  MySecondNode}
\end{pspicture}
```

16. \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

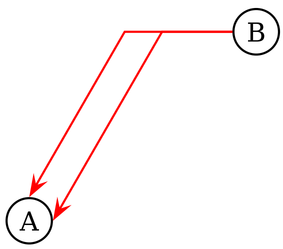
```
\ncdiag[<options>]{<Node A>}{<Node B>}
\pcdiag[<options>](<Node A>)(<Node B>)
```

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

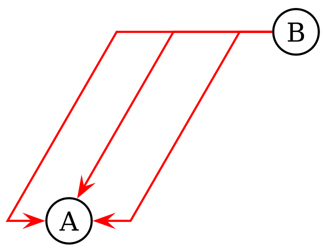



```
\begin{pspicture}(5,6)
  \circlenode{A}{A}\quad\circlenode{C}{C}%
    \quad\circlenode{E}{E}
  \rput(0,4){\circlenode{B}{B}}
  \rput(1,5){\circlenode{D}{D}}
  \rput(2,6){\circlenode{F}{F}}
  \psset{arrowscale=2,linearc=0.2,%
    linecolor=red,armA=0.5, angleA=90,angleB
      =-90}
  \ncdiag[lineAngle=20]{->}{A}{B}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

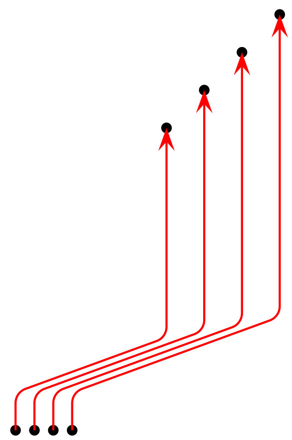
The `\ncdiag` macro sets the `armB` dynamically to the calculated value. Any user setting of `armB` is overwritten by the macro. The `armA` could be set to a zero length:




```
\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}
```

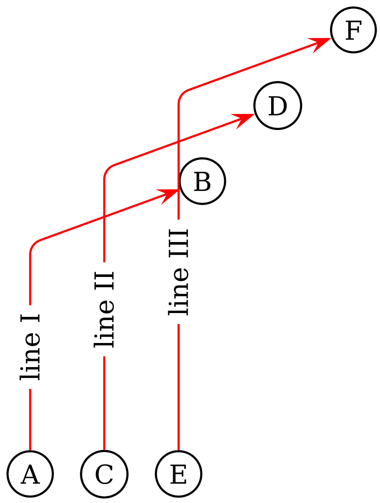


```
\begin{pspicture}(4,5.5)
  \cnode*(0,0){2pt}{A}%
  \cnode*(0.25,0){2pt}{C}%
  \cnode*(0.5,0){2pt}{E}%
  \cnode*(0.75,0){2pt}{G}%
  \cnode*(2,4){2pt}{B}%
  \cnode*(2.5,4.5){2pt}{D}%
  \cnode*(3,5){2pt}{F}%
  \cnode*(3.5,5.5){2pt}{H}%
  {\psset{arrowscale=2,linearc=0.2,%
    linecolor=red,armA=0.5, 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}
```

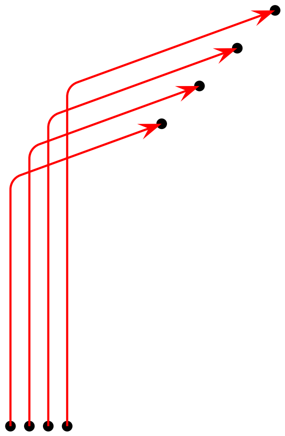
17. \ncdiagg and \pcdiagg

This is nearly the same as \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>)
```

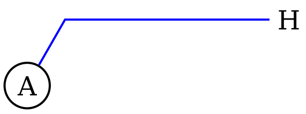



```
\begin{pspicture}(4,6)
  \psset{linecolor=black}
  \circlenode{A}{A}%
  \quad\circlenode{C}{C}%
  \quad\circlenode{E}{E}
  \rput(0,4){\circlenode{B}{B}}
  \rput(1,5){\circlenode{D}{D}}
  \rput(2,6){\circlenode{F}{F}}
  {\psset{arrowscale=2,linearc=0.2,linecolor=red
    ,armA=0.5, angleA=90}
  \ncdiagg[lineAngle=-160]{->}{A}{B}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



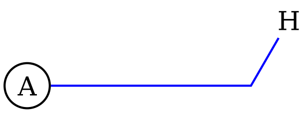
```
\begin{pspicture}(4,6)
  \psset{linecolor=black}
  \cnode*(0,0){2pt}{A}%
  \cnode*(0.25,0){2pt}{C}%
  \cnode*(0.5,0){2pt}{E}%
  \cnode*(0.75,0){2pt}{G}%
  \cnode*(2,4){2pt}{B}%
  \cnode*(2.5,4.5){2pt}{D}%
  \cnode*(3,5){2pt}{F}%
  \cnode*(3.5,5.5){2pt}{H}%
  {\psset{arrowscale=2,linearc=0.2,linecolor=red
    ,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, which differs by 180.




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

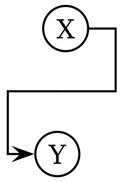

```
\begin{pspicture}(4,1.5)
  \circlenode{a}{A}
  \rput[l](3,1){\rnode{b}{H}}
  \ncdiagg[lineAngle=60,armA=.5,nodesepB=3pt,
    linecolor=blue]{a}{b}
\end{pspicture}
```



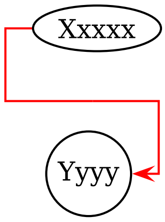

```
\begin{pspicture}(4,1.5)
  \circlenode{a}{A}
  \rput[l](3,1){\rnode{b}{H}}
  \ncdiagg[lineAngle=-120,armA=.5,nodesepB=3pt,
    linecolor=blue]{a}{b}
\end{pspicture}
```

18. \ncbarr

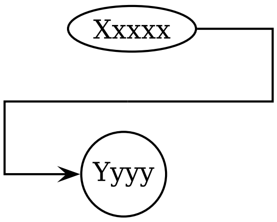
This has the same behaviour as `\ncbar`, but has 5 segments and all are horizontal ones. This is the reason why `angleA` must be 0 or alternatively 180. All other values are set to 0 by the macro. The intermediate horizontal line is symmetrical to the distance of the two nodes.




```
\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}
```

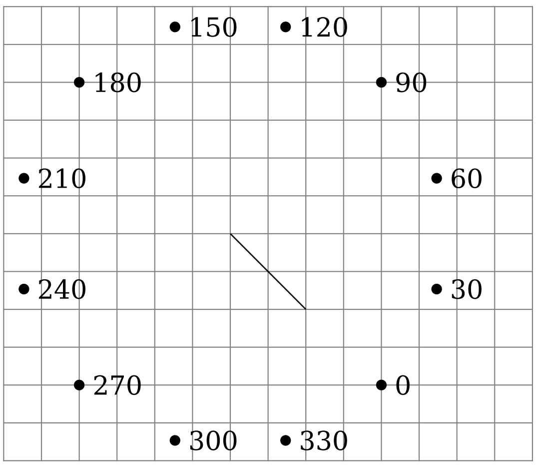
19. \psRelNode and \psDefPSPNodes

With these macros it is possible to put a node relative to a given line or given `pspicture`-environment. In the frist case the parameters are the angle and the length factor:

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

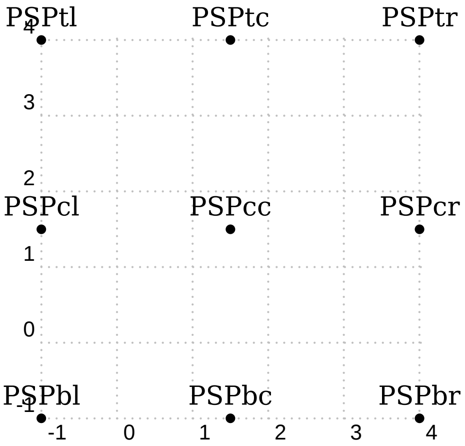
The length factor relates to the distance $\overline{P_0P_1}$ and the end node name must be a valid nodename and shouldn't contain any of the special PostScript characters. There are two valid options:

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 refers 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}
```

In the second case the new macro \psDefPSPNodes defines nine nodes that corresponds to nine particular points (namely bottom left, bottom center, bottom right, center left, center center, center right, top left, top center, top right) of the pspicture box.




```
\begin{pspicture}[showgrid=true](-1,-1)
  (4,4)
  \psDefPSPNodes
  \psdots(PSPbl)(PSPbc)(PSPbr)
     (PSPcl)(PSPcc)(PSPcr)(PSPtl)(PSPtc)(
       PSPtr)
  \uput[90](PSPbl){PSPbl} \uput[90](PSPbc){
    PSPbc}
  \uput[90](PSPbr){PSPbr} \uput[90](PSPcl){
    PSPcl}
  \uput[90](PSPcc){PSPcc} \uput[90](PSPcr){
    PSPcr}
  \uput[90](PSPtl){PSPtl} \uput[90](PSPtc){
    PSPtc}
  \uput[90](PSPtr){PSPtr}
\end{pspicture}
```

The name of the nodes are predefined as:

```
\psset[pst-PSPNodes]{blName=PSPbl,bcName=PSPbc,brName=PSPbr,
 clName=PSPcl,ccName=PSPcc,crName=PSPcr,tlName=PSPtl,tcName=PSPtc,trName=PSPtr
   }
```

and can be modified in the same way.

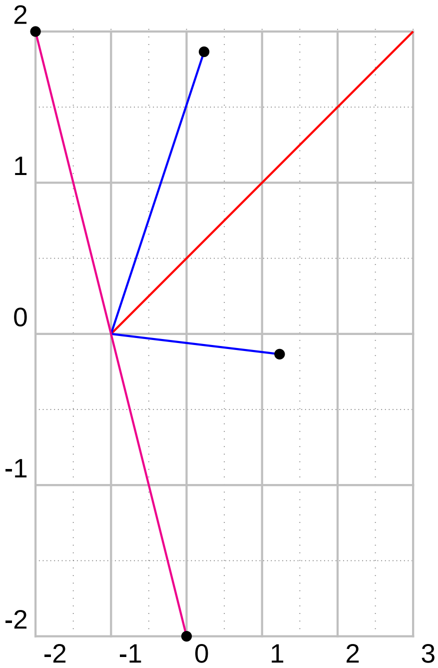
20. \psRelLine

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

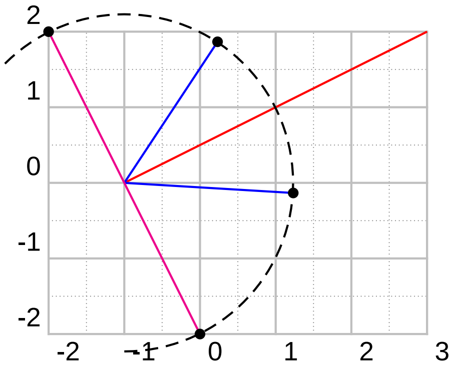
\psRelLine(*P0*)(*P1*){*length factor*}{*<end node name>*}
\psRelLine [{*arrows*}] (*P0*)(*P1*){*length factor*}{*end node name*}
\psRelLine [Options] (*P0*)(*P1*){*length factor*}{*end node name*}
\psRelLine [Options] [{*arrows*}] (*P0*)(*P1*){*length factor*}{*end node name*}

The length factor relates to the distance $\overline{P_0P_1}$ 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 foregoing 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.

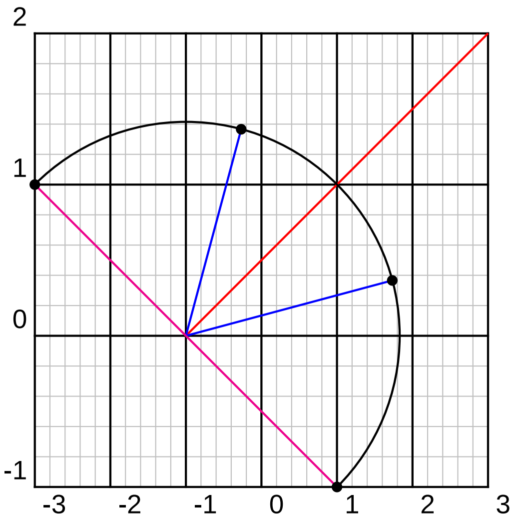



```
\psset{yunit=2,xunit=1}
\begin{pspicture}(-2,-2)(3,2)
\psgrid[subgriddiv=2,subgriddots=10,gridcolor=
  lightgray]
\pnode(-1,0){A}\pnode(3,2){B}
\psline[linecolor=red](A)(B)
\psRelLine[linecolor=blue,angle=30](-1,0)(B)
  {0.5}{EndNode}
\qdisk(EndNode){2pt}
\psRelLine[linecolor=blue,angle=-30](A)(B)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



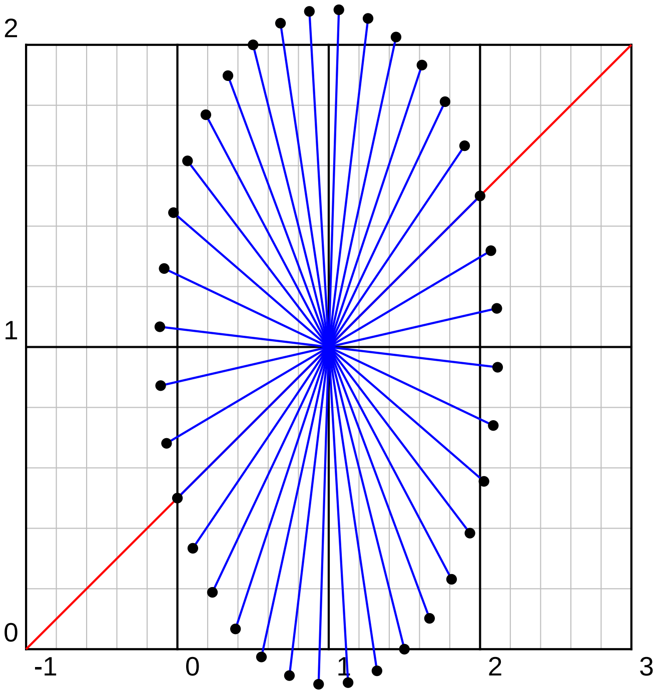

```
\begin{pspicture}(-2,-2)(3,2)
\psgrid[subgriddiv=2,subgriddots=10,gridcolor=
  lightgray]
\pnode(-1,0){A}\pnode(3,2){B}
\psline[linecolor=red](A)(B)
\psarc[linestyle=dashed](A){2.23}{-90}{135}
\psRelLine[linecolor=blue,angle=30](-1,0)(B)
  {0.5}{EndNode}
\qdisk(EndNode){2pt}
\psRelLine[linecolor=blue,angle=-30](A)(B)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

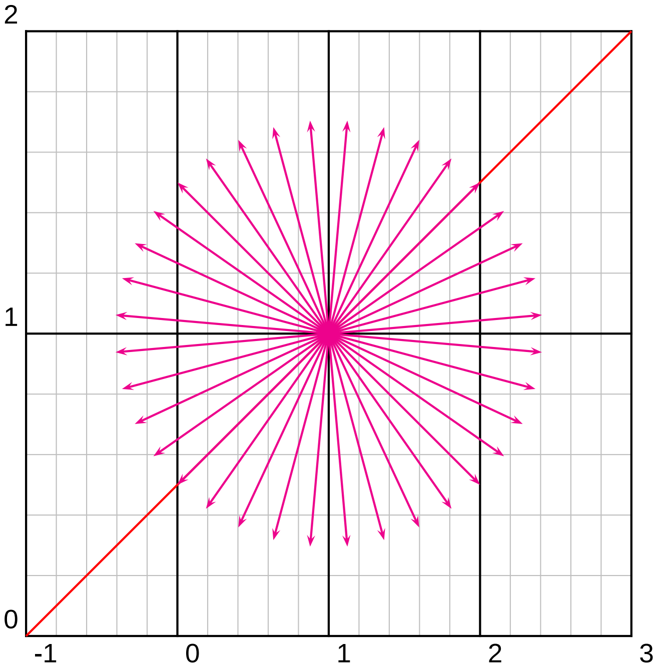



```
\psset{yunit=2,xunit=1}
\begin{pspicture}(-3,-1)(3,2)\psgrid[subgridcolor=lightgray]
\pnode(-1,0){A}\pnode(3,2){B}
\psline[linecolor=red](A)(B)
\psarc(A){2.83}{-45}{135}
\psRelLine[linecolor=blue,angle=30,trueAngle](A)(B){0.5}{EndNode}
\qdisk(EndNode){2pt}
\psRelLine[linecolor=blue,angle=-30,trueAngle](A)(B){0.5}{EndNode}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

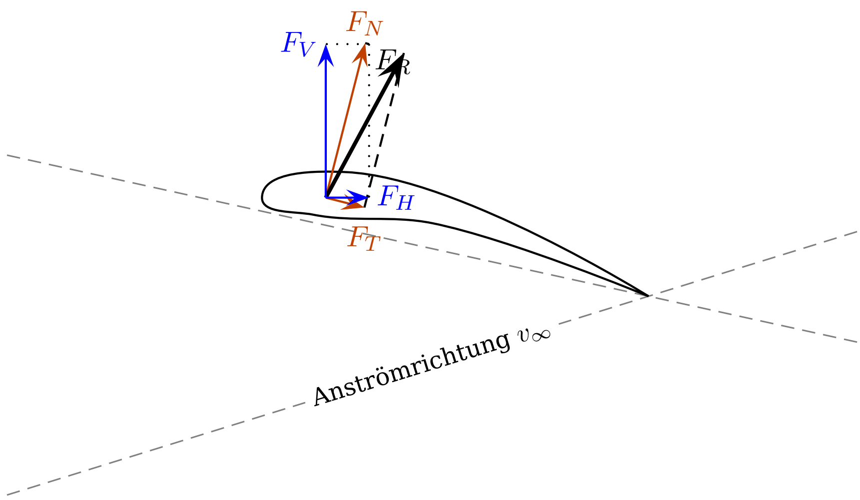
Two examples using `\multido` to show the behaviour of the options `trueAngle` and `angle`.




```
\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"?
```

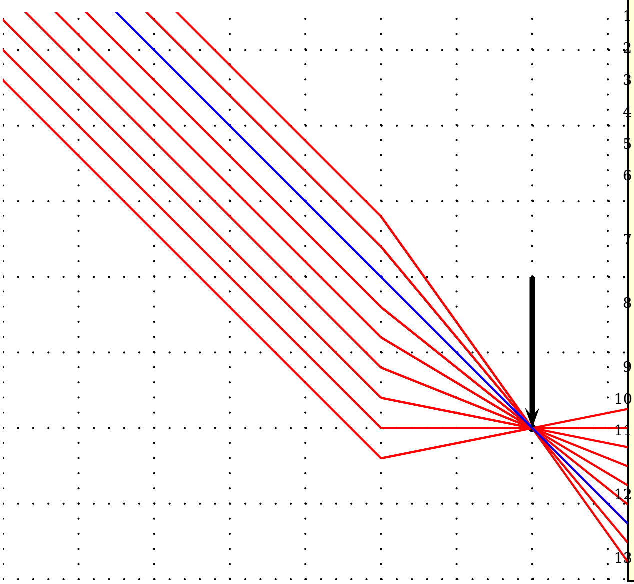
```
  \psParallelLine[linestyle=dashed](D)(FtE)(Fn){.1}{Fnr1}
  \psRelLine[linestyle=dashed,angle=90](FtE)(D){-4}{Fnr2} % why "-4"?
  \psline[linewidth=1.5pt,arrows=->,arrowscale=2](D)(Fnr2)
  \psIntersectionPoint(D)([nodesep=2]D)(Fnr1)([offset=-4]Fnr1){Fh}
  \psIntersectionPoint(D)([offset=2]D)(Fnr1)([nodesep=4]Fnr1){Fv}
  \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)(Fnr1)
  \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}
```

21. \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 on the length factor. The end node name must be a valid nodename and shouldn't contain any of the special PostScript characters.

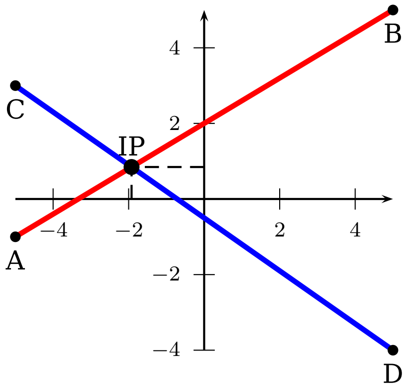


```
\begin{pspicture*}(-5,-4)(5,3.5)
  \psgrid[subgriddiv=0,griddots=5]
  \pnode(2,-2){FF}\qdisk(FF){1.5pt}
  \pnode(-5,5){A}\pnode(0,0){O}
  \multido{\nCountA=-2.4+0.4}{9}{%
    \psParallelLine[linecolor=red](O)(A)(0,\nCountA){9}{P1}
    \psline[linecolor=red](0,\nCountA)(FF)
    \psRelLine[linecolor=red](0,\nCountA)(FF){9}{P2}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

22. \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>}
```

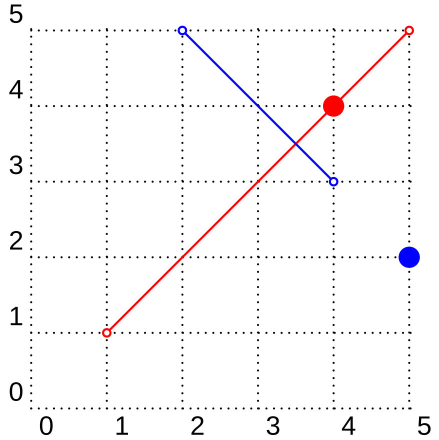



```
\psset{unit=0.5cm}
\begin{pspicture}(-5,-4)(5,5)
 \psaxes[labelFontSize=\scriptstyle,
  dx=2,Dx=2,dy=2,Dy=2]{->}(0,0)(-5,-4)(5,5)
 \psline[linecolor=red,linewidth=2pt](-5,-1)(5,5)
 \psline[linecolor=blue,linewidth=2pt](-5,3)(5,-4)
 \qdisk(-5,-1){2pt}\uput[-90](-5,-1){A}
 \qdisk(5,5){2pt}\uput[-90](5,5){B}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

23. \psLNode and \psLCNode

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

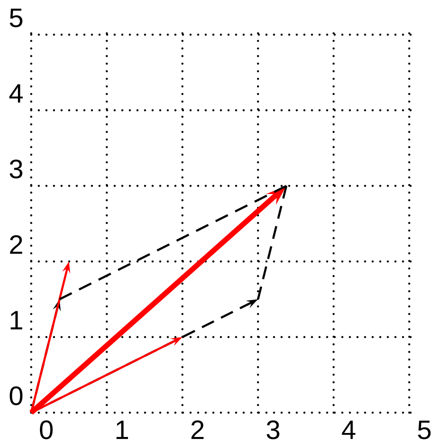
```
\psLNode(P1)(P2){value}{Node name}
```




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

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

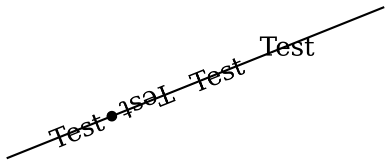



```
\begin{pspicture}(5,5)
\psgrid[subgriddiv=0,griddots=10]
\psset{linecolor=black}
\psline[linestyle=dashed]{->}(3,1.5)
\psline[linestyle=dashed]{->}(0.375,1.5)
\psset{linecolor=red}
\psline{->}(2,1)\psline{->}(0.5,2)
\psLCNode(2,1){1.5}(0.5,2){0.75}{PI}
\psline[linewidth=2pt]{->}(PI)
\psset{linecolor=black}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

24. \nlput and \psLDNode

\ncput allows you 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 macros. It uses internally the macro \psLDNode which places a node absolute to two given points, starting from the first one.

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



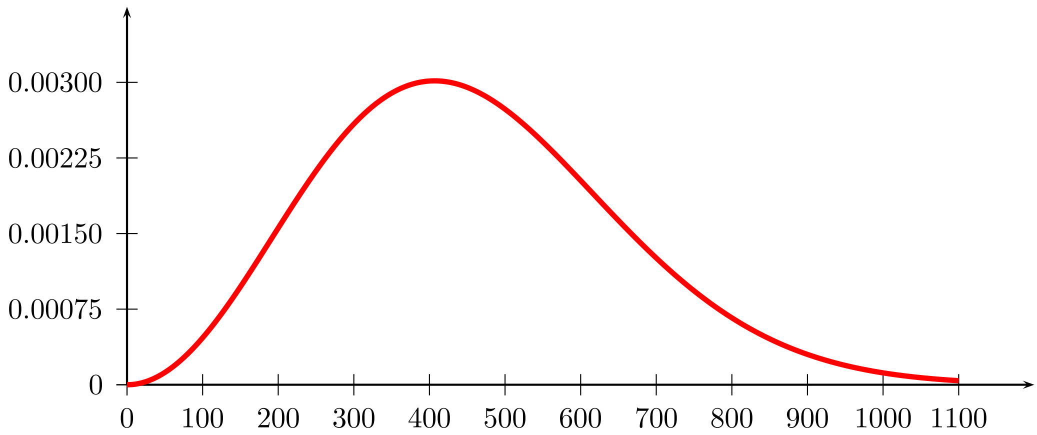

```
\begin{pspicture}(5,2)
\pnode(0,0){A}
\pnode(5,2){B}
\ncline{A}{B}
\psLDNode(A)(B){1.5cm}{KN}\qdisk(KN){2pt}
\nlput[nrot=:U](A)(B){1cm}{Test}
\nlput[nrot=:D](A)(B){2cm}{Test}
\nlput[nrot=:U](A)(B){3cm}{Test}
\nlput(A)(B){4cm}{Test}
\end{pspicture}
```

Part III. pst-plot

25. New syntax

There is now a new optional argument for \psplot and \parametricplot to pass additional PostScript commands into the code. This makes the use of \pstVerb in most cases superfluous.

\psplot [Options] {*x0*}{*x1*} [PS commands] {*function*}
\parametricplot [Options] {*t0*}{*t1*} [PS commands] {*x(t) y(t)*}




```
\begin{pspicture}(0,-0.5)(12,5)
 \psaxes[Dx=100,dx=1,Dy=0.00075,dy=1]{->}(0,0)(12,5)
 \psplot[linecolor=red, plotstyle=curve,linewidth=2pt,plotpoints=200]{0}{11}%
  [ /const1 3.3 10 8 neg exp mul def /s 10 def /const2 6.04 10 6 neg exp mul def ]%
  { const1 x 100 mul dup mul mul Euler const2 neg x 100 mul dup mul mul exp mul 2000
    mul}
\end{pspicture}
```

26. New or extended options

The axes macro has now two additional optional arguments for placing labels at the end of the axes:

```
\psaxes[settings]{arrows}(x0,y0)(x1,y1)(x2,y2)[Xlabel,Xangle][Ylabel,Yangle]
```

It has now four optional arguments, one for the setting, one for the arrows, one for the x-label and one for the y-label. If you want only a y-label, then leave the x one empty. A missing y-label is possible. The following examples show how it can be used.

The option `tickstyle=full |top|bottom` no longer works in the usual way. Only the additional value `inner` is valid for `pstricks-add`, because everything can be set by the `ticksize` option. When using the `comma` or `trigLabels` option, the macros \pshlabel and \psvlabel shouldn't be redefined, because the package does it itself internally in these cases. However, if you need a redefinition, then do it for `\pst@@hlabel` and `\pst@@vlabel` with

```
\makeatletter
\def\ps@@hlabel#1{...}
\def\ps@@vlabel#1{...}
\makeatother
```

Table 2: All new parameters for `pst-plot`

<table>
<tr><th>Name</th><th>Type</th><th>Default</th></tr>
<tr><td>axesstyle</td><td><none|axes|frame|polar></td><td>axes</td></tr>
<tr><td>labels</td><td><all|x|y|none></td><td>all</td></tr>
<tr><td>xlabelPos</td><td><bottom,axis,top></td><td>bottom</td></tr>
<tr><td>ylabelPos</td><td><left,axis,right></td><td>left</td></tr>
<tr><td>xlabelFactor</td><td><anything></td><td>{\ empty}</td></tr>
<tr><td>ylabelFactor</td><td><anything></td><td>{\ empty}</td></tr>
<tr><td>labelFontSize</td><td><fontsize macro></td><td>{}</td></tr>
<tr><td>trigLabels</td><td>false|true</td><td>false</td></tr>
<tr><td>trigLabelBase</td><td><number></td><td>0</td></tr>
<tr><td>algebraic</td><td>false|true</td><td>false</td></tr>
<tr><td>decimalSeparator</td><td><character></td><td>.</td></tr>
<tr><td>comma</td><td>false|true</td><td>false</td></tr>
<tr><td>xAxis</td><td>false|true</td><td>true</td></tr>
<tr><td>yAxis</td><td>false|true</td><td>true</td></tr>
<tr><td>xyAxes</td><td>false|true</td><td>true</td></tr>
<tr><td>xDecimals</td><td><number> or empty</td><td>{}</td></tr>
<tr><td>yDecimals</td><td><number> or empty</td><td>{}</td></tr>
<tr><td>xyDecimals</td><td><number> or empty</td><td>{}</td></tr>
<tr><td>ticks</td><td><all|x|y|none></td><td>all</td></tr>
<tr><td>tickstyle</td><td>full|top|bottom|inner</td><td>full</td></tr>
<tr><td>subticks</td><td><number></td><td>0</td></tr>
<tr><td>xsubticks</td><td><number></td><td>0</td></tr>
<tr><td>ysubticks</td><td><number></td><td>0</td></tr>
<tr><td>ticksize</td><td><length [length]></td><td>-4pt 4pt</td></tr>
<tr><td>subticksize</td><td><number></td><td>0.75</td></tr>
<tr><td>tickwidth</td><td><length></td><td>0.5\pslinewidth</td></tr>
<tr><td>subtickwidth</td><td><length></td><td>0.25\pslinewidth</td></tr>
<tr><td>tickcolor</td><td><color></td><td>black</td></tr>
<tr><td>xtickcolor</td><td><color></td><td>black</td></tr>
<tr><td>ytickcolor</td><td><color></td><td>black</td></tr>
<tr><td>subtickcolor</td><td><color></td><td>darkgray</td></tr>
<tr><td>xsubtickcolor</td><td><color></td><td>darkgray</td></tr>
<tr><td>ysubtickcolor</td><td><color></td><td>darkgray</td></tr>
<tr><td>ticklinestyle</td><td>solid | dashed | dotted | none</td><td>solid</td></tr>
<tr><td>subticklinestyle</td><td>solid | dashed | dotted | none</td><td>solid</td></tr>
<tr><td>xlogBase</td><td><number> or empty</td><td>{}</td></tr>
</table>

Name	Type	Default
ylogBase	<number> or empty	{}
xylogBase	<number> or empty	{}
logLines	<none|x|y|all>	none
yMaxValue	<real>	-1
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
ChangeOrder	false|true	false

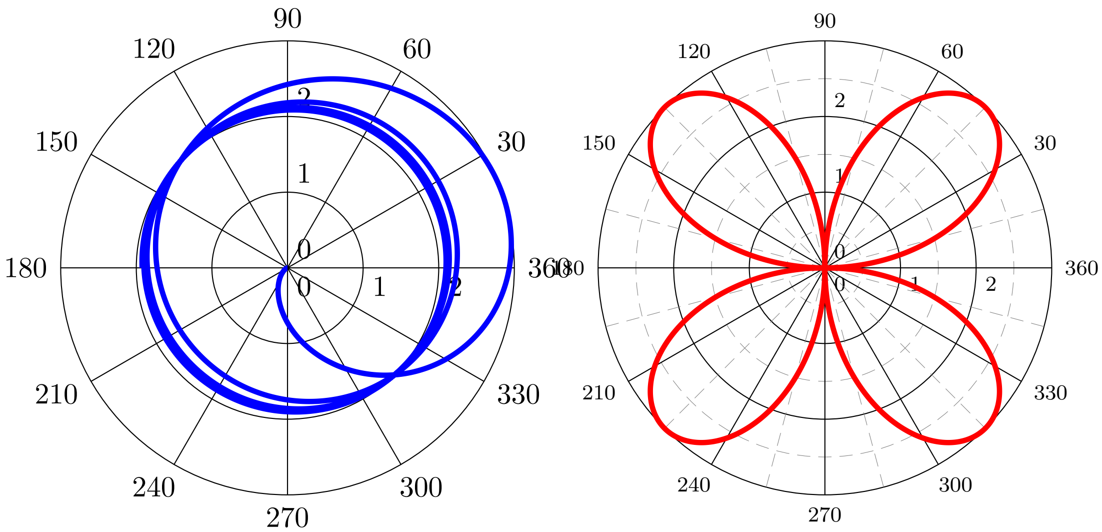
26.1. axesstyle

There is a new axes style `polar` which plots a polar coordinate system.

Syntax:

```
\psplot[axesstyle=polar](Rx,Ry)
\psplot[axesstyle=polar](...)(Rx,Ry)
\psplot[axesstyle=polar](...)(...)(Rx,Ry)
```

Important is the fact, that only one pair of coordinates is taken into account for the radius. It is *always* the last pair in a sequence of allowed coordinates for the `\psaxes` macro. The other ones are ignored; they are not valid for the polar coordinate system.




```
%\usepackage{pstricks-add}

\begin{pspicture}(-3.5,-3.5)(3.5,3.5)
\psaxes[axesstyle=polar](3,3)
\psplot[polarplot,algebraic,linecolor=blue,linewidth=2pt,
   plotpoints=2000]{0}{TwoPi 4 mul}{2*(sin(x)-x)/(cos(x)+x)}
\end{pspicture}
%
\begin{pspicture}(-3.5,-3.5)(3.5,3.5)
\psaxes[axesstyle=polar,subticklinestyle=dashed,subticks=2,
  labelFontSize=\scriptstyle](3,3)
\psplot[polarplot,algebraic,linecolor=red,linewidth=2pt,
   plotpoints=2000]{0}{TwoPi}{6*sin(x)*cos(x)}
\end{pspicture}
```

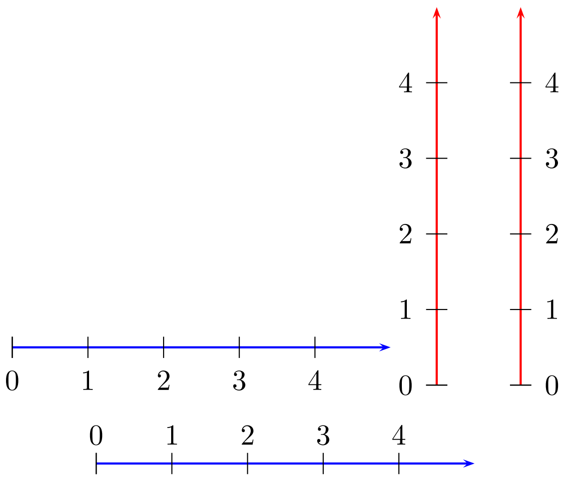
All valid optional arguments for the axes are also possible for the polar style, if they make sense ... :-) Important are the `Dy` option, it defines the angle interval and `subticks`, for the intermediate circles and lines. The number can be different for the circles (`ysubticks`) and the lines (`xsubticks`).

26.2. 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 preceding options to false. The `xyAxes` only makes sense when you want to set both x and y to true with only one command, back 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,
   ylabelPos=right]{->}(0.5,0)(0.5,5)
\end{pspicture}\\[0.5cm]
\begin{pspicture}(5,1)
\psaxes[yAxis=false,linecolor=blue,
   xlabelPos=top]{->}(0,0.5)(5,0.5)
\end{pspicture}
```

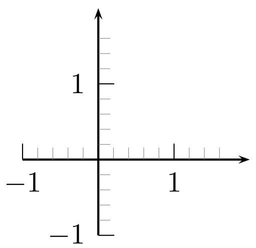
As seen in the example, a single y axis gets the labels on the left side. This can be changed with the option `ylabelPos` or with `xlabelPos` for the x-axis.

26.3. labels

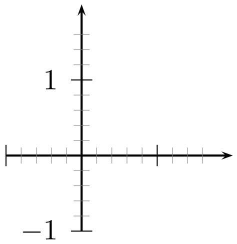
Syntax:

```
labels=all|x|y|none
```

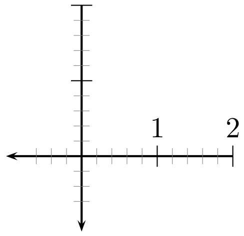
This option is also already in the `pst-plot` package and only mentioned here for completeness.



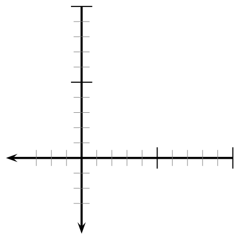

```
\psset{ticksize=6pt}
\begin{pspicture}(-1,-1)(2,2)
\psaxes[labels=all,subticks=5]{->}(0,0)(-1,-1)(2,2)
\end{pspicture}
```




```
\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}
```



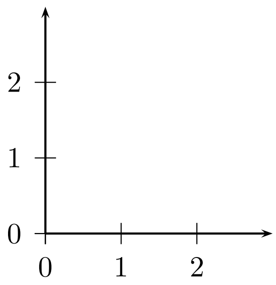
```
\begin{pspicture}(-1,-1)(2,2)
\psaxes[labels=none,subticks=5]{->}(0,0)(2,2)(-1,-1)
\end{pspicture}
```

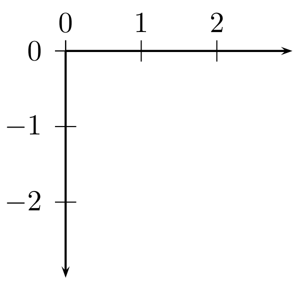
26.4. xlabelPos and ylabelPos

Syntax:

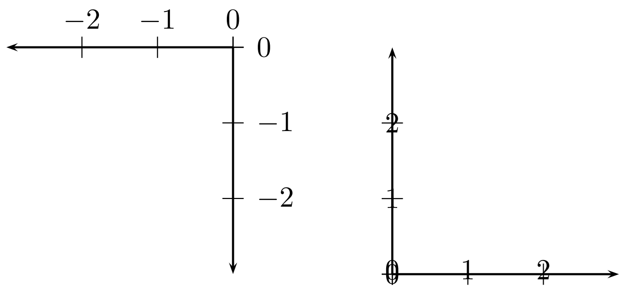
```
xlabelPos=bottom|axis|top
ylabelPos=left|axis|right
```

By default the labels for ticks are placed at the bottom (x axis) and left (y-axis). If both axes are drawn in the negative direction the default is top (x axis) and right (y axis). It be changed with the two options `xlabelPos` and `ylabelPos`. With the value `axis` the user can place the labels depending on the value of `labelsep`, which is taken into account for `axis`.

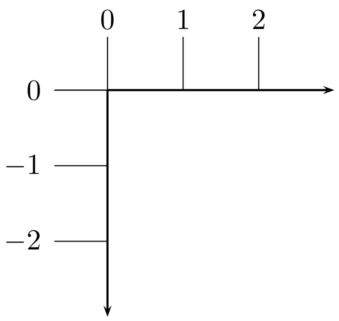





```
\begin{pspicture}(3,3)
\psaxes{->}(3,3)
\end{pspicture}\hspace{2cm
  }
\begin{pspicture}(3,-3)
\psaxes[xlabelPos=top
  ]{->}(3,-3)
\end{pspicture}
```




```
\begin{pspicture}(-3,-3)
\psaxes{->}(-3,-3)
\end{pspicture}\hspace{2cm}
\begin{pspicture}(3,3)
\psaxes[labelsep=0pt,
   ylabelPos=axis,
   xlabelPos=axis]{->}(3,3)
\end{pspicture}
```



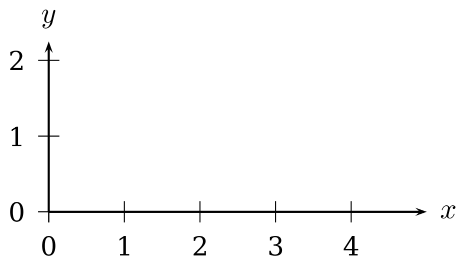

```
\begin{pspicture}(-1,1)(3,-3)
\psaxes[xlabelPos=top,
   xticksize=0 20pt,
   yticksize=-20pt 0]{->}(3,-3)
\end{pspicture}
```

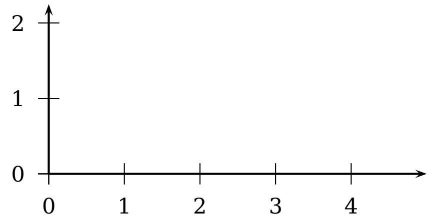
26.5. Changing the label font size with labelFontSize and mathLabel

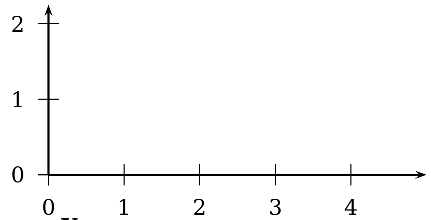
This option sets the horizontal **and** vertical font size for the labels depending on the option mathLabel for the text or the math mode. It will be overwritten when another package or a user defines

```
\def\pshlabel#1{\labelFontSize ...}
\def\psvlabel#1{\labelFontSize ...}
\def\pshlabel#1{$\labelFontSize ...$}% for mathLabel=true (default)
\def\psvlabel#1{$\labelFontSize ...$}% for mathLabel=true (default)
```

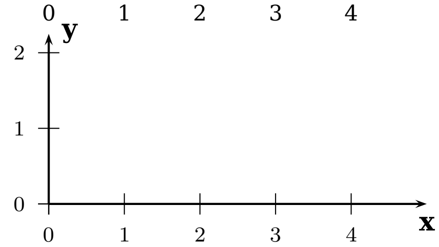
in another way. Note that for mathLabel=truethe font size must be set by one of the mathematical styles \textstyle, \displaystyle, \scriptstyle, or \scriptscriptstyle.

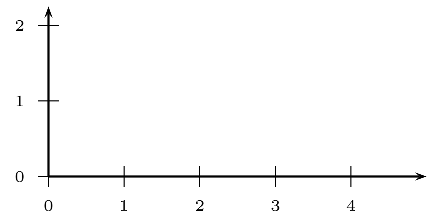







```
\psset{mathLabel=false}
\begin{pspicture}(-0.25,-0.25)(5,2.25)
\psaxes{->}(5,2.25)[$x$,0][$y$,90]
\end{pspicture}\\[20pt]
\begin{pspicture}(-0.25,-0.25)(5,2.25)
\psaxes[labelFontSize=\footnotesize
  ]{->}(5,2.25)
\end{pspicture}\\[20pt]
\begin{pspicture}(-0.25,-0.25)(5,2.25)
\psaxes[labelFontSize=\footnotesize
  ]{->}(5,2.25)
\end{pspicture}\\[20pt]
```

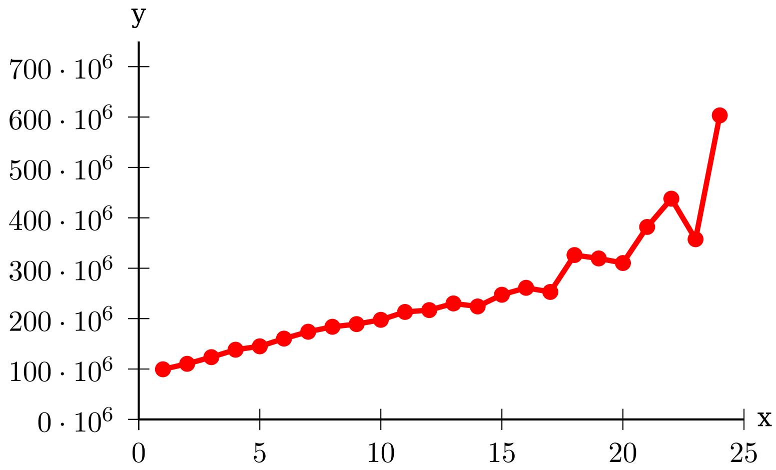





```
\begin{pspicture}(-0.25,-0.25)(5,2.25)
\psaxes[labelFontSize=\scriptstyle
  ]{->}(5,2.25)[\textbf{x},-90][\textbf{y
  },0]
\end{pspicture}\\[20pt]
\psset{mathLabel=true}
\begin{pspicture}(-0.25,-0.25)(5,2.25)
\psaxes[labelFontSize=\scriptscriptstyle
  ]{->}(5,2.25)
\end{pspicture}\\[20pt]
```

26.6. 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 you to define the additional part of the value, but it must be set in math mode when using math operators!



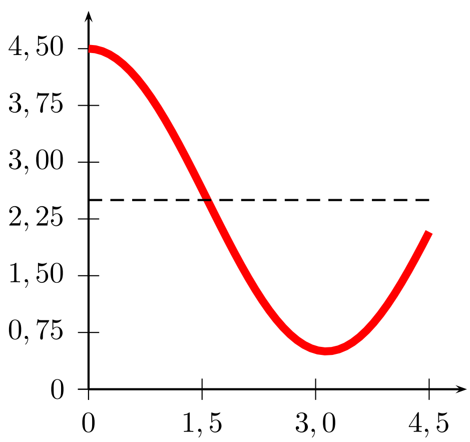

```
\readdata{\data}{demo1.data}
\pstScalePoints(1,0.000001){}{}% (x,y){additional x operator}{y op}
\psset{llx=-1cm,lly=-1cm}
\psgraph[ylabelFactor=\cdot 10^6,Dx=5,Dy=100](0,0)(25,750){8cm}{5cm}
  \listplot[linecolor=red, linewidth=2pt, showpoints=true]{\data}
\endpsgraph
\pstScalePoints(1,1){}{}% reset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

26.7. decimalSeparator and comma

Syntax:

```
comma=false|true
decimalSeparator=<charactor>
```

Setting the option comma to true gives labels with a comma as a decimal separator instead of the dot. comma and comma=true is the same. The optional argument decimalSeparator allows an individual setting for languages with a different character than a dot or a comma. The character has to set into braces, if it is an active, e. g. decimalSeparator={,}.



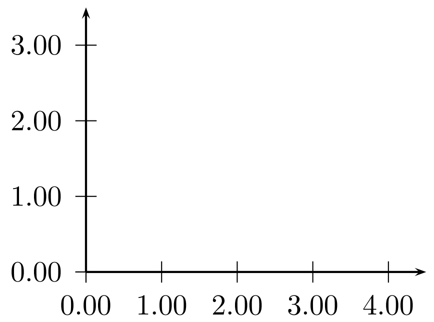

```
\begin{pspicture}(-0.5,-0.5)(5,5.5)
\psaxes[Dx=1.5,comma,Dy=0.75,dy
  =0.75]{->}(5,5)
\psplot[linecolor=red,linewidth=3pt]{0}{4.5}%
  {x RadtoDeg cos 2 mul 2.5 add}
\psline[linestyle=dashed](0,2.5)(4.5,2.5)
\end{pspicture}
```

26.8. xyDecimals, xDecimals and yDecimals

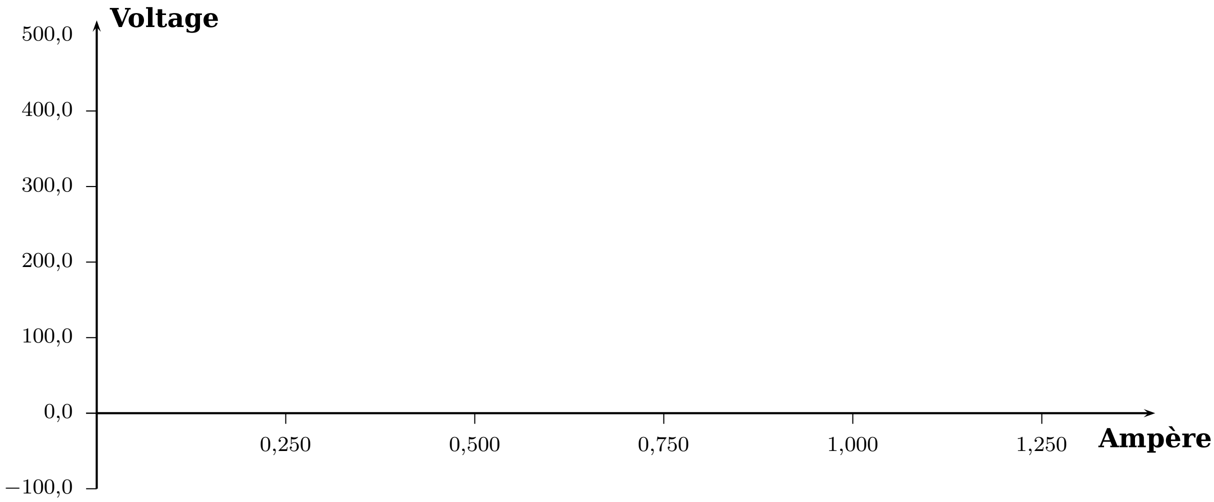
Syntax:

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

By default the labels of the axes get numbers with or without decimals, depending on 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)(5,3.75)
  \psaxes[xyDecimals=2]{->}(0,0)(4.5,3.5)
\end{pspicture}
```




```
\psset{xunit=10cm,yunit=0.01cm,labelFontSize=\scriptstyle}
\begin{pspicture}(-0.1,-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)[\textbf{Amp\'ere},-90][\textbf{Voltage},0]
\end{pspicture}
```

26.9. trigLabels and trigLabelBase – axis with trigonmetrical units

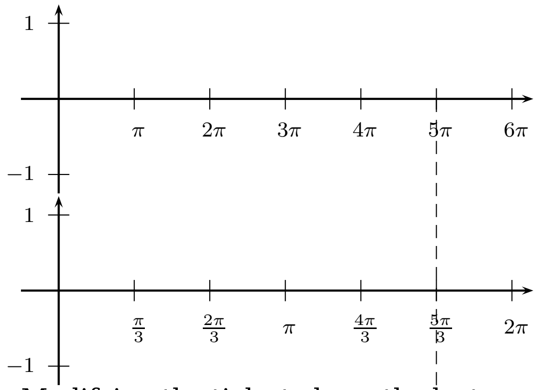
With the option `trigLabels=true` the labels on the x axis are trigonometrical ones. The option `trigLabelBase` set the denominator of fraction. The default value of 0 is the same as no fraction. The following constants are defined in the package:

```
\def\psPiFour{12.566371}
\def\psPiTwo{6.283185}
\def\psPi{3.14159265}
\def\psPiH{1.570796327}
\newdimen\pstRadUnit
\newdimen\pstRadUnitInv
\pstRadUnit=1.047198cm % this is pi/3
\pstRadUnitInv=0.95493cm % this is 3/pi
```

Because it is a bit complicated to set the right values, we show some more examples here.

For **all** following examples in this section we did a global `\psset{trigLabels=true,labelFontSize=\scriptstyle}`.

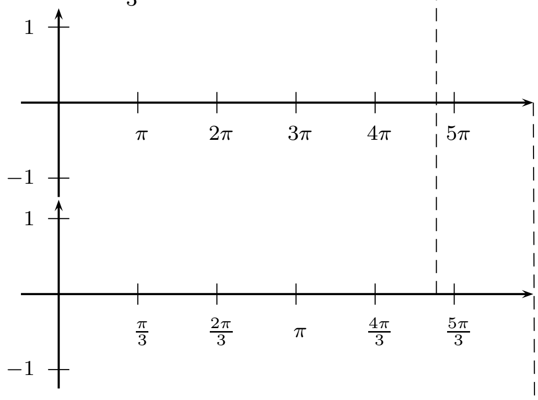
Translating the decimal ticks to trigonometrical ones makes no real sense, because every 1 xunit (1cm) is a tick and the last one is at 6cm.




```
\begin{pspicture}(-0.5,-1.25)(6.5,1.25)%
  \pnode(5,0){A}%
  \psaxes{->}(0,0)(-.5,-1.25)(\psPiTwo,1.25)
\end{pspicture}
```

```
\begin{pspicture}(-0.5,-1.25)(6.5,1.25)%
  \psaxes[trigLabelBase=3]{->}(0,0)(-0.5,-1.25)
    (\psPiTwo,1.25)
\end{pspicture}
```

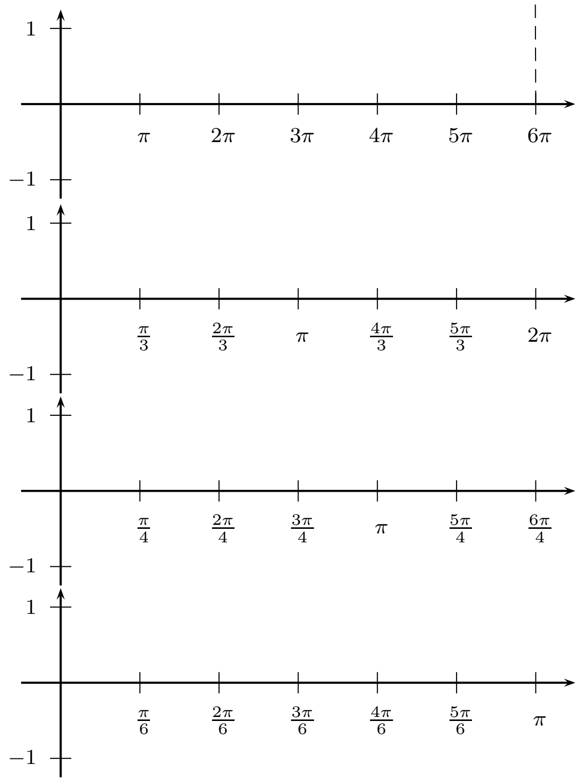
Modifying the ticks to have the last one exactly at the end is possible with a different dx value ($\frac{\pi}{3} \approx 1.047$):




```
\begin{pspicture}(-0.5,-1.25)(6.5,1.25)\pnode
  (\psPiTwo,0){C}%
  \psaxes[dx=\pstRadUnit]{->}(0,0)(-0.5,-1.25)
    (\psPiTwo,1.25)
\end{pspicture}%
```

```
\begin{pspicture}(-0.5,-1.25)(6.5,1.25)\pnode
  (5,0){B}%
  \psaxes[dx=\pstRadUnit,trigLabelBase=3]
    {->}(0,0)(-0.5,-1.25)(\psPiTwo,1.25)
\end{pspicture}%
```

Set everything globally in radian units. Now 6 units on the x-axis are 6π. Using `trigLabelBase=3` reduces this value to 2π, a.s.o.



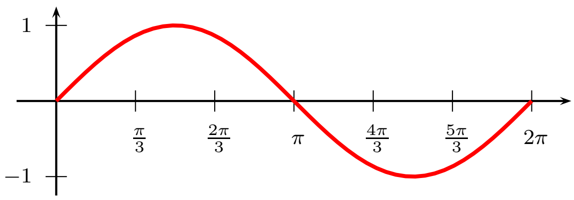

```
\psset{xunit=\pstRadUnit}%
\begin{pspicture}(-0.5,-1.25)(6.5,1.25)\pnode
  (6,0){D}%
  \psaxes{->}(0,0)(-0.5,-1.25)(6.5,1.25)%
\end{pspicture}%
```

```
\psset{xunit=\pstRadUnit}%
\begin{pspicture}(-0.5,-1.25)(6.5,1.25)
  \psaxes[trigLabelBase=3]{->}(0,0)(-0.5,-1.25)
    (6.5,1.25)
\end{pspicture}%
```

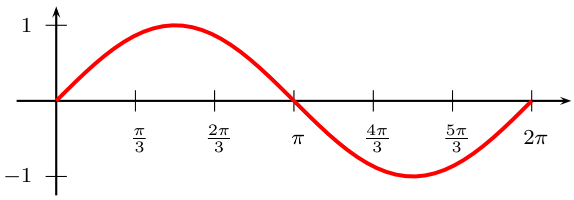
```
\psset{xunit=\pstRadUnit}%
\begin{pspicture}(-0.5,-1.25)(6.5,1.25)
  \psaxes[trigLabelBase=4]{->}(0,0)(-0.5,-1.25)
    (6.5,1.25)
\end{pspicture}%
```

```
\psset{xunit=\pstRadUnit}%
\begin{pspicture}(-0.5,-1.25)(6.5,1.25)
  \psaxes[trigLabelBase=6]{->}(0,0)(-0.5,-1.25)
    (6.5,1.25)
\end{pspicture}%
```

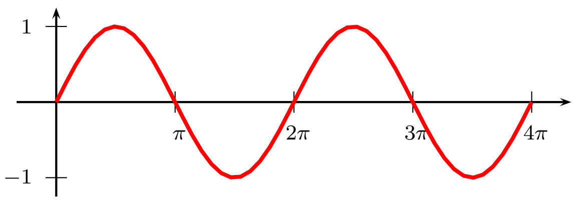
The best way seems to be to set the x-unit to `\pstRadUnit`. Plotting a function doesn't consider the value for `trigLabelBase`, it has to be done by the user. The first example sets the unit locally for the `\psplot` back to 1cm, which is needed, because we use this unit on the PostScript side.



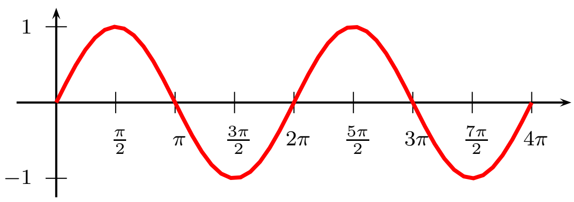

```
\psset{xunit=\pstRadUnit}%
\begin{pspicture}(-0.5,-1.25)(6.5,1.25)
  \psaxes[trigLabelBase=3]{->}(0,0)(-0.5,-1.25)
    (6.5,1.25)
  \psplot[xunit=1cm,linecolor=red,linewidth=1.5
    pt]{0}{\psPiTwo}{x RadtoDeg sin}
\end{pspicture}
```




```
\psset{xunit=\pstRadUnit}%
\begin{pspicture}(-0.5,-1.25)(6.5,1.25)
  \psaxes[trigLabelBase=3]{->}(0,0)(-0.5,-1.25)
    (6.5,1.25)
  \psplot[linecolor=red,linewidth=1.5pt]{0}{6}{
    x Pi 3 div mul RadtoDeg sin}
\end{pspicture}
```

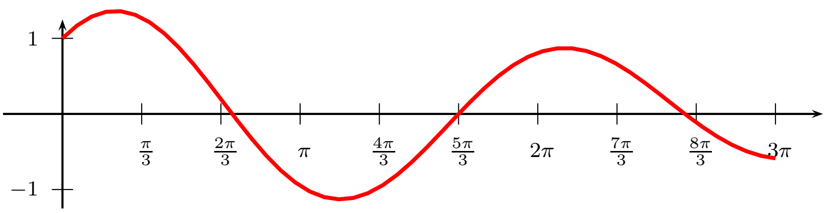



```
\psset{xunit=\pstRadUnit}%
\begin{pspicture}(-0.5,-1.25)(6.5,1.25)
  \psaxes[dx=1.5]{->}(0,0)(-0.5,-1.25)
    (6.5,1.25)
  \psplot[xunit=0.5cm,linecolor=red,linewidth
    =1.5pt]{0}{\psPiFour}{x RadtoDeg sin}
\end{pspicture}
```



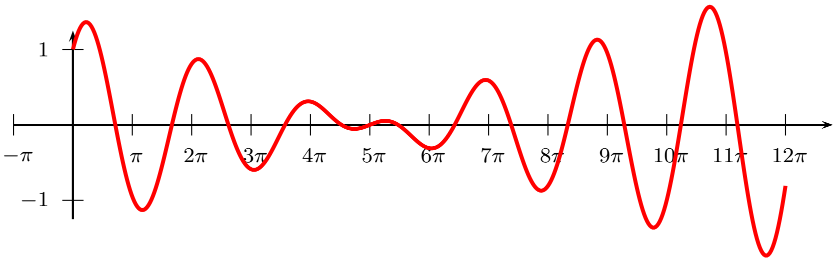

```
\psset{xunit=\pstRadUnit}%
\begin{pspicture}(-0.5,-1.25)(6.5,1.25)
  \psaxes[dx=0.75,trigLabelBase=2]{->}(0,0)
    (-0.5,-1.25)(6.5,1.25)
  \psplot[xunit=0.5cm,linecolor=red,linewidth
    =1.5pt]{0}{\psPiFour}{x RadtoDeg sin}
\end{pspicture}
```

It is also possible to set the x unit and dx value to get the labels right. But this needs some more understanding as to how it really works. A `xunit=1.570796327` sets the unit to $\pi/2$ and a `dx=0.666667` then puts at every $2/3$ of the unit a tick mark and a label. The length of the x-axis is 6.4 units which is $6.4 \cdot 1.570796327cm \approx 10cm$. The function then is plotted from 0 to $3\pi = 9.424777961$.

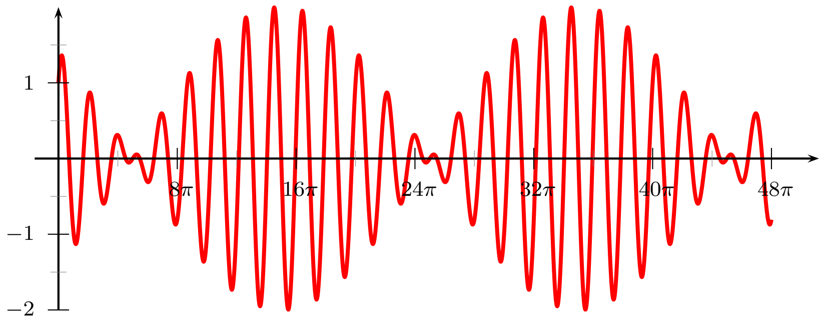



```
\begin{pspicture}(-0.5,-1.25)(10,1.25)
  \psaxes[xunit=\psPiH,trigLabelBase=3,dx=0.666667]{->}(0,0)(-0.5,-1.25)
    (6.4,1.25)
```

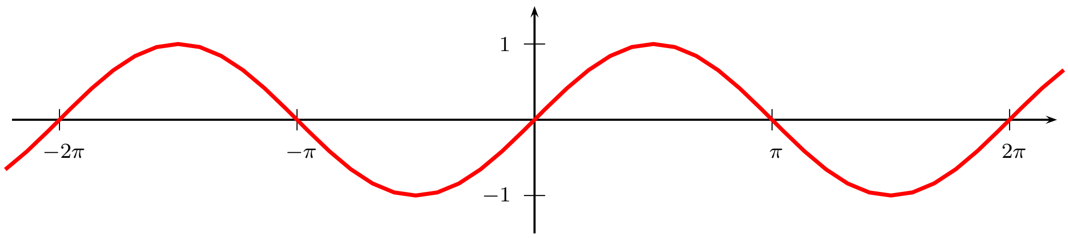
```
  \psplot[linecolor=red,linewidth=1.5pt]{0}{9.424777961}{%
  x RadtoDeg dup sin exch 1.1 mul cos add}
\end{pspicture}
```



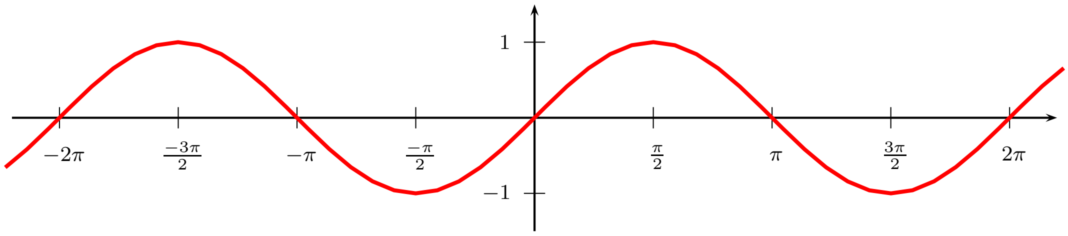

```
\psset{unit=1cm}
  \psplot[xunit=0.25,plotpoints=500,linecolor=red,linewidth=1.5pt
    ]{0}{37.70}{%
  x RadtoDeg dup sin exch 1.1 mul cos add}
\end{pspicture}
```




```
\psset{unit=1cm}
\begin{pspicture}(-0.5,-1.25)(10,1.25)
  \psplot[xunit=0.0625,linecolor=red,linewidth=1.5pt,%
     plotpoints=5000]{0}{150.80}%
  {x RadtoDeg dup sin exch 1.1 mul cos add}
  \psaxes[xunit=\psPi,dx=0.5,Dx=8]{->}(0,0)(-0.25,-1.25)(3.2,1.25)
\end{pspicture}
```




```
\begin{pspicture}(-7,-1.5)(7,1.5)
  \psaxes[trigLabels=true,xunit=\psPi]{->}(0,0)(-2.2,-1.5)(2.2,1.5)
  \psplot[linecolor=red,linewidth=1.5pt]{-7}{7}{x RadtoDeg sin}
\end{pspicture}
```



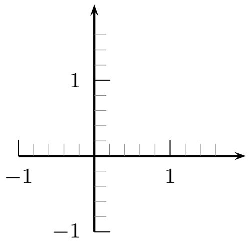

```
\begin{pspicture}(-7,-1.5)(7,1.5)
  \psaxes[trigLabels=true,
  trigLabelBase=2,dx=\psPiH,xunit=\psPi]{->}(0,0)(-2.2,-1.5)(2.2,1.5)
  \psplot[linecolor=red,linewidth=1.5pt]{-7}{7}{x RadtoDeg sin}
\end{pspicture}
```

26.10. ticks

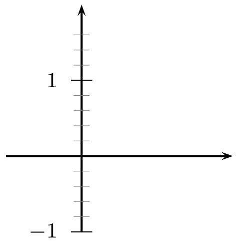
Syntax:

```
ticks=all|x|y|none
```

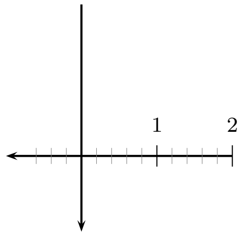
This option is also already in the `pst-plot` package and only mentioned here for some completeness.



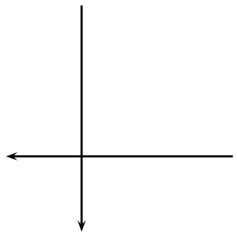

```
\psset{ticksize=6pt}
\begin{pspicture}(-1,-1)(2,2)
\psaxes[ticks=all,subticks=5]{->}(0,0)(-1,-1)(2,2)
\end{pspicture}
```




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



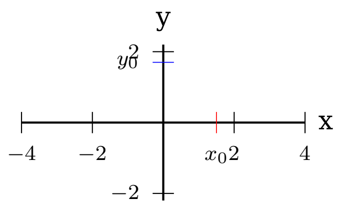

```
\begin{pspicture}(-1,-1)(2,2)
\psaxes[ticks=x,subticks=5]{->}(0,0)(2,2)(-1,-1)
\end{pspicture}
```



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

Single ticks with labels can be set with the two macros `\psxTick` and `\psyTick`:

```
\psxTick[options](x value){label}
\psyTick[options](y value){label}
```



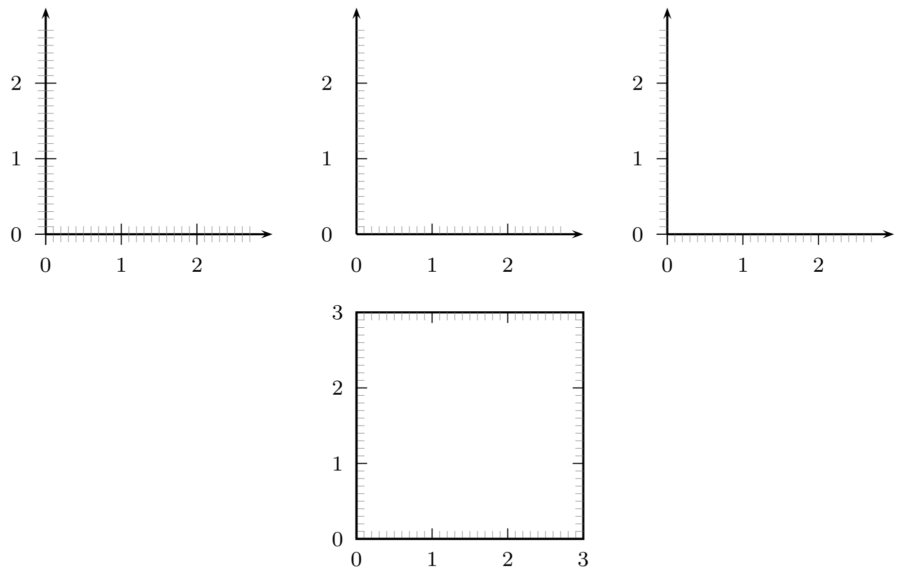

```
\begin{psgraph}[Dx=2,Dy=2](0,0)(-4,-2.2)(4,2.2){.5\textwidth}{!}
  \psxTick[linecolor=red](1.5){x_0}
  \psyTick[linecolor=blue](1.7){y_0}
\end{psgraph}
```

26.11. tickstyle

Syntax:

```
tickstyle=full|top|bottom|inner
```

The value `inner` (not available with the basic `pstricks` package) is only valid for the axes style `frame`.




```
\psset{subticks=10}
\begin{pspicture}(-1,-1)(3,3) \psaxes[tickstyle=full]{->}(3,3) \end{pspicture}
\begin{pspicture}(-1,-1)(3,3) \psaxes[tickstyle=top]{->}(3,3) \end{pspicture}
\begin{pspicture}(-1,-1)(3,3) \psaxes[tickstyle=bottom]{->}(3,3)\end{pspicture}
\begin{pspicture}(-1,-1)(3,3)
 \psaxes[axesstyle=frame, tickstyle=inner, ticksize=0 4pt]{->}(3,3)
\end{pspicture}
```

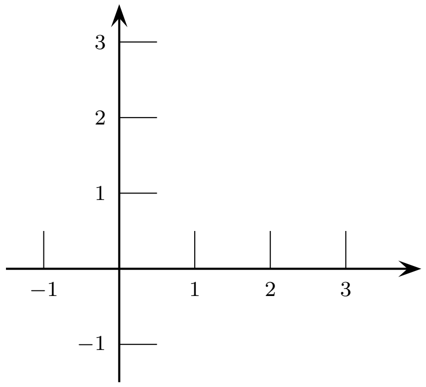
26.12. ticksize, xticksize, yticksize

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

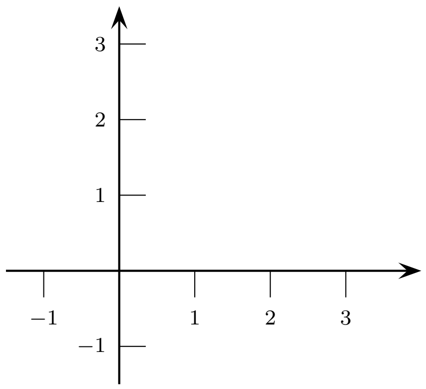
Syntax:

```
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.

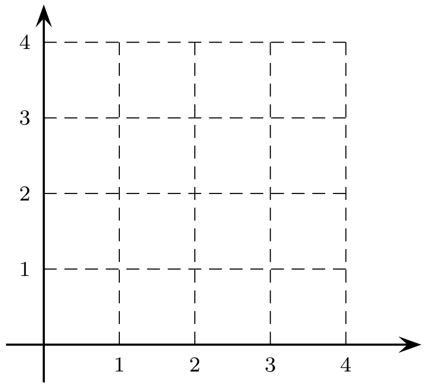



```
\psset{arrowscale=2}
\begin{pspicture}(-1.5,-1.5)(4,3.5)
  \psaxes[ticksize=0.5cm]{->}(0,0)
    (-1.5,-1.5)(4,3.5)
\end{pspicture}
```




```
\psset{arrowscale=2}
\begin{pspicture}(-1.5,-1.5)(4,3.5)
  \psaxes[xticksize=-10pt 0,yticksize=0 10
    pt]%
    {->}(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.



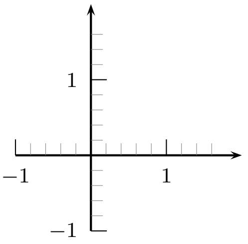

```
\psset{arrowscale=2}
\begin{pspicture}(-.5,-.5)(5,4.5)
 \psaxes[ticklinestyle=dashed,
   ticksize=0 4cm]{->}(0,0)(-.5,-.5)
     (5,4.5)
\end{pspicture}
```

26.13. subticks

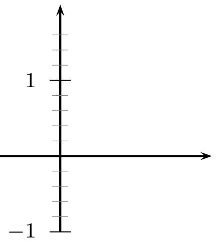
Syntax:

```
subticks=<number>
```

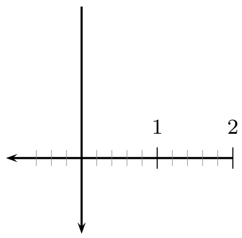
By default `subticks` cannot have labels.



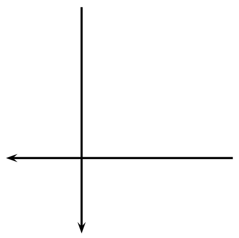

```
\psset{ticksize=6pt}
\begin{pspicture}(-1,-1)(2,2)
\psaxes[ticks=all,subticks=5]{->}(0,0)(-1,-1)(2,2)
\end{pspicture}
```




```
\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}
```



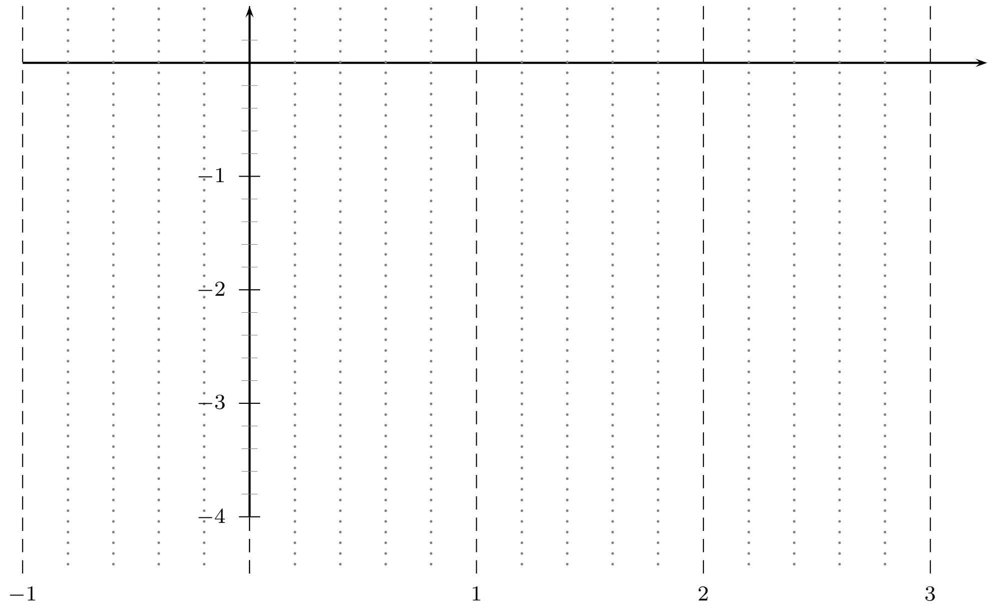
```
\begin{pspicture}(-1,-1)(2,2)
\psaxes[ticks=none,subticks=5]{->}(0,0)(2,2)(-1,-1)
\end{pspicture}
```

26.14. 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.



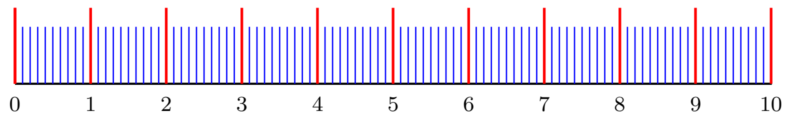

```
\psset{yunit=1.5cm,xunit=3cm}
\begin{pspicture}(-1.25,-4.75)(3.25,.75)
 \psaxes[xticksize=-4.5 0.5,ticklinestyle=dashed,subticks=5,xsubticksize=1,
   %
    ysubticksize=0.75,xsubticklinestyle=dotted,xsubtickwidth=1pt,
    subtickcolor=gray]{->}(0,0)(-1,-4)(3.25,0.5)
\end{pspicture}
```

26.15. tickcolor, subtickcolor

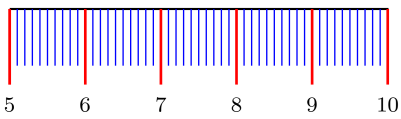
Syntax:

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

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




```
\begin{pspicture}(0,-0.75)(10,1)
\psaxes[yAxis=false,labelFontSize=\scriptstyle,ticksize=0 10mm,subticks=10,
  subticksize=0.75,
 tickcolor=red,subtickcolor=blue,tickwidth=1pt,subtickwidth=0.5pt](10.01,0)
\end{pspicture}
```



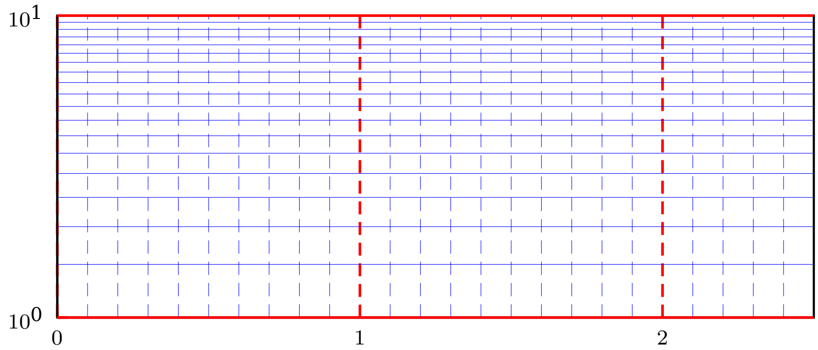

```
\begin{pspicture}(5,-0.75)(10,1)
\psaxes[yAxis=false,labelFontSize=\scriptstyle,
  ticksize=0 -10mm,subticks=10,subticksize=0.75,
  tickcolor=red,subtickcolor=blue,tickwidth=1pt,
    subtickwidth=0.5pt,Ox=5](5,0)(5,0)(10.01,0)
\end{pspicture}
```

26.16. 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 as `[sub]ticklines=0`



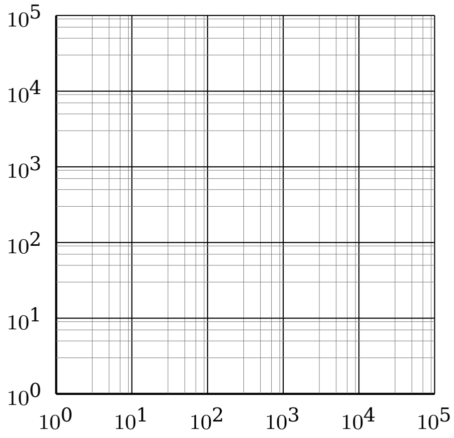

```
\psset{unit=4cm}
\pspicture(-0.15,-0.15)(2.5,1)
  \psaxes[axesstyle=frame,logLines=y,xticksize=0 1,xsubticksize=1,ylogBase
    =10,
    tickcolor=red,subtickcolor=blue,tickwidth=1pt,subticks=20,xsubticks=10,
    xticklinestyle=dashed,xsubticklinestyle=dashed](2.5,1)
\endpspicture
```

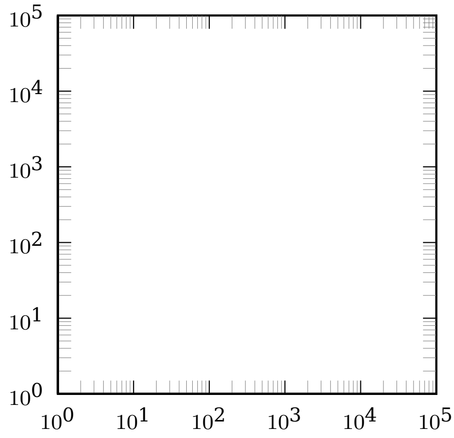
26.17. logLines

Syntax:

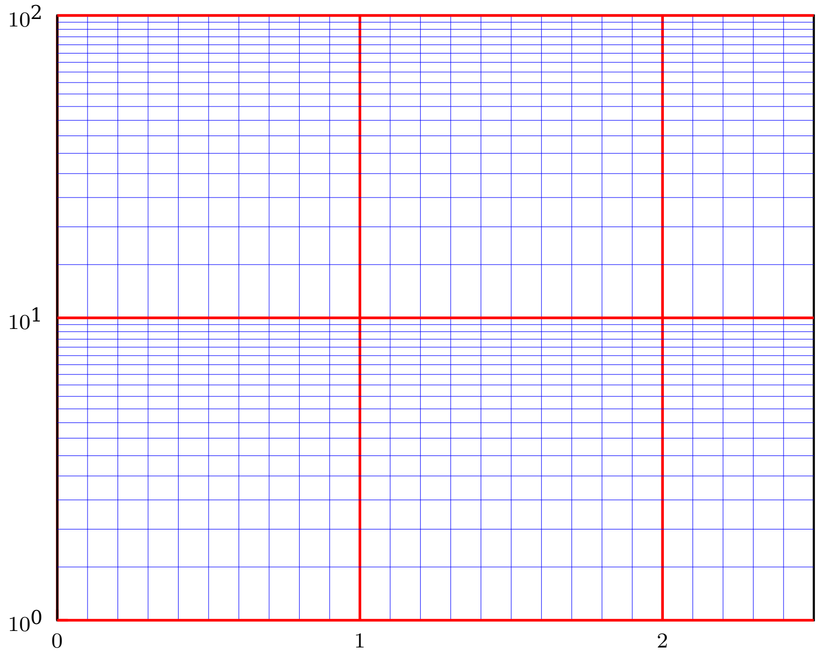
logLines=all|x|y

By default the option `logLines` sets the ticksize to the maximal length for x, y, or both. It can be changed, when *after* the option `logLines` the ticksize is set.

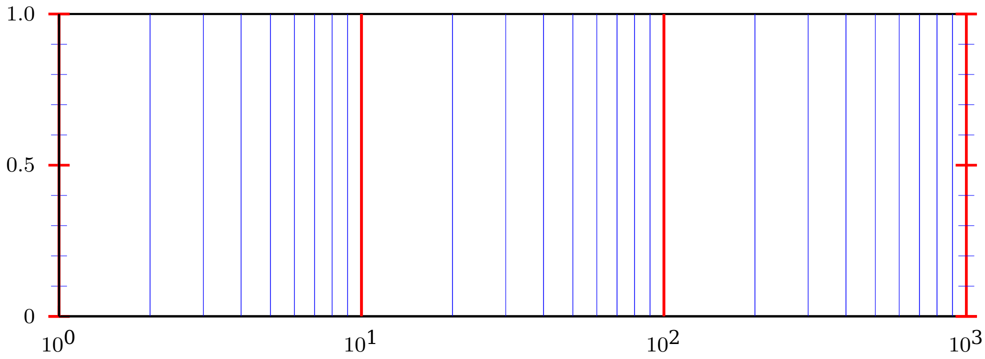





```
\pspicture(-1,-1)(5,5)
  \psaxes[subticks=5,xylogBase=10,logLines=all](5,5)
\endpspicture\hspace{1cm}
\pspicture(-1,-1)(5,5)
  \psaxes[subticks=10,axesstyle=frame,xylogBase=10,logLines=all,ticksize=0
    5pt,tickstyle=inner](5,5)
\endpspicture
```




```
\psset{unit=4cm}
\pspicture(-0.15,-0.15)(2.5,2)
 \psaxes[axesstyle=frame,logLines=y,xticksize=max,xsubticksize=1,ylogBase
   =10,
   tickcolor=red,subtickcolor=blue,tickwidth=1pt,subticks=20,xsubticks
     =10](2.5,2)
\endpspicture
```



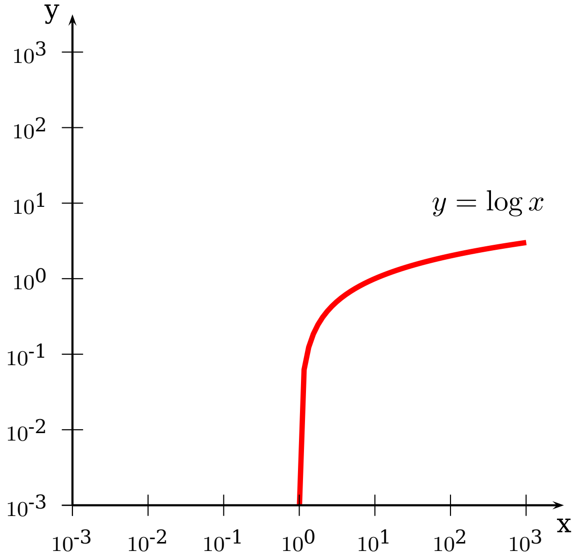

```
\psset{unit=4}
\pspicture(-0.5,-0.3)(3,1.2)
  \psaxes[axesstyle=frame,tickstyle=inner,logLines=x,xlogBase=10,Dy=0.5,
    tickcolor=red,
    subtickcolor=blue,tickwidth=1pt,ysubticks=5,xsubticks=10](3,1)
\endpspicture
```

26.18. xylogBase, xlogBase and ylogBase

There are additional options `xylogBase`, `xlogBase`, `ylogBase` to get one or both axes with logarithmic labels. For an interval of $[10^{-3}...10^{2}]$ choose a `pstricks` interval of [-3,2]. `pstricks` takes 0 as the origin of this axes, which is wrong if we want to have a logarithmic 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.

xylogBase

This mode in math is also called double logarithmic. It is a combination of the two foregoing 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,Ox
    =-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}
```

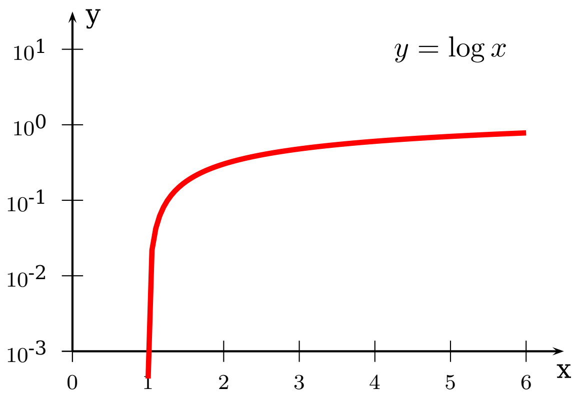
ylogBase

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

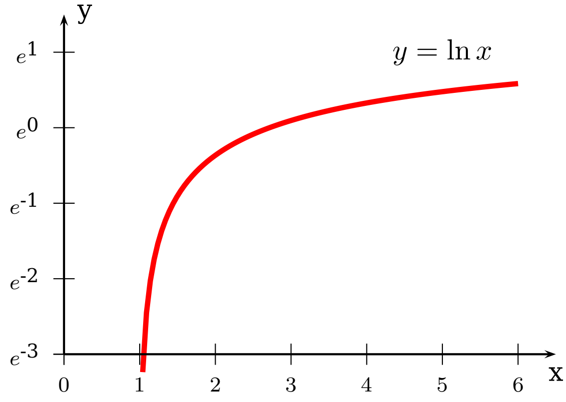
$$y = \log\{\log x\}$$

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

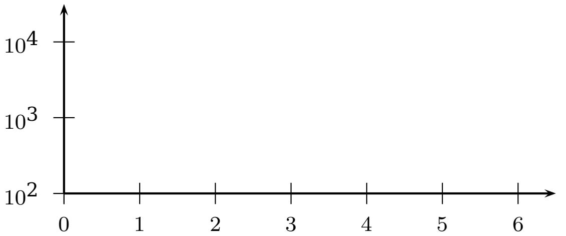
with an drawing interval of $1.001 \ldots 6$.



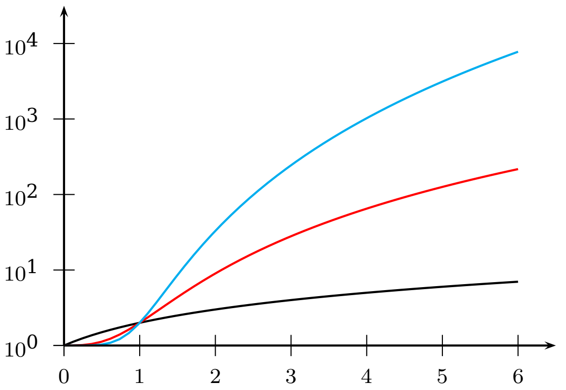

```
\begin{pspicture}(-0.5,-3.5)(6.5,1.5)
  \psaxes[ylogBase=10,Oy
    =-3]{->}(0,-3)(6.5,1.5)
  \uput[-90](6.5,-3){x}
  \uput[0](0,1.4){y}
  \rput(5,1){$y=\log x$}
  \psplot[linewidth=2pt,%
  plotpoints=100,linecolor=red
    ]{1.001}{6}{x log log} % log(log(x
    ))
\end{pspicture}
```



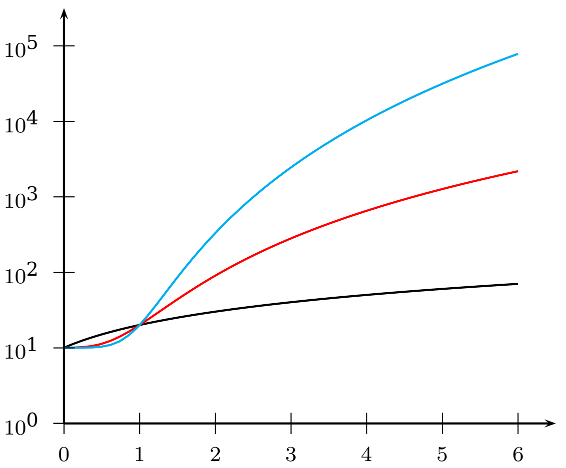

```
\begin{pspicture}(-0.5,-3.5)(6.5,1.5)
  \psplot[linewidth=2pt,plotpoints
    =100,linecolor=red]%
    {1.04}{6}[ /ln {log 0.4343 div}
      def ]{x ln ln} % log(x)
  \psaxes[ylogBase=e,Oy=-3]{->}(0,-3)
    (6.5,1.5)
  \uput[-90](6.5,-3){x}
  \uput[0](0,1.5){y}
  \rput(5,1){$y=\ln x$}
\end{pspicture}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



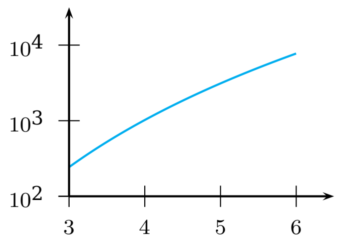

```
  \begin{pspicture}(-0.5,1.75)
    (6.5,4.5)
    \psaxes[ylogBase=10,Oy=2]{->}(0,2)
      (0,2)(6.5,4.5)
  \end{pspicture}
```




```
  \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}
```



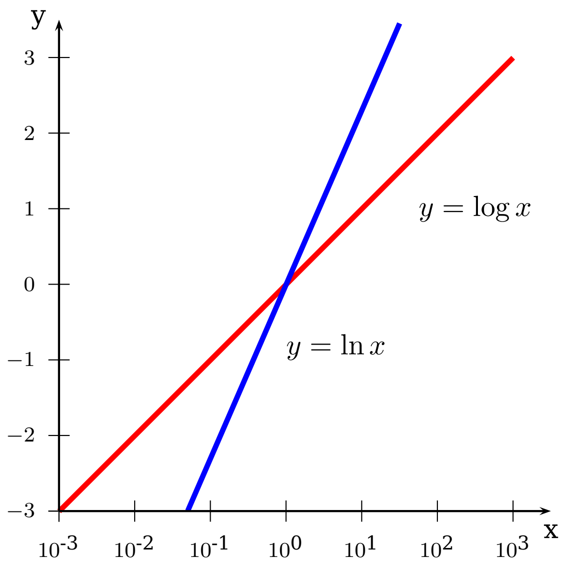

```
\begin{pspicture}(-0.5,-1.25)
  (6.5,4.5)
  \psplot{0}{6}{x x cos add log}
           % x  + cox(x)
  \psplot[linecolor=red]{0}{6}{x 3
    exp x cos add log} % x^3 + cos(x)
  \psplot[linecolor=cyan]{0}{6}{x 5
    exp x cos add log} % x^5 + cos(x)
  \psaxes[ylogBase=10]{->}(0,-1)
    (0,-1)(6.5,4.5)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



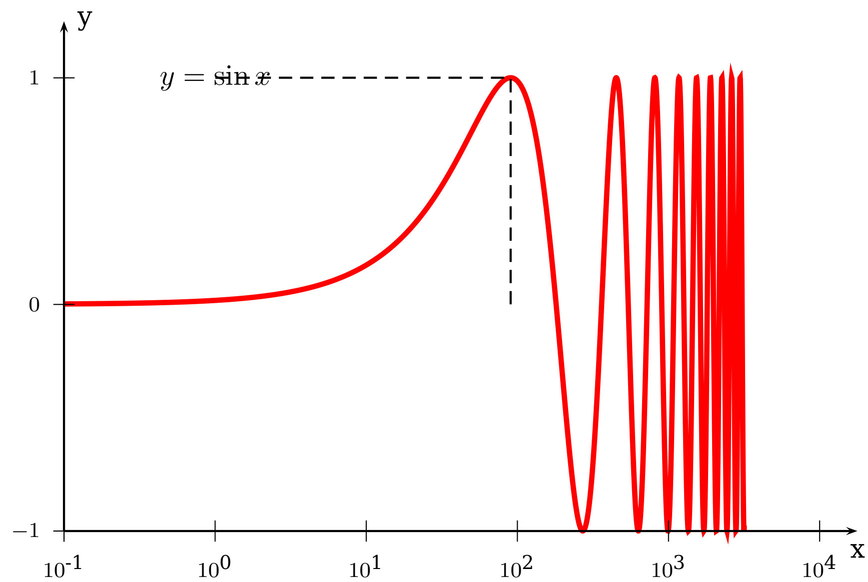

```
\begin{pspicture}(2.5,1.75)(6.5,4.5)
  \psplot[linecolor=cyan]{3}{6}{x 5 exp x cos add log
    } % x^5 + cos(x)
  \psaxes[ylogBase=10,Ox=3,Oy=2]{->}(3,2)(3,2)
    (6.5,4.5)
\end{pspicture}
```

xlogBase

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

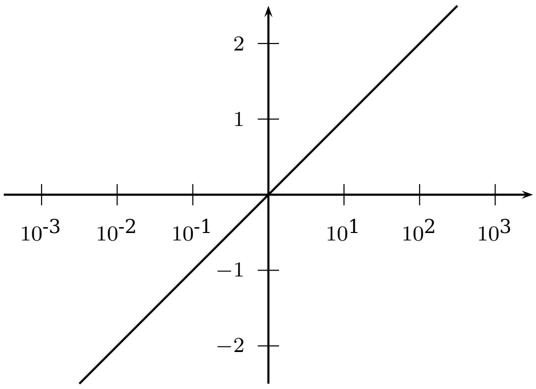



```
\begin{pspicture}(-3.5,-3.5)(3.5,3.5)
  \psplot[linewidth=2pt,linecolor=red
    ]{-3}{3}{x} % log(x)
  \psplot[linewidth=2pt,linecolor=
    blue]{-1.3}{1.5}{x 0.4343 div} %
    ln(x)
  \psaxes[xlogBase=10,Oy=-3,Ox
    =-3]{->}(-3,-3)(3.5,3.5)
  \uput[-90](3.5,-3){x}
  \uput[180](-3,3.5){y}
  \rput(2.5,1){$y=\log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

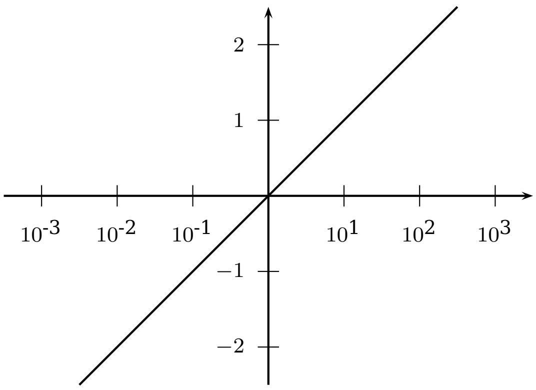



```
\psset{yunit=3cm,xunit=2cm}
\begin{pspicture}(-1.25,-1.25)(4.25,1.5)
```

```
  \uput[-90](4.25,-1){x}
  \uput[0](-1,1.25){y}
  \rput(0,1){$y=\sin x$}
  \psplot[linewidth=2pt,plotpoints=5000,linecolor=red]{-1}{3.5}{10 x exp sin }
  \psaxes[xlogBase=10,Ox=-1,Oy=-1]{->}(-1,-1)(4.25,1.25)
   \psline[linestyle=dashed](-1,0)(4,0)
   \psline[linestyle=dashed](!-1 1)(!90 log 1)(!90 log -1)
   \psline[linestyle=dashed](!90 log 1)(!180 log 1)(!180 log -1)
\end{pspicture}
```



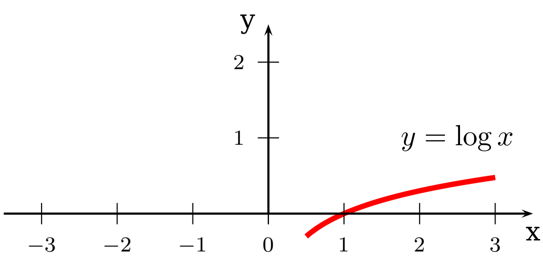

```
\begin{pspicture}(-3.5,-2.5)(3.5,2.5)
  \psaxes[xlogBase=10]{->}(0,0)(-3.5,-2.5)(3.5,2.5)
  \psplot{-2.5}{2.5}{10 x exp log}
\end{pspicture}
```




```
\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}
```

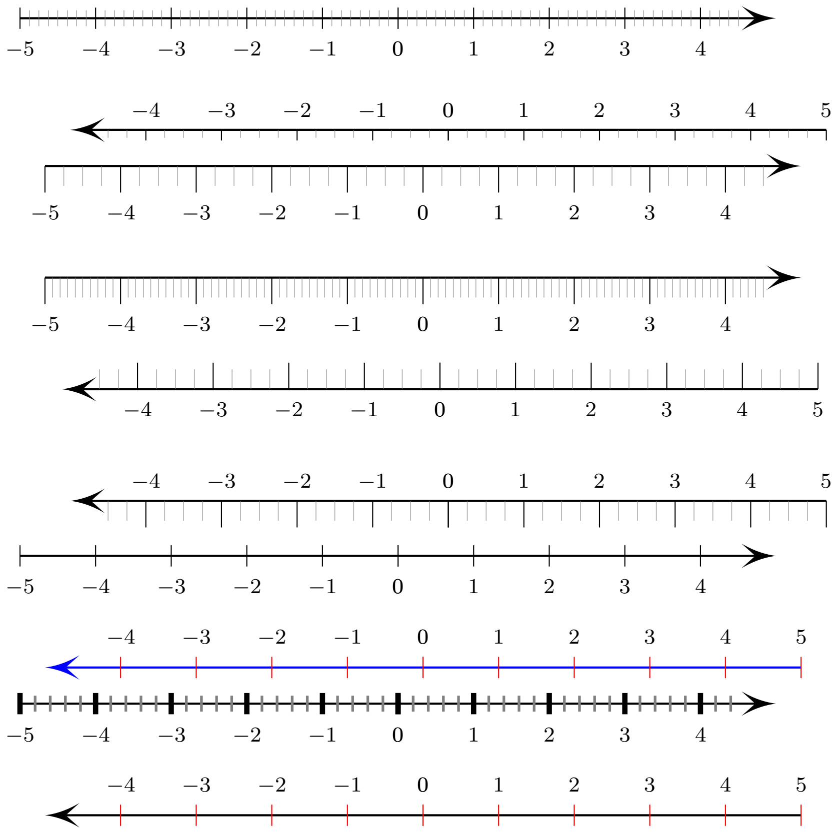
No logstyle (xylogBase={})

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

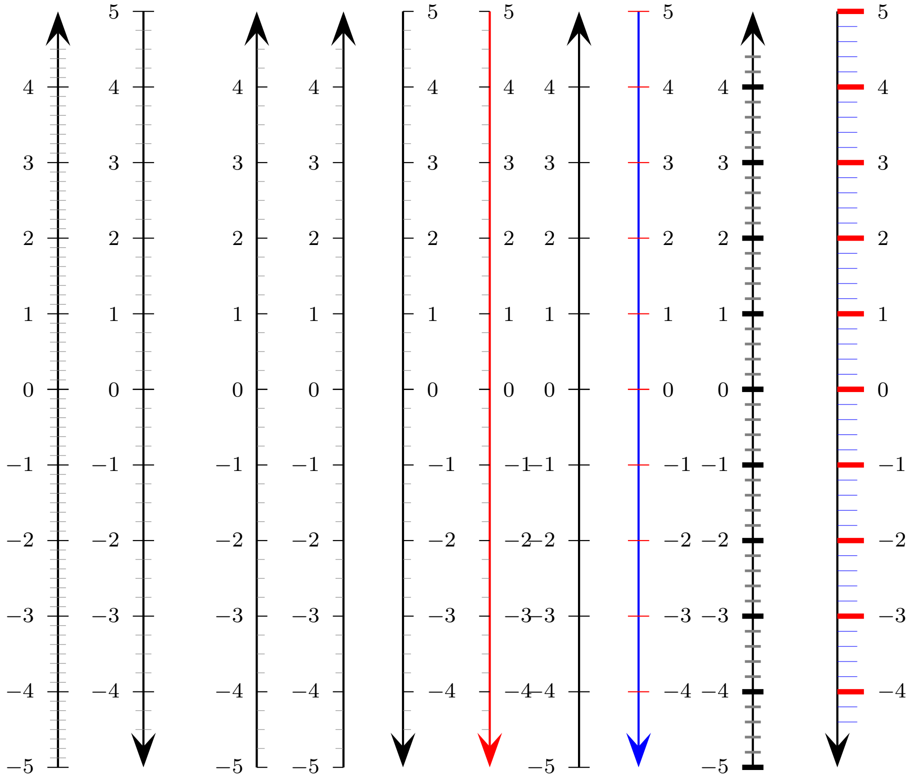



```
\begin{pspicture}(-3.5,-0.5)(3.5,2.5)
  \psplot[linewidth=2pt,linecolor=red,xylogBase={}]{0.5}{3}{x log} % log(x)
  \psaxes{->}(0,0)(-3.5,0)(3.5,2.5)
  \uput[-90](3.5,0){x}
  \uput[180](0,2.5){y}
  \rput(2.5,1){$y=\log x$}
\end{pspicture}
```

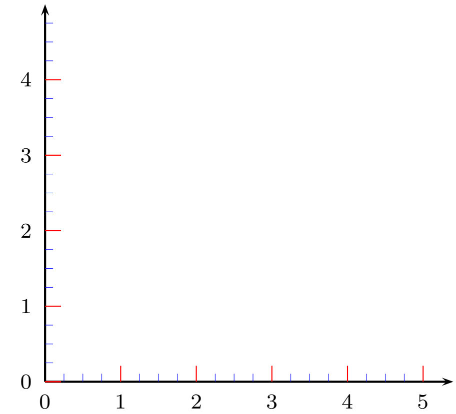
26.19. subticks, tickwidth and subtickwidth



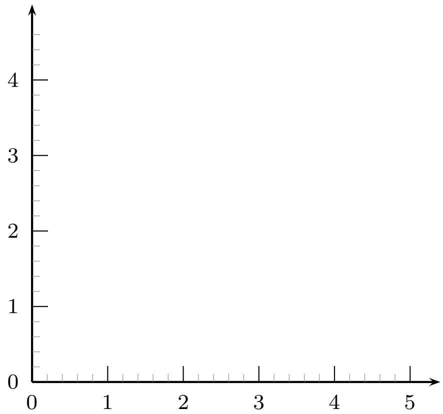
```
\psset{arrowscale=3,arrows=-D>,yAxis=false}
 \psaxes[subticks=8](0,0)(-5,-1)(5,1)\\[1cm]
 \psaxes[subticks=4,ticksize=-4pt 0,xlabelPos=top](0,0)(5,1)(-5,-1)\\
 \psaxes[subticks=4,ticksize=-10pt 0](0,0)(-5,-5)(5,5)\\[1cm]
 \psaxes[subticks=10,ticksize=0 -10pt](0,0)(-5,-5)(5,5)\\[1cm]
 \psaxes[subticks=4,ticksize=0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



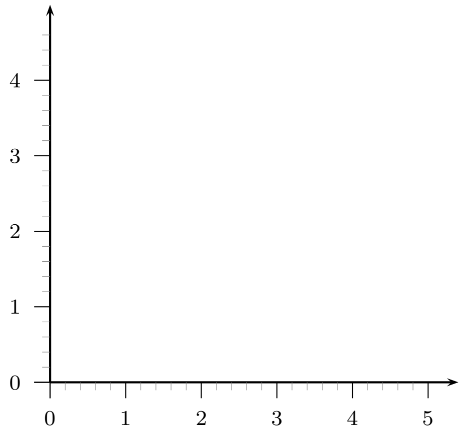
```
\psset{arrowscale=3,xAxis=false}
\psaxes[subticks=8]{->}(0,0)(-5,-5)(5,5)\hspace{2em}
\psaxes[subticks=4,ylabelPos=right,ylabelPos=left]{->}(0,0)(5,5)(-5,-5)\hspace{4em}
\psaxes[subticks=4,ticksize=0 4pt]{->}(0,0)(-5,-5)(5,5)\hspace{3em}
\psaxes[subticks=4,ticksize=-4pt 0]{->}(0,0)(-5,-5)(5,5)\hspace{1em}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



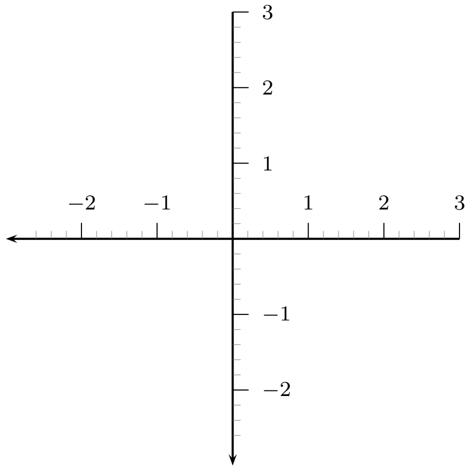

```
\pspicture(5,5.5)
\psaxes[subticks=4,ticksize=6pt,subticksize
  =0.5,%
  tickcolor=red,subtickcolor=blue]{->}(5.4,5)
\endpspicture
```



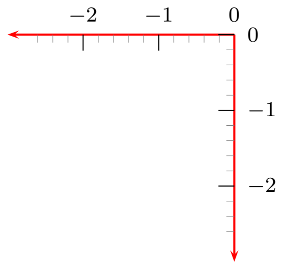

```
\pspicture(5,5.5)
  \psaxes[subticks=5,ticksize=0 6pt,
    subticksize=0.5]{->}(5.4,5)
\endpspicture
```



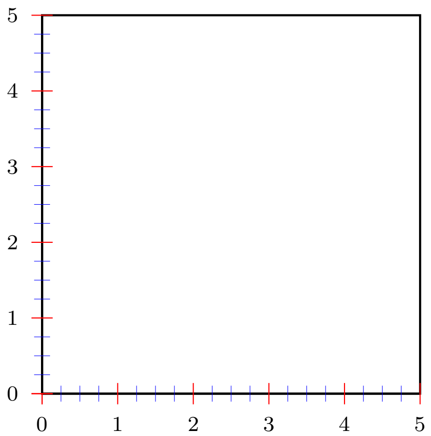

```
\pspicture(5,5.5)
  \psaxes[subticks=5,ticksize=-6pt 0,
    subticksize=0.5]{->}(5.4,5)
\endpspicture
```



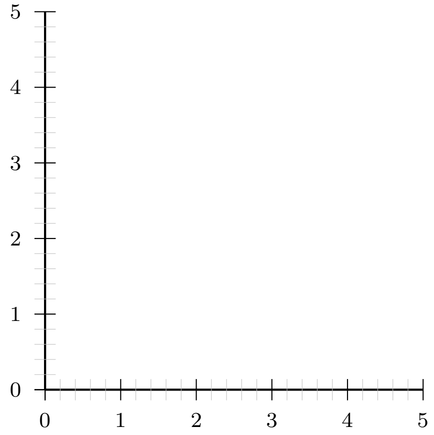

```
\pspicture(-3,-3)(3,3.5)
  \psaxes[subticks=5,ticksize=0 6pt,
    subticksize=0.5]{->}(0,0)(3,3)
    (-3,-3)
\endpspicture
```



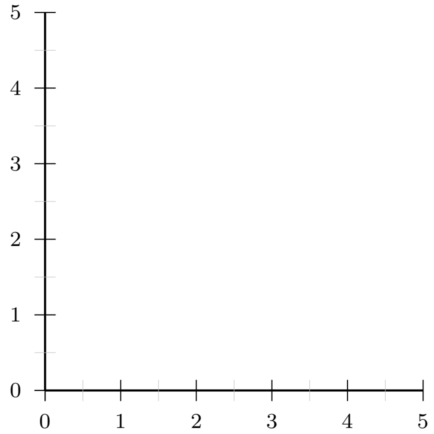

```
\pspicture(0,0.5)(-3,-3)
  \psaxes[subticks=5,ticksize=-6pt 0,
    subticksize=0.5,linecolor=red
    ]{->}(-3,-3)
\endpspicture
```



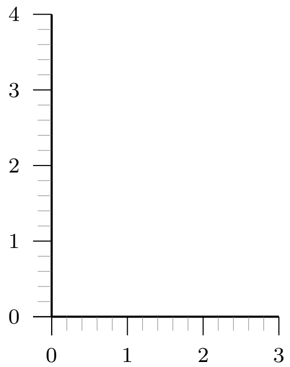

```
\psset{axesstyle=frame}
\pspicture(5,5.5)
  \psaxes[subticks=4,tickcolor=red,
    subtickcolor=blue](5,5)
\endpspicture
```



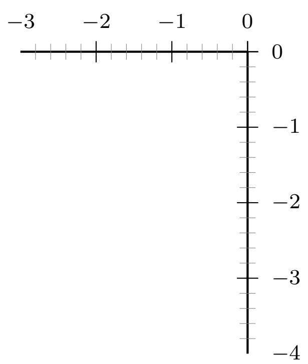

```
\pspicture(5,5.5)
  \psaxes[subticks=5,subticksize=1,
    subtickcolor=lightgray](5,5)
\endpspicture
```



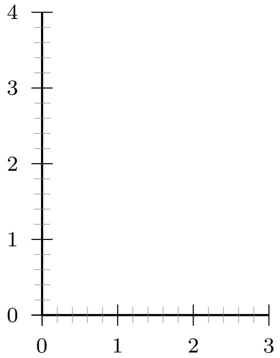

```
\pspicture(5,5.5)
  \psaxes[subticks=2,subticksize=1,
    subtickcolor=lightgray](5,5)
\endpspicture
```



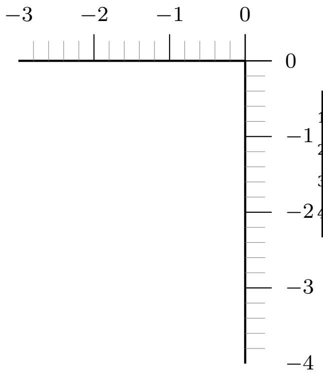

```
\pspicture(3,4.5)
  \psaxes[subticks=5,ticksize=-7pt 0](3,4)
\endpspicture
```




```
\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](-3,-4)
\endpspicture
```

26.20. algebraic

By default the function in `\psplot` has to be described in Reversed Polish Notation. The option `algebraic` allows you 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)`

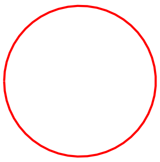
Setting the option `algebraic` to `true`, allow the user to describe all expression to be written in the classical algebraic notation (infix notation). The four arithmetic operations are obviously defined `+-*/`, and also the exponential operator `^`. The natural priorities are used : $3+4\times5^5 =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

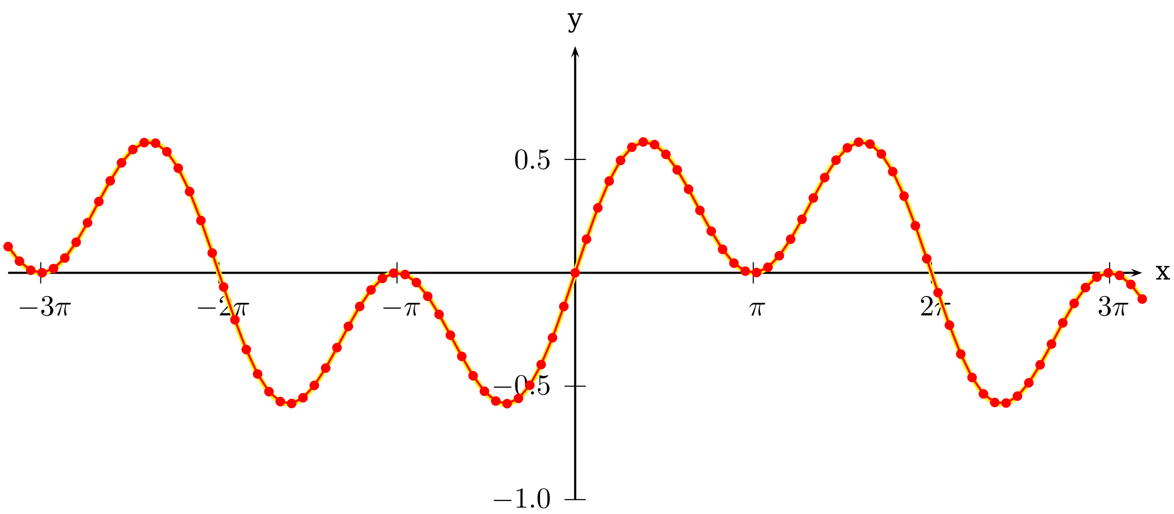
These options can be used with **all** plot macros.

Using the option `algebraic` implies that all angles have to be in radians!

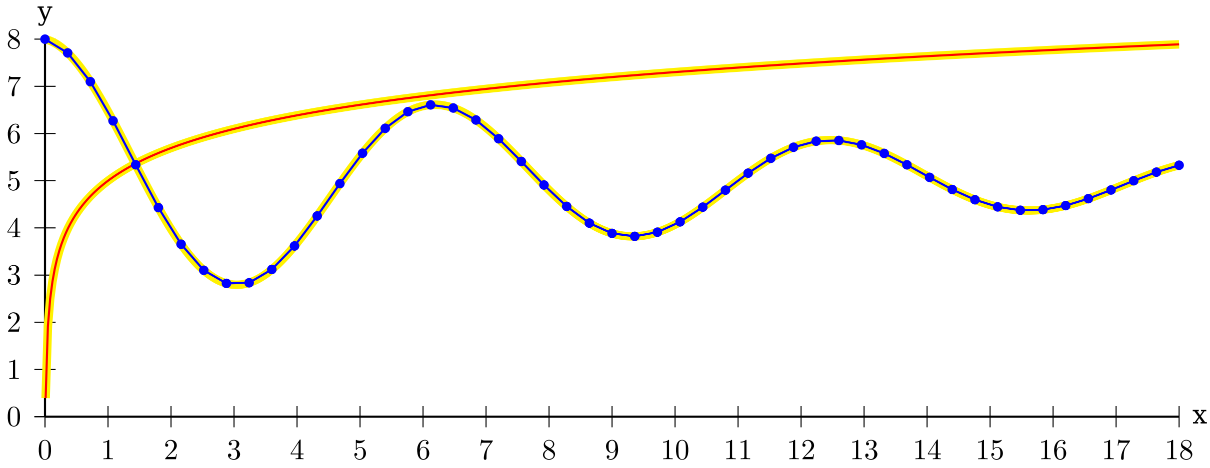
For the `\parametricplot` the two parts must be divided by the `|` character:



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



```
\psset{lly=-0.5cm}
\psgraph[trigLabels,dx=\psPi,dy=0.5,Dy=0.5]{->}(0,0)(-10,-1)(10,1){\linewidth}{6cm}
  \psset{algebraic,plotpoints=1000}
  \psplot[linecolor=yellow,linewidth=2pt]{-10}{10}{0.75*sin(x)*cos(x/2)}
  \psplot[linecolor=red,showpoints=true,plotpoints=101]{-10}{10}{0.75*sin(x)*cos(x/2)}
\endpsgraph
```

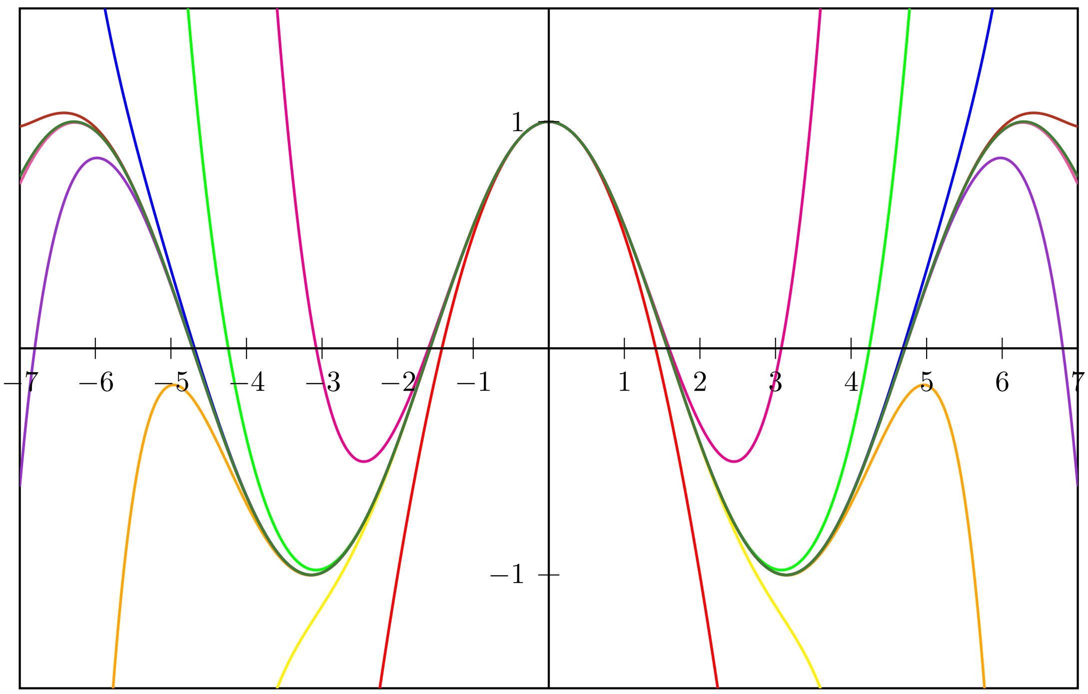



```
\psset{lly=-0.5cm}
\psgraph(0,-5)(18,3){15cm}{5cm}
  \psset{algebraic,plotpoints=501}
  \psplot[linecolor=yellow, linewidth=4\pslinewidth]{0.01}{18}{ln(x)}
  \psplot[linecolor=red]{0.01}{18}{ln(x)}
  \psplot[linecolor=yellow,linewidth=4\pslinewidth]{0}{18}{3*cos(x)*2.71^(-x/10)}
  \psplot[linecolor=blue,showpoints=true,plotpoints=51]{0}{18}{3*cos(x)*2.71^(-x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

Using the Sum function

`\Sum(`*<index name>*`,`*<start>*`,`*<step>*`,`*<end>*`,`*<function>*`)`

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

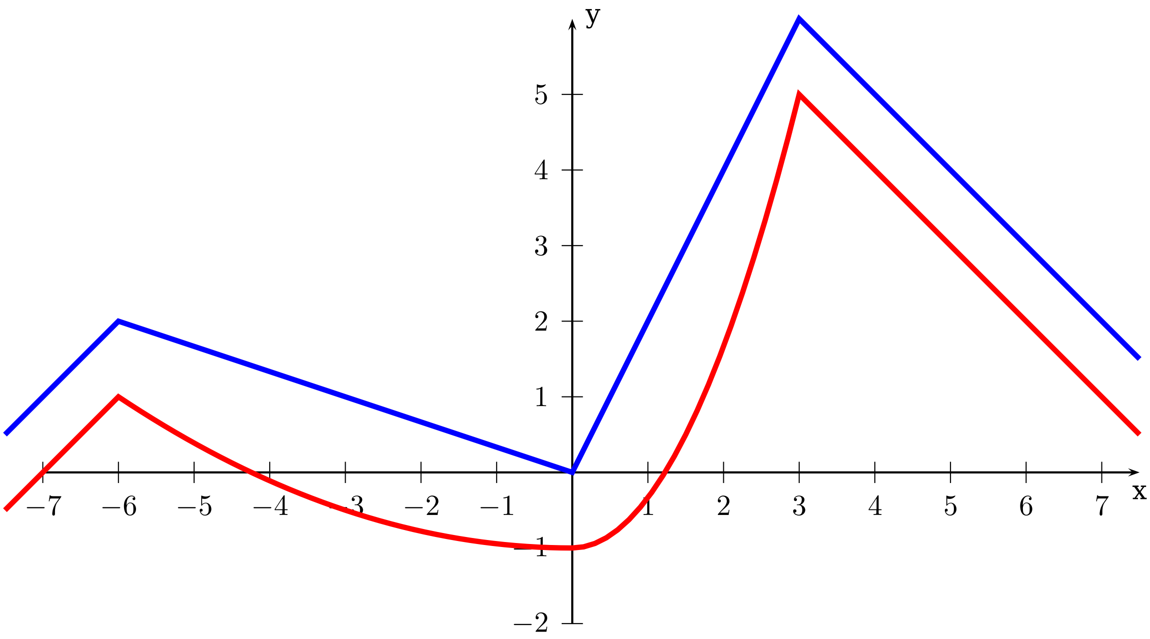



```
\psset{algebraic, plotpoints=501, yunit=3}
\def\getColor#1{\ifcase#1 black\or red\or magenta\or yellow\or green\or Orange\or blue\or
 DarkOrchid\or BrickRed\or Rhodamine\or OliveGreen\fi}
\begin{pspicture}(-7,-1.5)(7,1.5)
  \psclip{\psframe(-7,-1.5)(7,1.5)}
    \psplot{-7}{7}{cos(x)}
    \multido{\n=1+1}{10}{%
      \psplot[linewidth=1pt,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}
```

Using the IfTE function

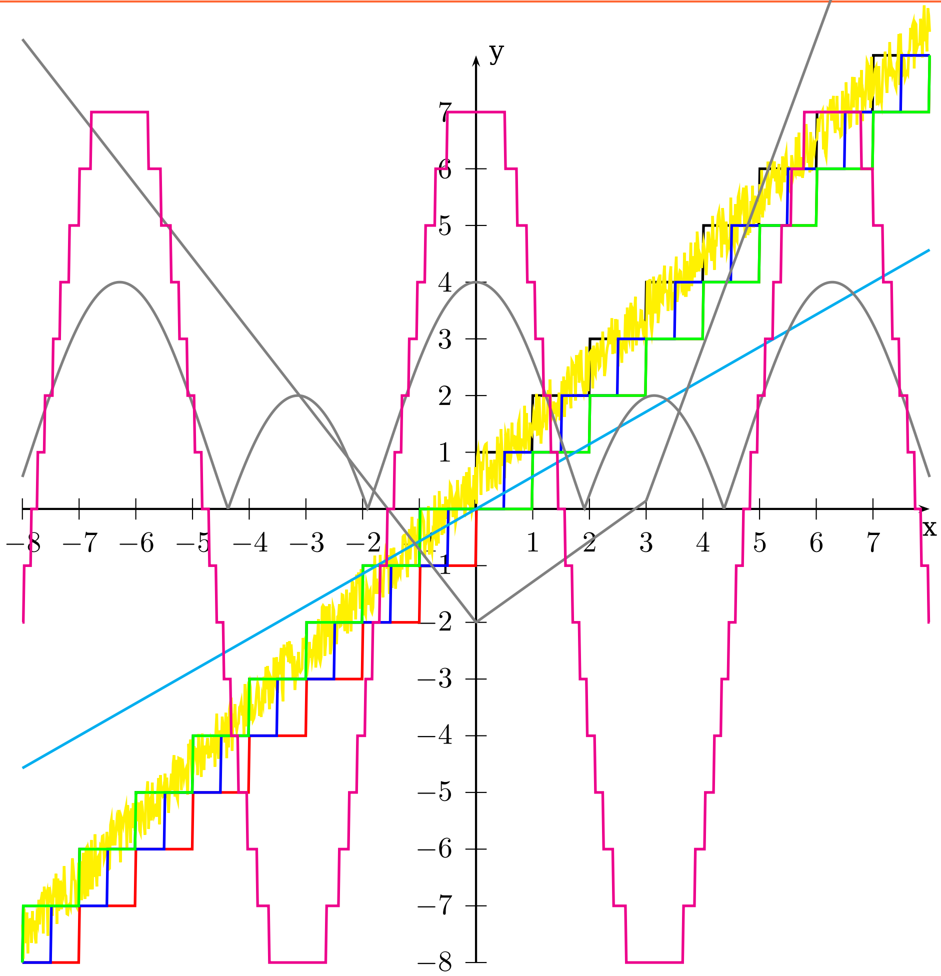
`IfTE(`*<condition>*`,`*<true part>*`,`*<false part>*`)`

Nesting of several `IfTE` is possible and seen in the following examples. A classic example is a piece-wise linear function.




```
\begin{pspicture}(-7.5,-2.5)(7.5,6)
  \psaxes{->}(0,0)(-7,-2)(7.5,6)[x,-90][y,0]
  \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}
```

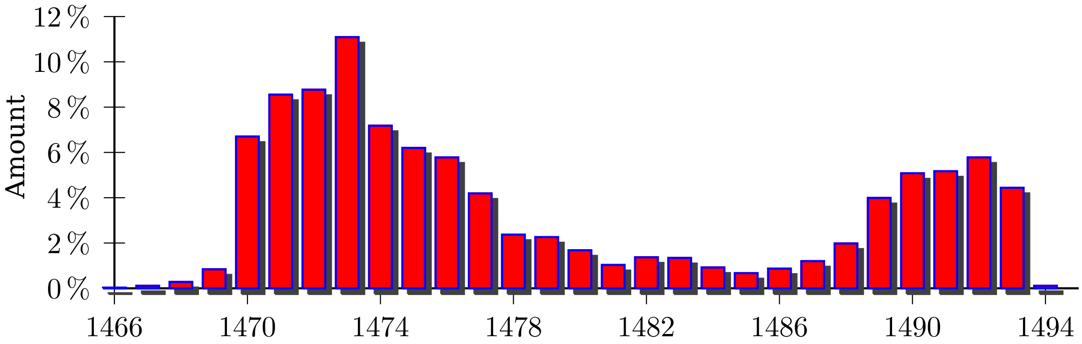
When you program a piece-wise defined function you must take care that a plotting point must be put at each point where the description changes. Use `showpoints=true` to see what's going on when there is a problem. You are on the safe side when you choose a big number for `plotpoints`.



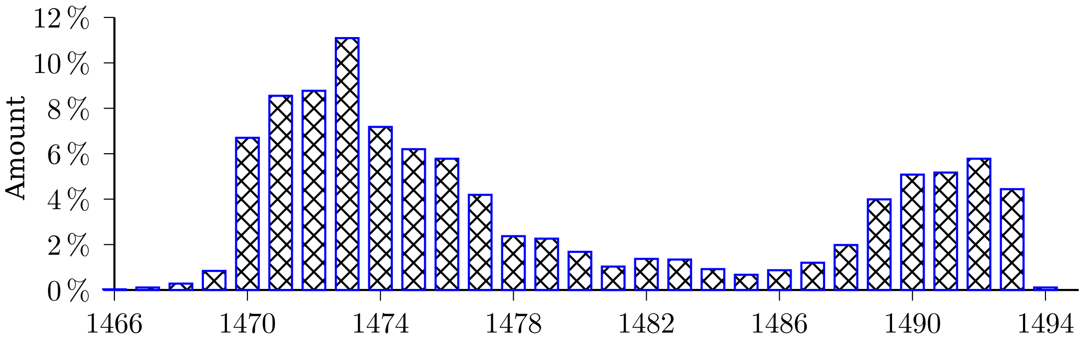

```
\psset{unit=0.75}
\begin{pspicture}(-8,-8)(8,8)
  \psaxes{->}(0,0)(-8,-8)(8,8)[x,-90][y,0]
  \psset{plotpoints=1000,linewidth=1pt}
  \psplot[algebraic, linecolor=yellow]{-8}{8}{rand/(2^31-1)+x}
  \psplot[algebraic]{-8}{8}{ceiling(x)}
  \psplot[algebraic, linecolor=red]{-8}{8}{floor(x)}
  \psplot[algebraic, linecolor=blue]{-8}{8}{round(x)}
  \psplot[algebraic, linecolor=green]{-8}{8}{truncate(x)}
  \psplot[algebraic, linecolor=cyan]{-8}{8}{div(mul(4,x),7)}
  \psplot[algebraic, linecolor=gray]{-8}{8}{abs(x)+abs(x-3)-abs(5-5*x/7)}
  \psplot[algebraic, linecolor=gray]{-8}{8}{abs(3*cos(x)+1)}
  \psplot[algebraic, linecolor=magenta]{-8}{8}{floor(8*cos(x))}
\end{pspicture}
```

26.21. Plot style bar and option barwidth

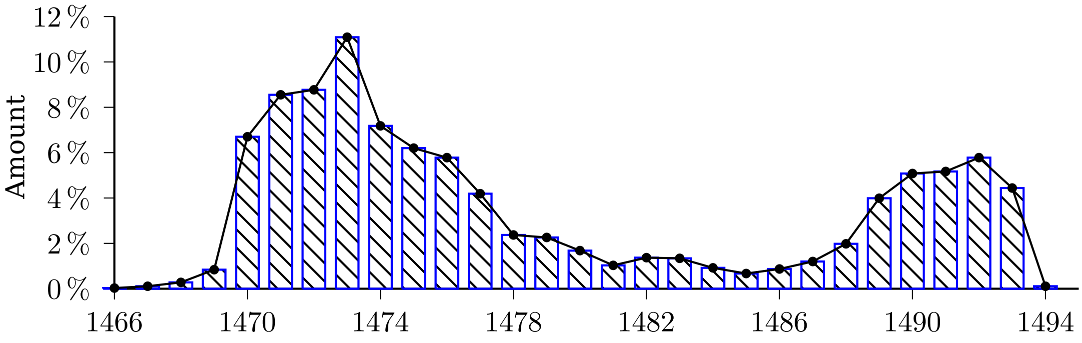
This option allows you to draw bars for the data records. The width of the bars is controlled by the option `barwidth`, which is set by default to value of `0.25cm`, which is the total width.




```
\psset{xunit=.44cm,yunit=.3cm}
\begin{pspicture}(-2,-3)(29,13)
  \psaxes[axesstyle=axes,Ox=1466,Oy=0,Dx=4,Dy=2,xticksize=-6pt 0,
     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}
```



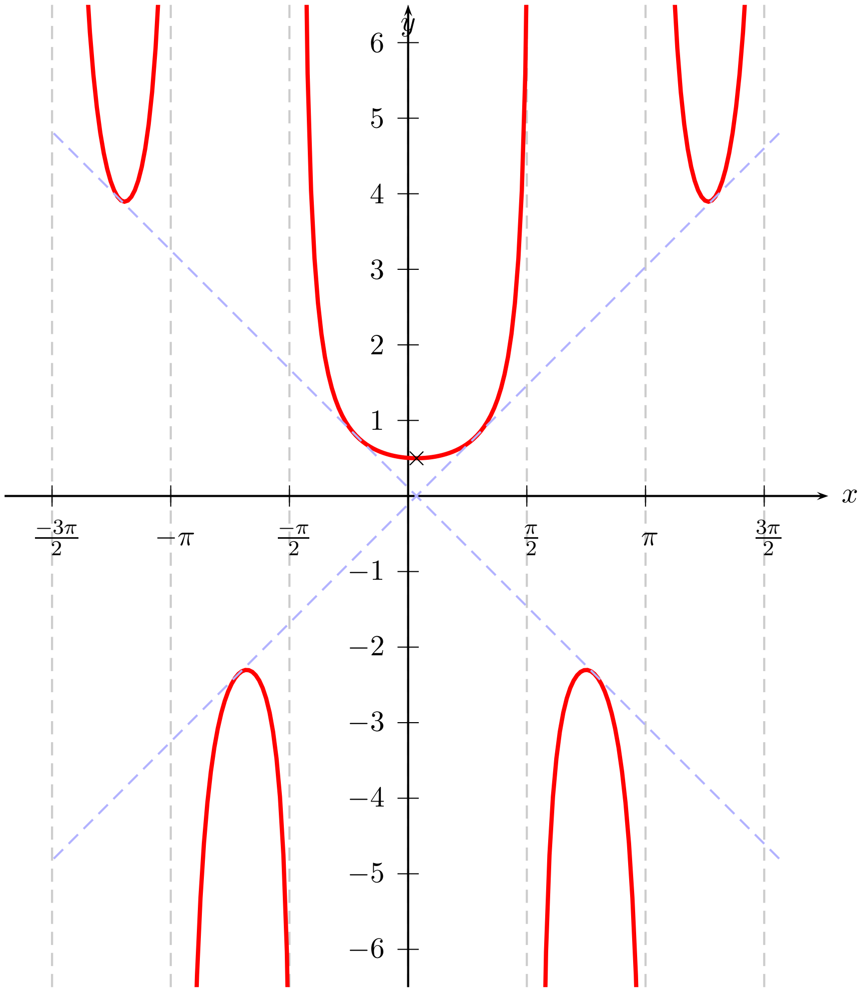

```
\psset{xunit=.44cm,yunit=.3cm}
\begin{pspicture}(-2,-3)(29,13)
  \psaxes[axesstyle=axes,Ox=1466,Oy=0,Dx=4,Dy=2,ticksize=-4pt 0,
     ylabelFactor={\,\%}]{-}(29,12)
  \listplot[linecolor=blue,plotstyle=bar,barwidth=0.3cm,
     fillcolor=red,fillstyle=crosshatch]{\barData}
  \rput{90}(-3,6.25){Amount}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



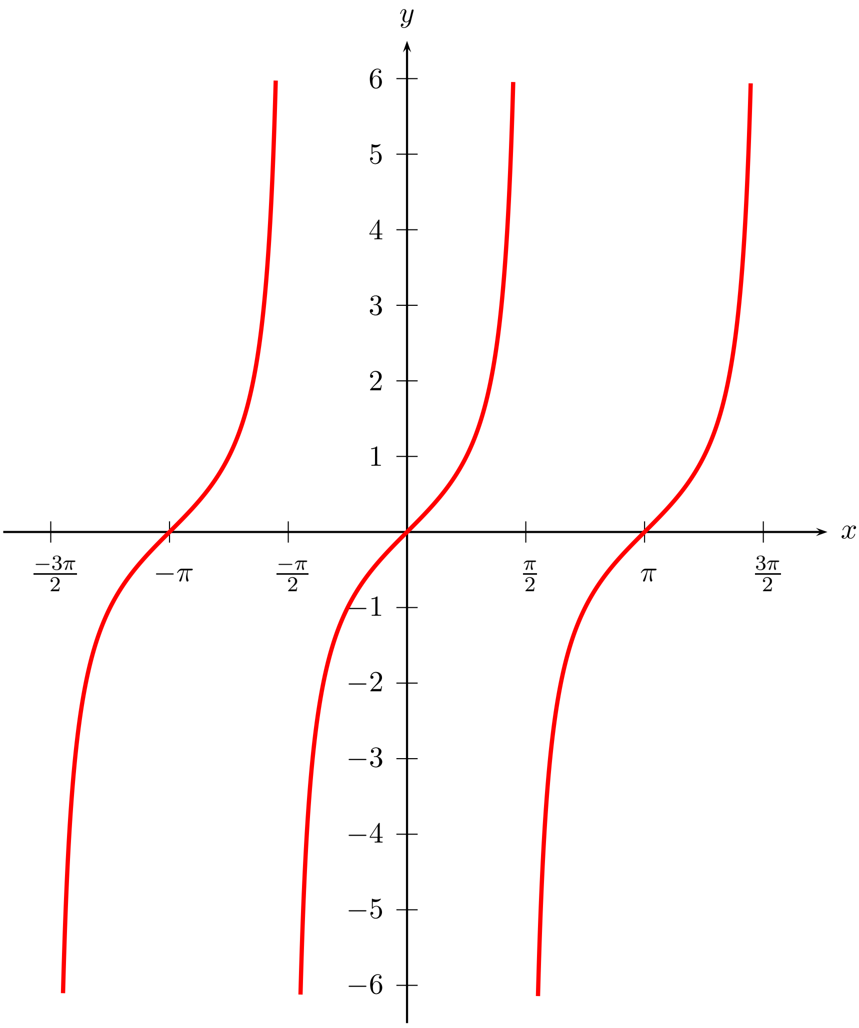

```
\psset{xunit=.44cm,yunit=.3cm}
\begin{pspicture}(-2,-3)(29,13)
  \psaxes[axesstyle=axes,Ox=1466,Oy=0,Dx=4,Dy=2,ticksize=-4pt 0,
     ylabelFactor={\,\%}]{-}(29,12)
  \listplot[linecolor=blue,plotstyle=bar,barwidth=0.3cm,
     fillcolor=red,fillstyle=vlines]{\barData}
  \listplot[showpoints=true]{\barData}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

26.22. New options yMaxValue

With the new optional argument yMaxValue one can control the behaviour of discontinued functions, like the tangent function. If yMaxValue is set to a negative value, then the internal if clause is disabled, the function is plotted in the usual way as known from pst-plot.




```
\begin{pspicture}(-6.5,-7)(6.5,7.5)
\multido{\rA=-4.71239+\psPiH}{7}{%
  \psline[linecolor=black!20,linestyle=dashed](\rA,-6.5)(\rA,6.5)}
\psaxes[trigLabelBase=2,dx=\psPiH,
  xunit=\psPi,trigLabels]{->}(0,0)(-1.7,-6.5)(1.77,6.5)[$x$,0][$y$,-90]
\psset{algebraic,plotpoints=200,plotstyle=line}
\psclip{\psframe[linestyle=none](-4.55,-6.5)(5.55,6.5)}
  \psplot[yMaxValue=10,linewidth=1.6pt,linecolor=red]{-4.55}{4.55}{(x)/(sin
    (2*x))}
\endpsclip
\psplot[linestyle=dashed,linecolor=blue!30]{-4.8}{4.8}{x}
\psplot[linestyle=dashed,linecolor=blue!30]{-4.8}{4.8}{-x}
\rput(0,0.5){$\times$}
\end{pspicture}
```




```
\begin{pspicture}(-6.5,-7)(6.5,7.5)
\psaxes[trigLabelBase=2,dx=\psPiH,
  xunit=\psPi,trigLabels]{->}(0,0)(-1.7,-6.5)(1.77,6.5)[$x$,0][$y$,90]
\psset{algebraic}
\psplot[yMaxValue=6,linewidth=1.6pt,plotpoints=2000,
  linecolor=red]{-4.55}{4.55}{tan(x)}
\end{pspicture}
```

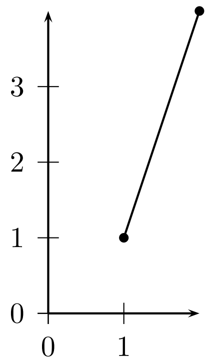
26.23. New options for \readdata

By default the macro \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 you to ignore preceding lines, e.g. ignoreLines=2, or to read only a selected part of the data records, e.g. nStep=10, only every 10^{th} record is saved.

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

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

26.24. 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 $(0, 1, 2, \ldots)$ 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
(plotstyle)	line

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 [5]):

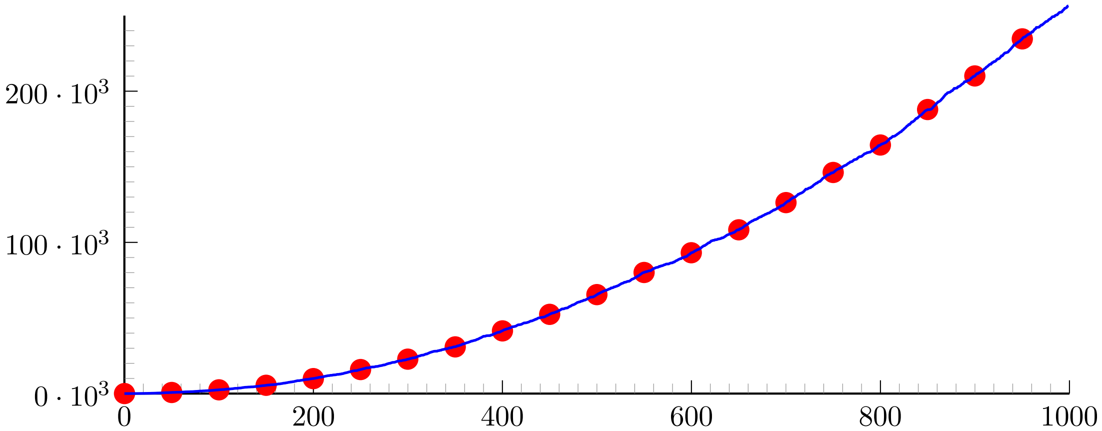
```
\readdata[nStep=10]{\data}{/home/voss/data/data1.data}
```

The use nStep and xStep options only make 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. Note that nStep can be used for \readdata and for \listplot. If used in both macros then the effect is multiplied, e.g. \readdata with nStep=5 and \listplot with nStep=10 means, that only every 50^{th} data record is read and plotted.

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

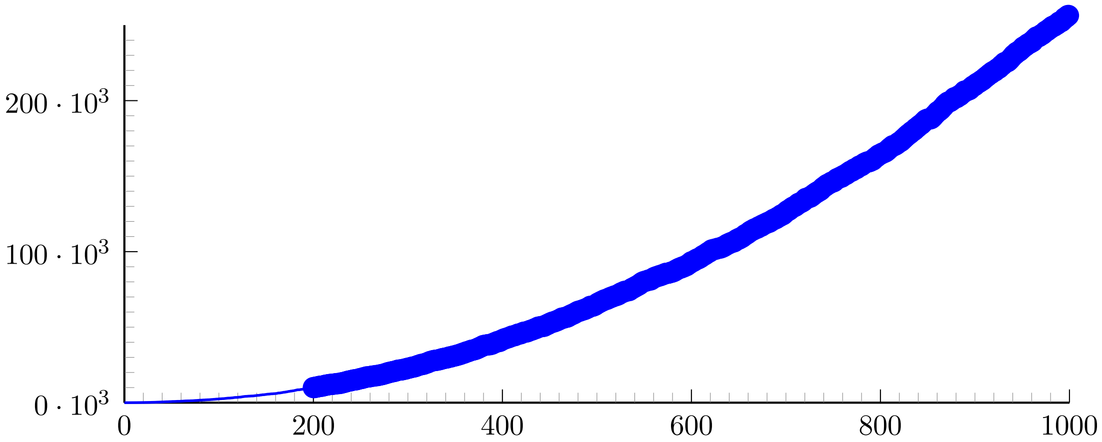
Example for nStep/xStep

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



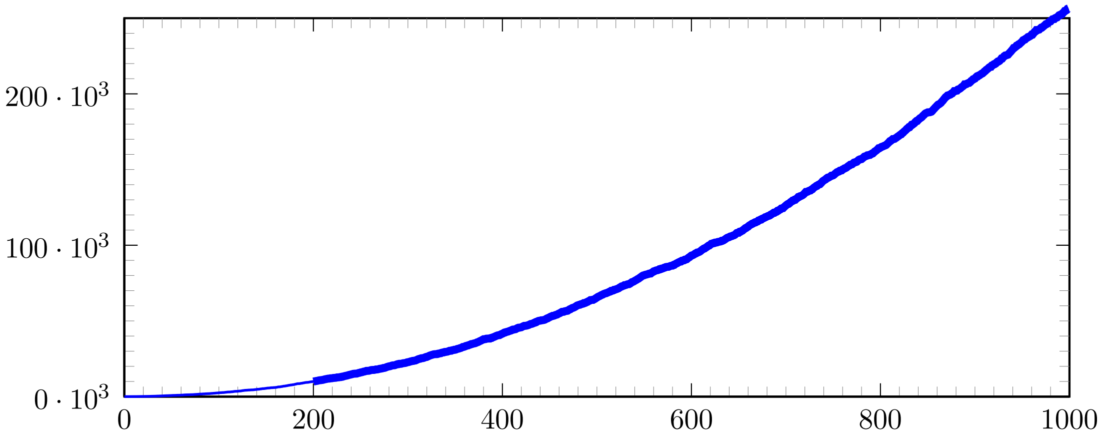

```
\readdata{\data}{data.data}
\psset{xunit=12.5cm,yunit=0.2mm}
\begin{pspicture}(-0.080,-30)(1,270)
\pstScalePoints(1,1){1000 div}{1000 div}
\psaxes[Dx=200,dx=2.5cm,Dy=100,ticksize=0 5pt,tickstyle=inner,
   subticks=10,ylabelFactor=\cdot10^3,dy=2cm](0,0)(1,250)
\listplot[nStep=50,linewidth=3pt,linecolor=red,plotstyle=dots]{\data}
\listplot[linewidth=1pt,linecolor=blue]{\data}
\end{pspicture}
```

Example for nStart/xStart



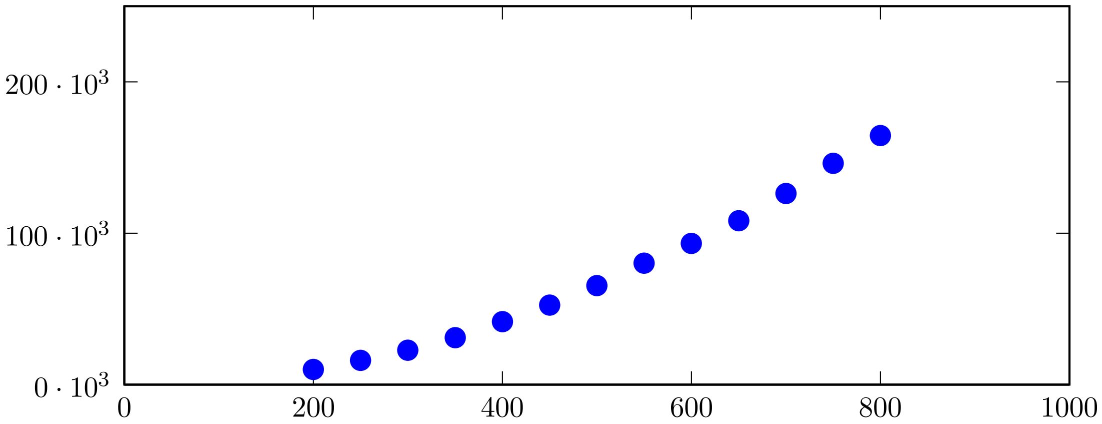

```
\readdata{\data}{data.data}
\psset{xunit=12.5cm,yunit=0.2mm}
\begin{pspicture}(-0.080,-30)(1,270)
\pstScalePoints(1,1){1000 div}{1000 div}
\psaxes[Dx=200,dx=2.5cm,Dy=100,ticksize=0 5pt,tickstyle=inner,
   subticks=10,ylabelFactor=\cdot10^3,dy=2cm](0,0)(1,250)
\listplot[nStart=200,linewidth=3pt,
   linecolor=blue,plotstyle=dots]{\data}
\listplot[linewidth=1pt,linecolor=blue]{\data}
\end{pspicture}
```

Example for nEnd/xEnd




```
\readdata{\data}{data.data}
\psset{xunit=12.5cm,yunit=0.2mm}
\begin{pspicture}(-0.080,-30)(1,270)
\pstScalePoints(1,1){1000 div}{1000 div}
\psaxes[axesstyle=frame,Dx=200,dx=2.5cm,Dy=100,ticksize=0 5pt,tickstyle=
  inner,
   subticks=10,ylabelFactor=\cdot10^3,dy=2cm](0,0)(1,250)
\listplot[nStart=200,linewidth=3pt,
   linecolor=blue]{\data}
\listplot[linewidth=1pt,linecolor=blue]{\data}
\end{pspicture}
```

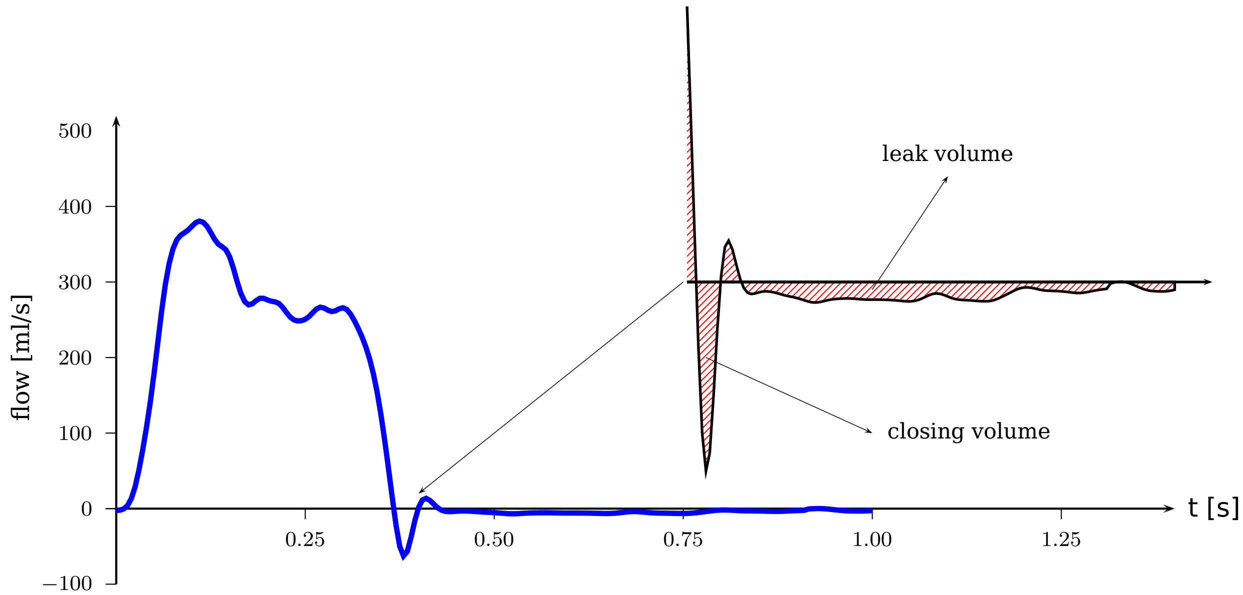
Example for all new options




```
\readdata{\data}{data.data}
\psset{xunit=12.5cm,yunit=0.2mm}
\begin{pspicture}(-0.080,-30)(1,270)
\pstScalePoints(1,1){1000 div}{1000 div}
\psaxes[axesstyle=frame,Dx=200,dx=2.5cm,Dy=100,,ticksize=0 5pt,tickstyle=
  inner,
   ylabelFactor=\cdot10^3,dy=2cm](0,0)(1,250)
\listplot[nStart=200, nEnd=800, nStep=50,
   linewidth=3pt,linecolor=blue,plotstyle=dots]{\data}
\end{pspicture}
```

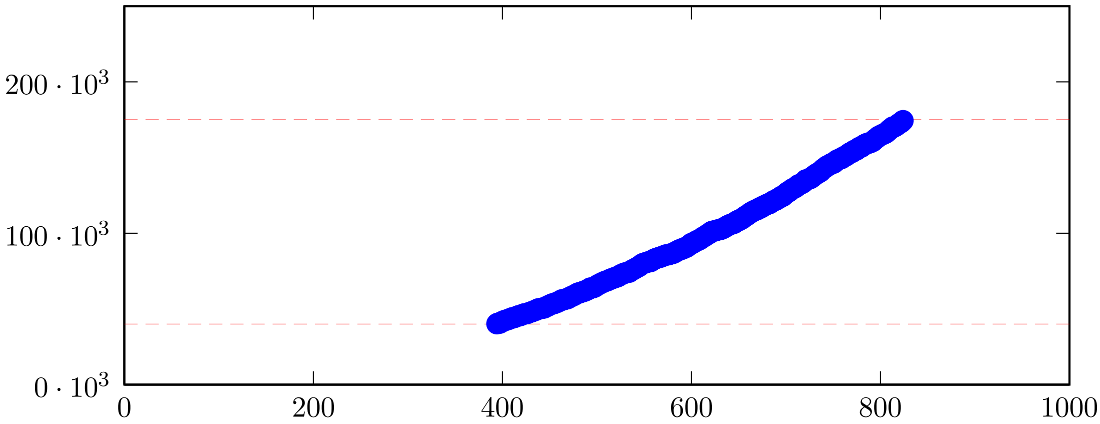
Example for xStart

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 zoom to the original graphic.




```
\psset{xunit=10cm, yunit=0.01cm}
\readdata{\data}{data3.data}
\begin{pspicture}(-0.1,-100)(1.5,700.0)
  \psaxes[Dx=0.25,Dy=100,dy=100\psyunit,ticksize=-4pt 0,%
    labelFontSize={\scriptstyle}]{->}(0,0)(0,-100)(1.4,520)
  \uput[0](1.4,0){\textsf{t [s]}}
  \rput(-0.125,200){\psrotateleft{\small 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 leak volume}
  \psline[linewidth=.01]{->}(0.78,200)(1,100)
  \rput[l](1.02,100){\footnotesize closing volume}
\end{pspicture}
```

Example for yStart/yEnd




```
\readdata{\data}{data.data}
\psset{xunit=12.5cm,yunit=0.2mm}
\begin{pspicture}(-0.080,-30)(1,270)
\pstScalePoints(1,1){1000 div}{1000 div}
\psaxes[axesstyle=frame,Dx=200,dx=2.5cm,Dy=100,ticksize=0 5pt,tickstyle=
  inner,
   ylabelFactor=\cdot10^3,dy=2cm](0,0)(1,250)
  \psset{linewidth=0.1pt, linestyle=dashed,linecolor=red}
  \psline(0,40)(1,40)
  \psline(0,175)(1,175)
  \listplot[yStart=40000, yEnd=175000,linewidth=3pt,linecolor=blue,plotstyle
    =dots]{\data}
\end{pspicture}
```

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
...
```

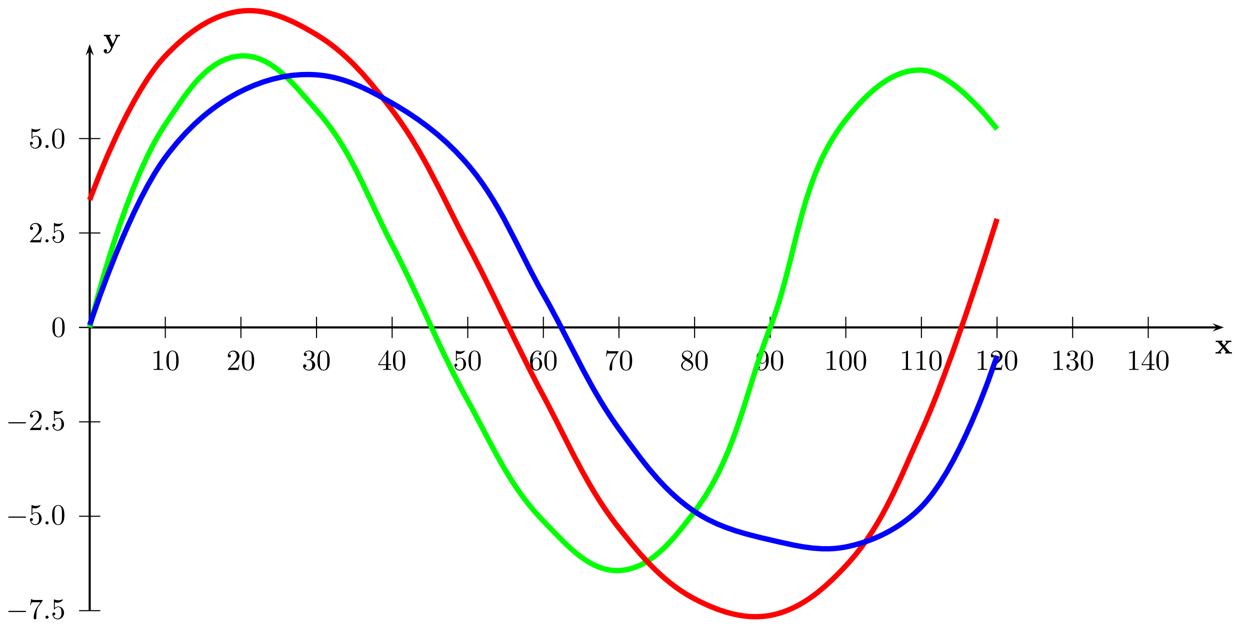
you can select the y value which should be plotted. The option `plotNo` marks the plotted value (default 1) and the option `plotNoMax` tells `pst-plot` how many y values are present. There are no real restrictions in the maximum number for `plotNoMax`.

We have the following data file:

```
[% file data.data
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
]%
```

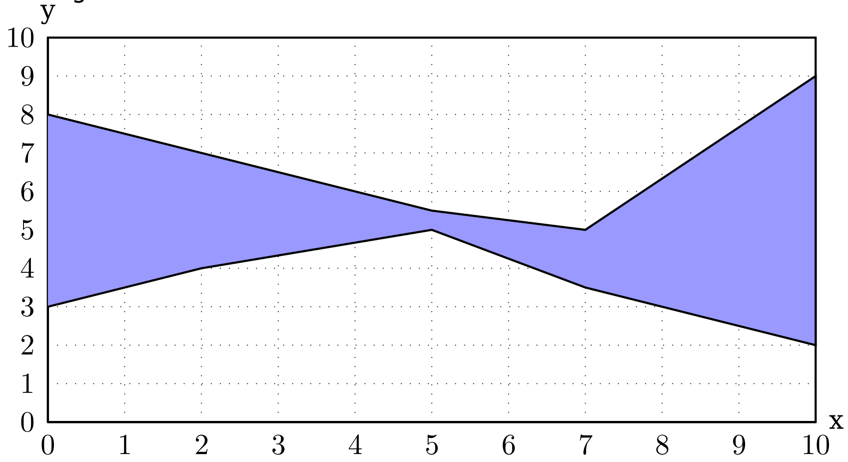
which holds data records for multiple plots (`x y1 y2 y3`). This can be plotted without any modification to the data file:




```
\readdata\Data{dataMul.data}
\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)[$\mathbf{x}$,-90][$\mathbf{y
  }$,0]
\psset{linewidth=2pt,plotstyle=curve}
\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}
```

Example for changeOrder

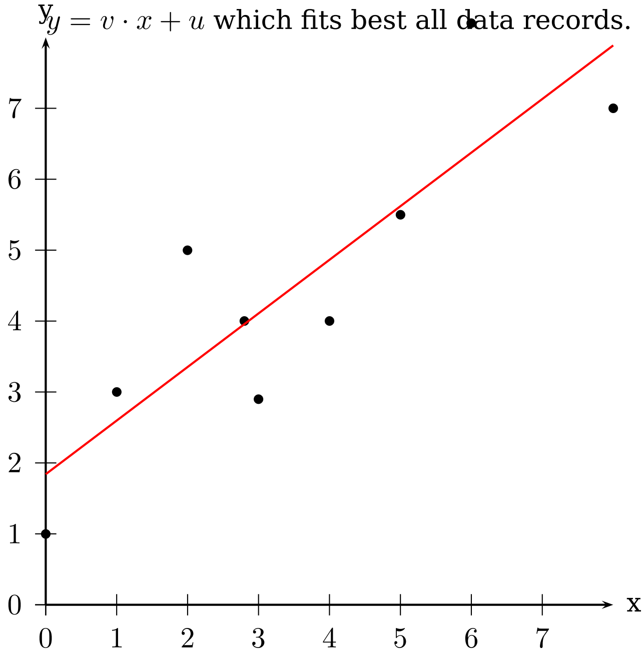
It is only possible to fill the region between two listplots with `\pscustom` if one of them has the values in reverse order. Otherwise we do not get a closed path. With the option `ChangeOrder` the values are used in reverse order:




```
\begin{filecontents*}{test.data}
 0 3 8
 2 4 7
 5 5 5.5
 7 3.5 5
 10 2 9
\end{filecontents*}
\psset{lly=-.5cm}
\begin{psgraph}[axesstyle=frame,ticklinestyle=dotted,ticksize=0 10](0,0)
  (10,10){4in}{2in}%
  \readdata{\data}{test.data}%
  \pscustom[fillstyle=solid,fillcolor=blue!40]{%
    \listplot[plotNo=2,plotNoMax=2]{\data}%
    \listplot[plotNo=1,plotNoMax=2,ChangeOrder]{\data}}
\end{psgraph}
```

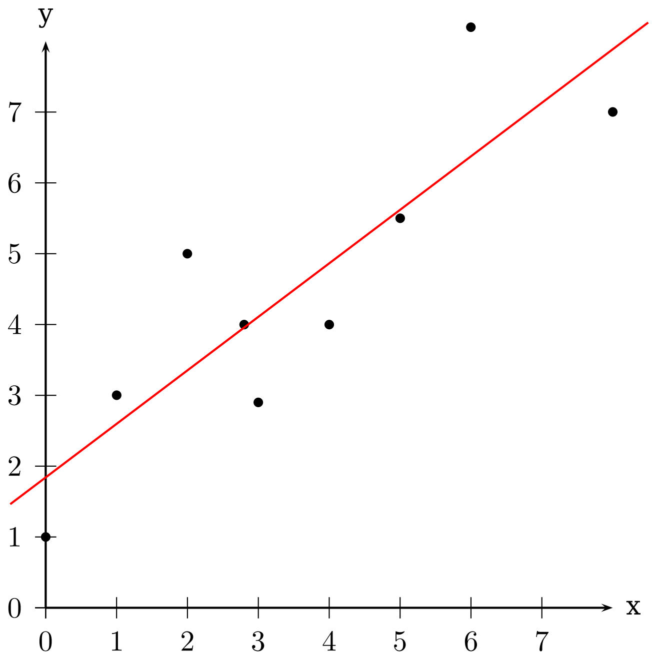
Example for plotstyle

The plotstyle option is defined in the package pst-plot, but its value LSM (**L**east **S**quare **Method**) is only valid for the pstricks-add package. Instead of plotting the data records as dots or a line, the \listplot macro calculates the values for a line $y = v \cdot x + u$ which fits best all data records.




```
\begin{filecontents*}{LSM.data}
0 1 1 3 2.8 4 3 2.9 2 5 4 4 5 5.5 6 8.2 8 7
\end{filecontents*}
\psset{lly=-.5cm}
\readdata{\data}{LSM.data}
\begin{psgraph}[arrows=->](0,0)(0,0)(8,8){.5\textwidth}{!}
  \listplot[plotstyle=dots]{\data}
  \listplot[plotstyle=LSM,linecolor=red]{\data}
\end{psgraph}
```

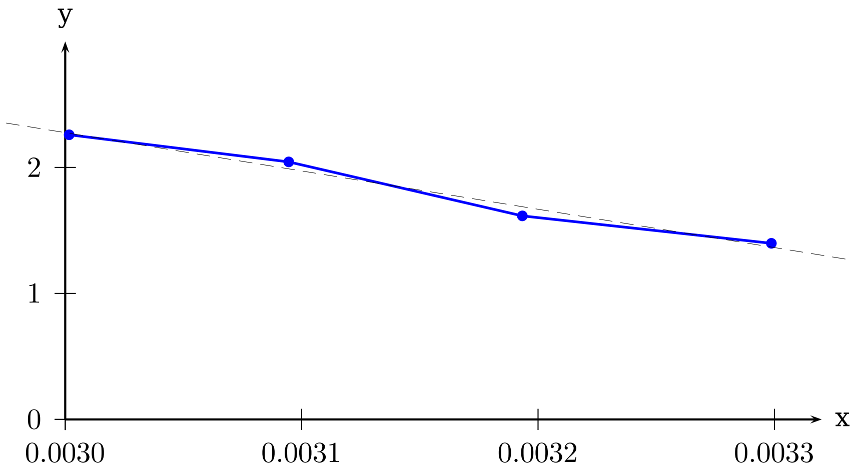
The macro looks for the lowest and biggest x-value and draws the line for this interval. It is possible to pass other values to the macro by setting the xStart and/or xEnd options. They are preset with an empty value {}.



y=0.755679 x+1.84105

```
\begin{filecontents*}{LSM.data}
0 1 1 3 2.8 4 3 2.9 2 5 4 4 5 5.5 6 8.2 8 7
\end{filecontents*}
\readdata{\data}{LSM.data}
\psset{lly=-1.75cm}
\begin{psgraph}[arrows=->](0,0)(0,0)(8,8){.5\textwidth}{!}
  \listplot[plotstyle=dots]{\data}
  \listplot[PstDebug=1,plotstyle=LSM,xStart=-0.5,xEnd=8.5,linecolor=red]{\
    data}
\end{psgraph}
```

With PstDebug=1 one gets the equation $y = v \cdot x + u$ printed, beginning at the position (0|-50pt). This cannot be changed, because it is only for some kind of debugging. Pay attention for the correct xStart and xEnd values, when you use the \pstScalePoints Macro. In the following example we use an x-interval from 0 to 3 to plot the values; first we subtract 0.003 from all x-values and then scale them with 10000. This is not taken into account for the xStart and xEnd values.



y=-0.162184 x+2.27634

```
\begin{filecontents*}{LSM.data}
0.003298697 1.397785583
0.003193358 1.615489564
0.003094538 2.044019006
0.003001651 2.259240127
\end{filecontents*}
\readdata{\data}{LSM.data}
\pstScalePoints(10000,1){ 0.003 sub }{}
\psset{lly=-1.75cm}
\psgraph[arrows=->,Ox=0.0030,Dx=0.0001,dx=\psxunit](0,0)(3.2,3){10cm}{5cm}
  \listplot[showpoints=true,linewidth=1pt,linecolor=blue]{\data}
  \listplot[PstDebug=1,plotstyle=LSM,linewidth=0.1pt,linestyle=dashed,%
    xStart=-0.25,xEnd=3.3]{\data}
\endpsgraph
```

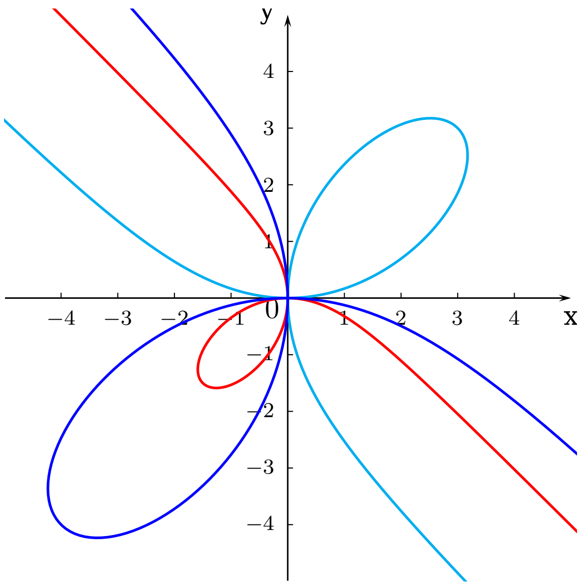
27. Polar plots

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

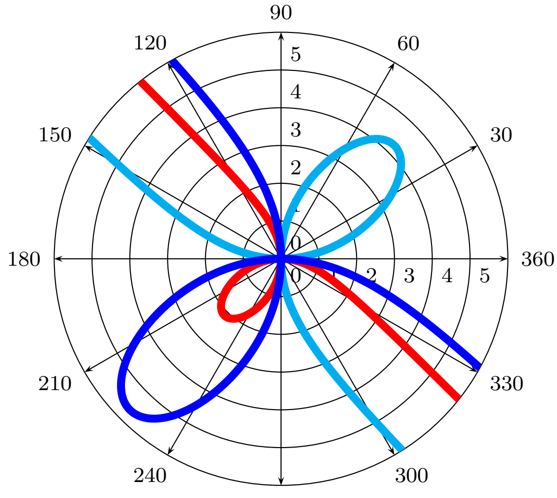
`\psplot` `[polarplot=true,...]` {<*start angle*>}{<*end angle*>}%
`[PS command]` {<*r(alpha)*>}

The equation in PostScript code is interpreted as a function $r = f(\alpha)$, e.g. for the circle with radius 1 as $r = \sqrt{\sin^2 x + \cos^2 x}$, or $r = a * \frac{sin(x) * cos(x)}{(sin(x)^3 + cos(x)^3)}$ for the following examples:

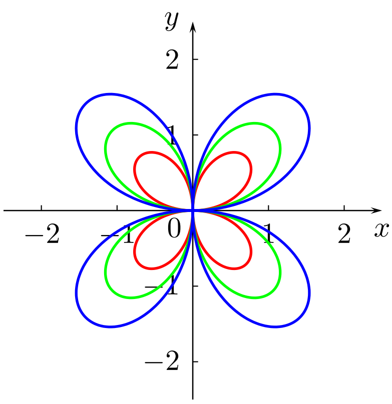
```
x sin dup mul x cos dup mul add sqrt
```



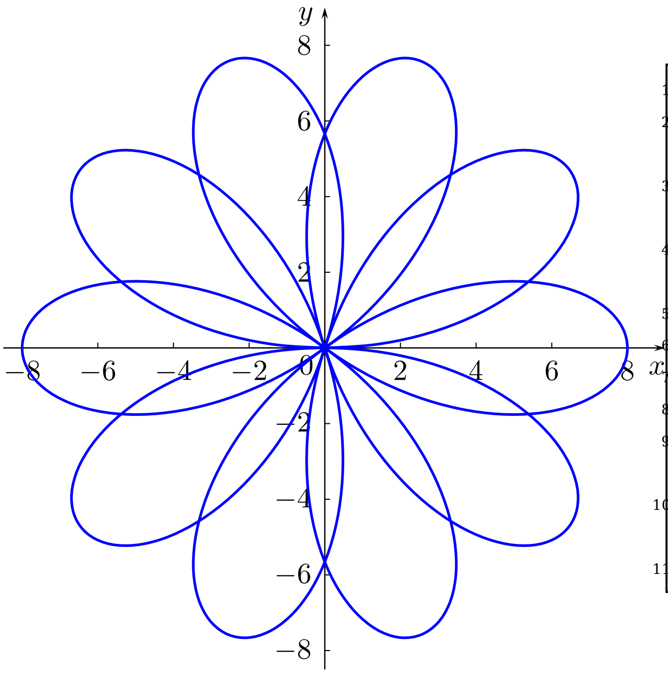

```
\psset{plotpoints=200,unit=0.75}
\begin{pspicture*}(-5,-5)(5.1,5.1)
  \psaxes[arrowlength=1.75,ticksize=2pt,labelFontSize=\scriptstyle,
    linewidth=0.2mm]{->}(0,0)(-4.99,-4.99)(5,5)[x,-90][y,180]
  \rput[Br](-.15,-.35){$0$} \psset{linewidth=.35mm,polarplot}
  \psplot[linecolor=red]{140}{310}{3 neg x sin mul x cos mul x sin 3 exp x
    cos 3 exp add div}
  \psplot[linecolor=cyan]{140}{310}{6 x sin mul x cos mul x sin 3 exp x cos
    3 exp add div}
  \psplot[linecolor=blue,algebraic]{2.44}{5.41}{-8*sin(x)*cos(x)/(sin(x)^3+
    cos(x)^3)}
\end{pspicture*}
```




```
\psset{unit=0.5cm}
\begin{pspicture}(-6,-6)(6,6)
\psaxes[axesstyle=polar,labelFontSize=\scriptstyle,linewidth=0.2mm]{->}(6,6)
\psset{linewidth=3pt,polarplot,plotpoints=500,plotstyle=curve}
\psclip{\pscircle[linestyle=none]{6}}
  \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 x sin mul x cos mul x sin 3 exp x cos 3 exp add div}
  \psplot[linecolor=blue,algebraic]{2.44}{5.41}{-8*sin(x)*cos(x)/(sin(x)^3+cos(x)^3)}
\endpsclip
\end{pspicture}
```




```
\psset{plotpoints=200,unit=1}
\begin{pspicture}(-2.5,-2.5)(2.5,2.5)% Ulrich Dirr
 \psaxes[arrowlength=1.75,%
    ticksize=2pt,linewidth=0.17mm]{->}%
      (0,0)(-2.5,-2.5)(2.5,2.5)[$x$,-90][$y$,180]
  \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}
```



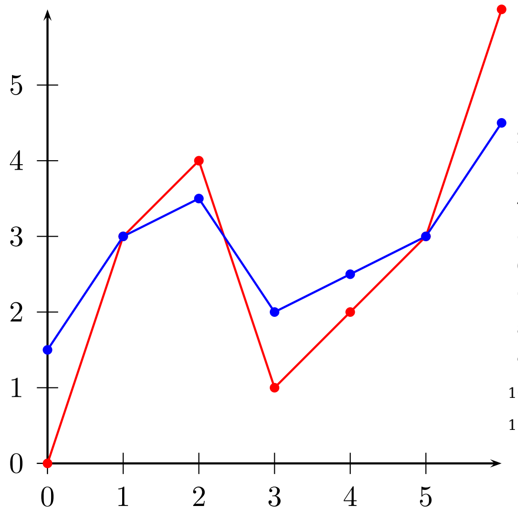

```
\psset{plotpoints=200,unit=0.5}
\begin{pspicture}(-8.5,-8.5)
  (9,9)% Ulrich Dirr
\psaxes[Dx=2,dx=2,Dy=2,dy=2,
  arrowlength=1.75,
    ticksize=2pt,linewidth=0.17mm
      ]{->}(0,0)(-8.5,-8.5)(9,9)
\rput[Br](9,-.7){$x$}
\rput[tr](-.3,9){$y$}
\rput[Br](-.3,-.7){$0$}
%
\psset{linewidth=.35mm,plotstyle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

28. \pstScalePoints

The syntax is

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

`xScale,yScale` are decimal values used as scaling factors, the `xPs` and `yPS` are additional PostScript code applied 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.

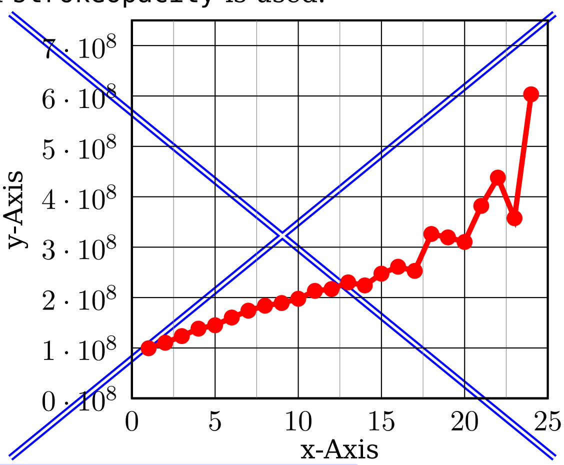
Part IV. New commands and environments

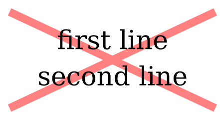
29. psCancel environment[1]

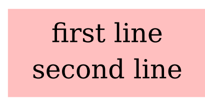
This macro works like the `\cancel` macro from the package of the same name but it allows as argument any contents, not only letters but also a complex graphic.

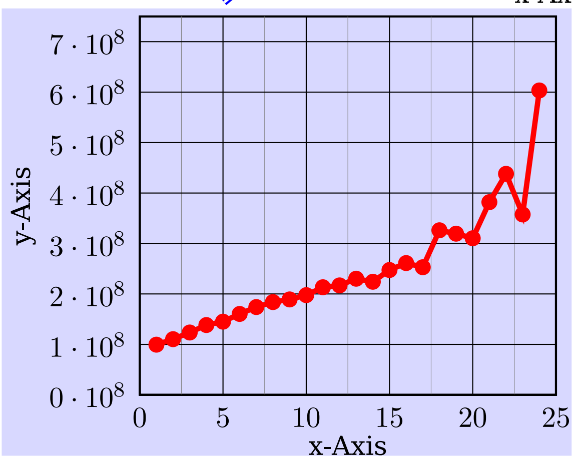
`\psCancel*[Options]{contents}`

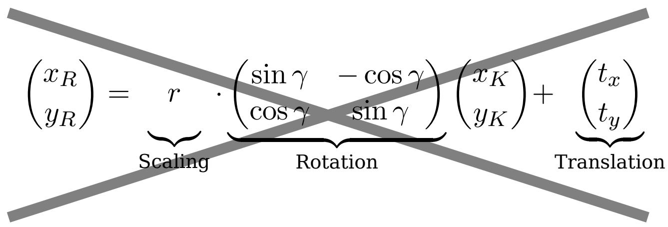
All optional arguments for lines and boxes are valid and can be used in the usual way. The star option fills the underlying box rectangle with the linecolor. This can be transparent if `opacity` is set to a value less than 1. This can be used in presentation to strike out words, equations, and graphic objects. Lines can also be transparent when the option `strokeopacity` is used.











1 Thanks to by Stefano Baroni

```
\psCancel{A} \psCancel[linecolor=red]{Tikz :-)} \quad
\psCancel[linecolor=blue,doubleline=true]{%
  \readdata{\data}{demo1.data}
  \psset{shift=*,xAxisLabel=x-Axis,yAxisLabel=y-Axis,llx=-13mm,lly=-7mm,
      xAxisLabelPos={c,-1},yAxisLabelPos={-7,c}}
  \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}} \qquad% end of Cancel
\psCancel[linewidth=3pt,linecolor=red,
    strokeopacity=0.5]{\tabular[b]{c}first line\\second line\endtabular}\quad
\psCancel*[linecolor=red!50,opacity=0.5]{\tabular[b]{c}first line\\second
  line\endtabular}
\quad
\psCancel*[linecolor=blue!30,opacity=0.5]{%
  \readdata{\data}{demo1.data}
  \psset{shift=*,xAxisLabel=x-Axis,yAxisLabel=y-Axis,llx=-15mm,lly=-7mm,urx
    =1mm,
      xAxisLabelPos={c,-1},yAxisLabelPos={-7,c}}
  \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}} \quad% end of Cancel
\psCancel[linewidth=4pt,strokeopacity=0.5]{\parbox{8cm}{\[
  \binom{x_R}{y_R} = \underbrace{r\vphantom{\binom{A}{B}}}_{\text{Scaling}}\
    cdot
    \underbrace{\begin{pmatrix}
        \sin\gamma & -\cos\gamma \\
      \cos \gamma & \sin \gamma \\
      \end{pmatrix}}_{\text{Rotation}} \binom{x_K}{y_K} +
  \underbrace{\binom{t_x}{t_y}}_{\text{Translation}} \]} }% end of psCancel
```

30. psgraph environment

This new environment psgraph 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:

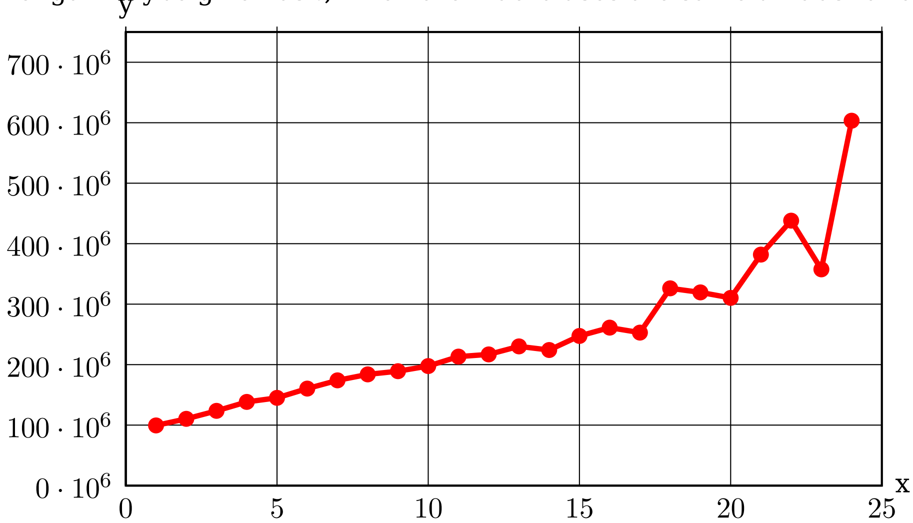
```
\psgraph [Options] {<arrows>}%
    (xOrig,yOrig)(xMin,yMin)(xMax,yMax){xLength}{yLength}
...
\endpsgraph

\begin{psgraph} [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.

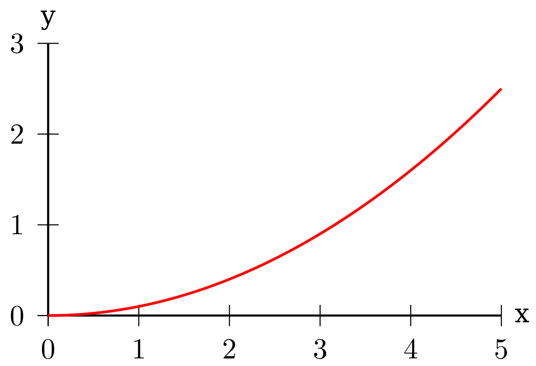
- if `(xOrig,yOrig)` is missing, it is substituted to `(xMin,xMax)`;
- if `(xOrig,yOrig)` **and** `(xMin,yMin)` are missing, they are both substituted to `(0,0)`.

The y-length maybe given as !, when the macro uses the same unit as for the x-axis.

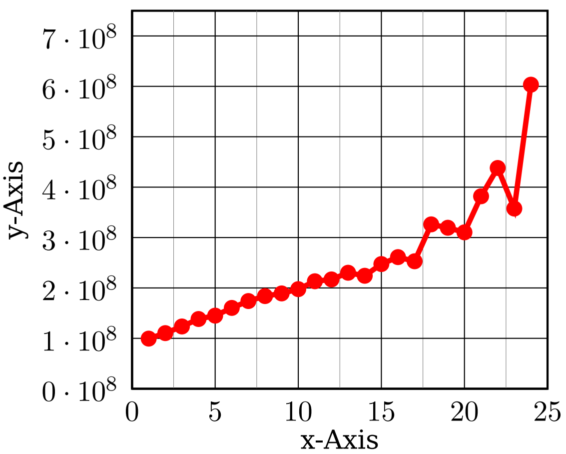



```
\readdata{\data}{demo1.data}
\pstScalePoints(1,0.000001){}{}% (x,y){additional x operator}{y op}
\psset{llx=-1cm,lly=-1cm}
\begin{psgraph}[axesstyle=frame,xticksize=0 759,yticksize=0 25,%
   subticks=0,ylabelFactor=\cdot 10^6,
   Dx=5,dy=100\psyunit,Dy=100](0,0)(25,750){10cm}{6cm} % parameters
 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

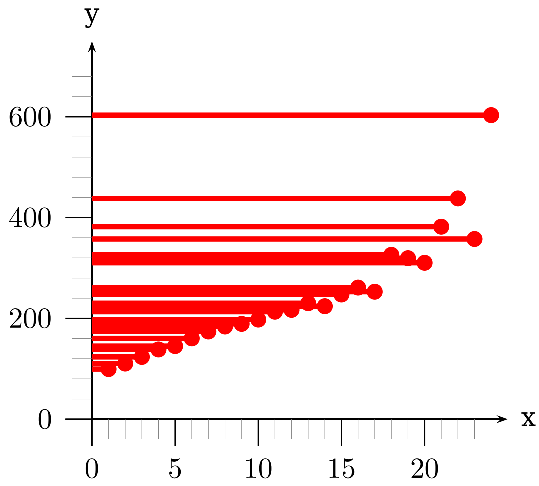
In the following example, the y unit gets the same value as the one for the x-axis.



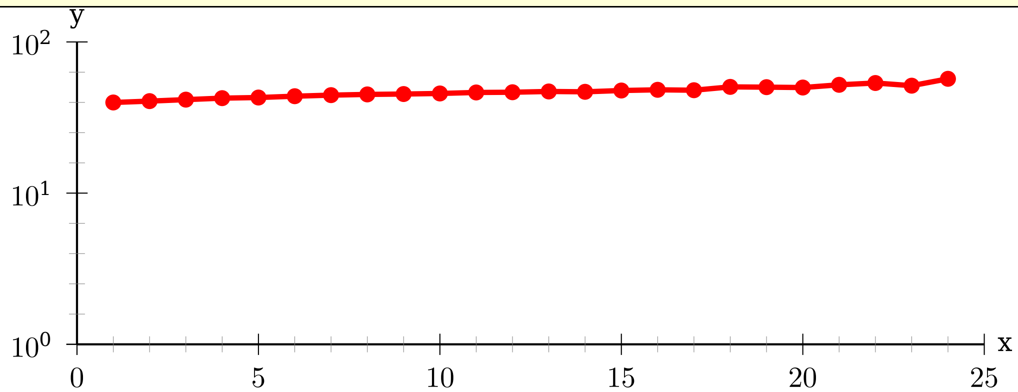

```
\psset{llx=-1cm,lly=-0.5cm,ury=0.5cm}
\begin{psgraph}(0,0)(5,3){6cm}{!} % x-y-axis with same unit
  \psplot[linecolor=red,linewidth=1pt]{0}{5}{x dup mul 10 div}
\end{psgraph}
```



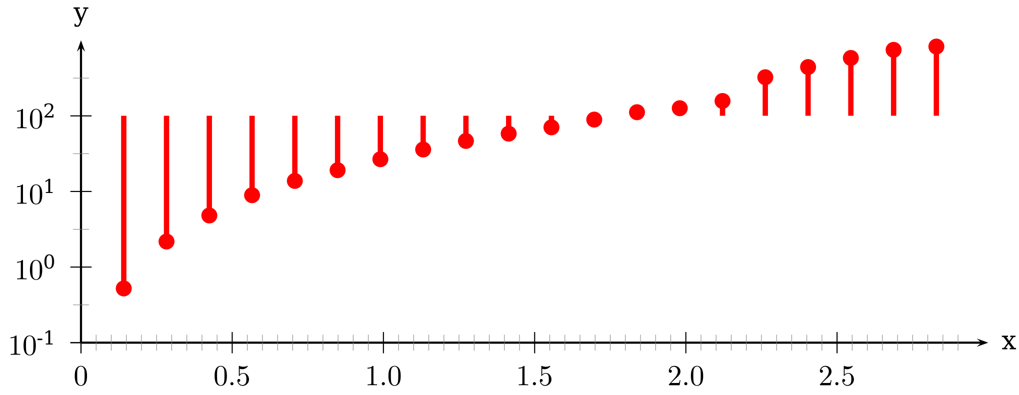

```
\readdata{\data}{demo1.data}
\psset{xAxisLabel=x-Axis,yAxisLabel=y-Axis,llx=-.5cm,lly=-1cm,ury=0.5cm,
  xAxisLabelPos={c,-1},yAxisLabelPos={-7,c}}
\pstScalePoints(1,0.00000001){}{}
\begin{psgraph}[axesstyle=frame,xticksize=0 7.5,yticksize=0 25,subticksize
  =1,
    ylabelFactor=\cdot 10^8,Dx=5,Dy=1,xsubticks=2](0,0)(25,7.5){5.5cm}{5cm}
  \listplot[linecolor=red, linewidth=2pt, showpoints=true]{\data}
\end{psgraph}
```



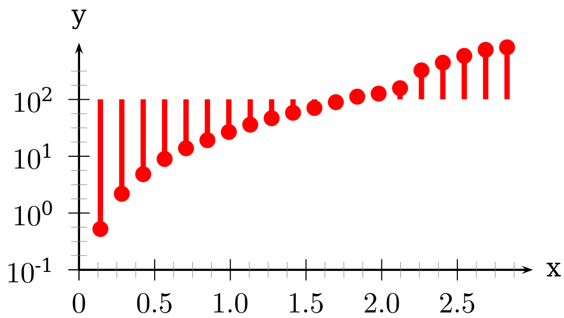

```
\readdata{\data}{demo1.data}
\psset{llx=-0.5cm,lly=-1cm}
\pstScalePoints(1,0.000001){}{}
\psgraph[arrows=->,Dx=5,dy=200\psyunit,Dy=200,subticks=5,ticksize=-10pt 0,
   tickwidth=0.5pt,subtickwidth=0.1pt](0,0)(25,750){5.5cm}{5cm}
\listplot[linecolor=red,linewidth=2pt,showpoints=true,plotstyle=LineToYAxis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



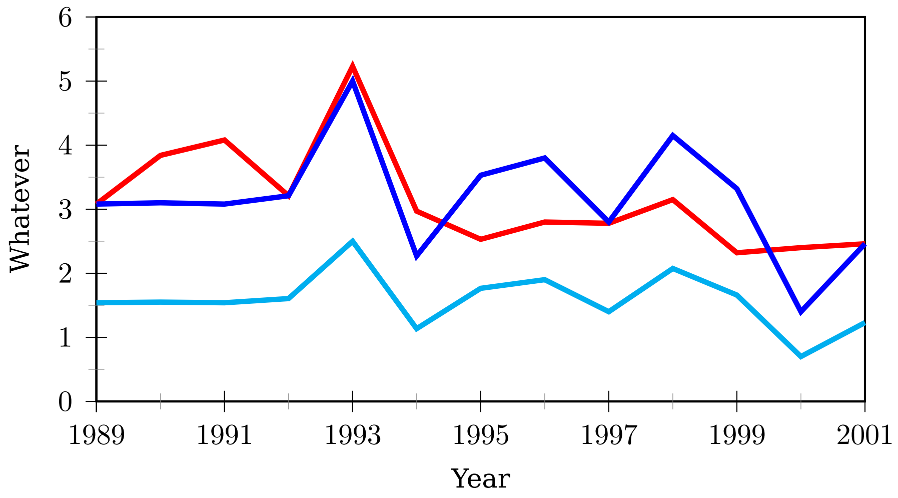

```
\readdata{\data}{demo1.data}
\pstScalePoints(1,0.2){}{log}
\psset{lly=-0.75cm}
\psgraph[ylogBase=10,Dx=5,Dy=1,subticks=5](0,0)(25,2){12cm}{4cm}
  \listplot[linecolor=red, linewidth=2pt, showpoints=true]{\data}
\endpsgraph
```



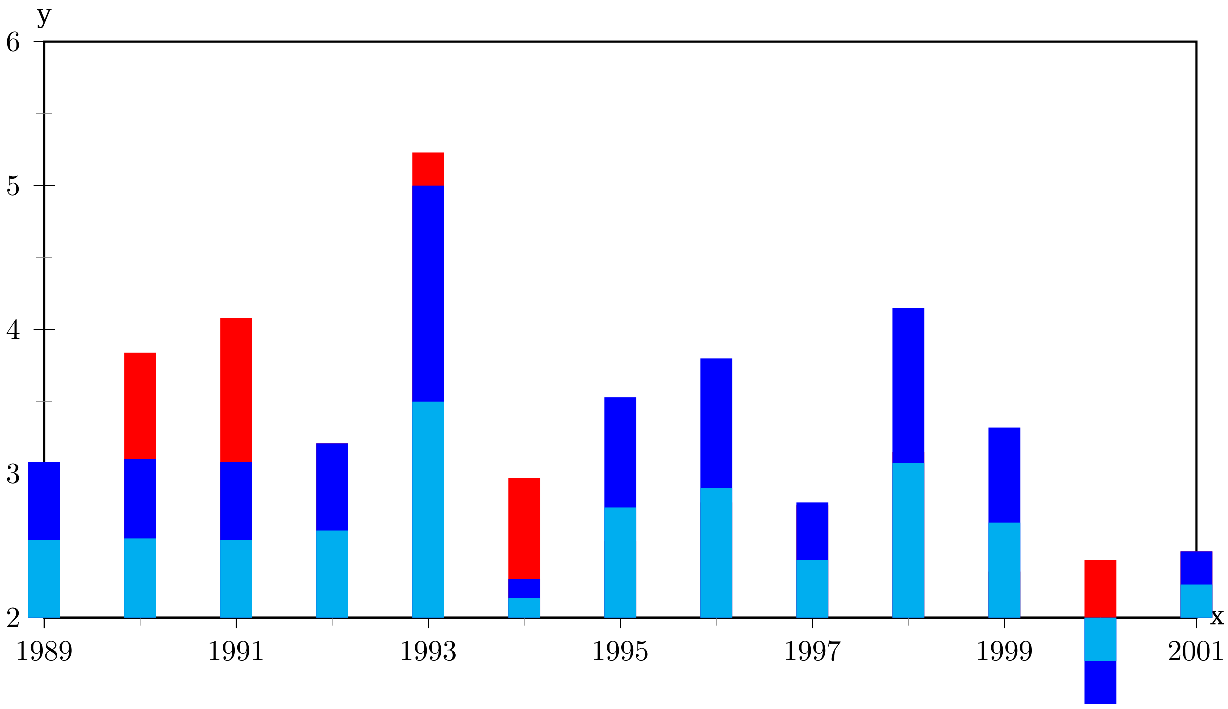

```
\readdata{\data}{demo0.data}
\psset{lly=-0.75cm,ury=0.5cm}
\pstScalePoints(1,1){}{log}
\begin{psgraph}[arrows=->,Dx=0.5,ylogBase=10,Oy=-1,xsubticks=10,%
    ysubticks=2](0,-3)(3,1){12cm}{4cm}
  \listplot[linecolor=red,linewidth=2pt,showpoints=true,plotstyle=
    LineToXAxis]{\data}
\end{psgraph}
```




```
\psset{lly=-0.75cm,ury=0.5cm}
\readdata{\data}{demo0.data}
\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,plotstyle=
    LineToXAxis]{\data}
\endpsgraph
```



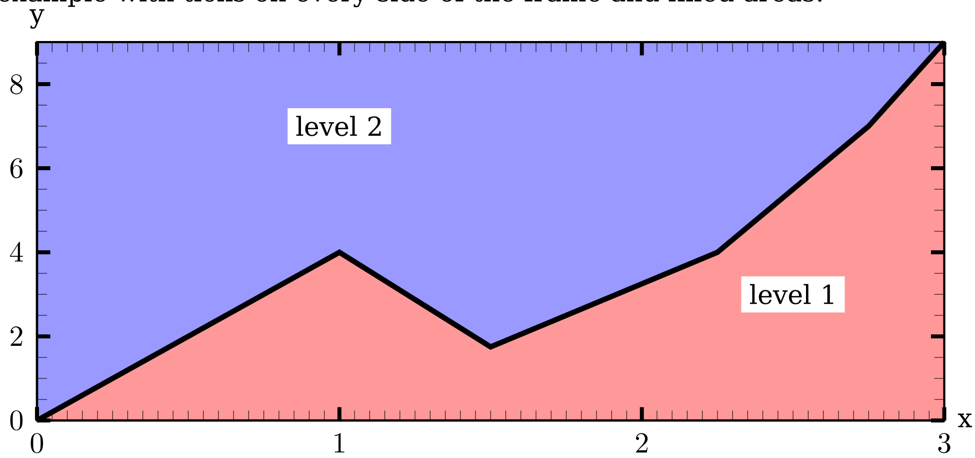

```
\readdata{\data}{demo2.data}%
\readdata{\dataII}{demo3.data}%
\pstScalePoints(1,1){1989 sub}{}
\psset{llx=-0.5cm,lly=-1cm, xAxisLabel=Year,yAxisLabel=Whatever,%
    xAxisLabelPos={c,-0.4in},yAxisLabelPos={-0.4in,c}}
\psgraph[axesstyle=frame,Dx=2,Ox=1989,subticks=2](0,0)(12,6){4in}{2in}%
  \listplot[linecolor=red,linewidth=2pt]{\data}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




```
\readdata{\data}{demo2.data}%
\readdata{\dataII}{demo3.data}%
\psset{llx=-0.5cm,lly=-0.75cm,plotstyle=LineToXAxis}
```

```
\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=12pt]{\data}
  \listplot[linecolor=blue,linewidth=12pt]{\dataII}
  \listplot[linecolor=cyan,linewidth=12pt,yunit=0.5]{\dataII}
\end{psgraph}
```

An example with ticks on every side of the frame and filled areas:

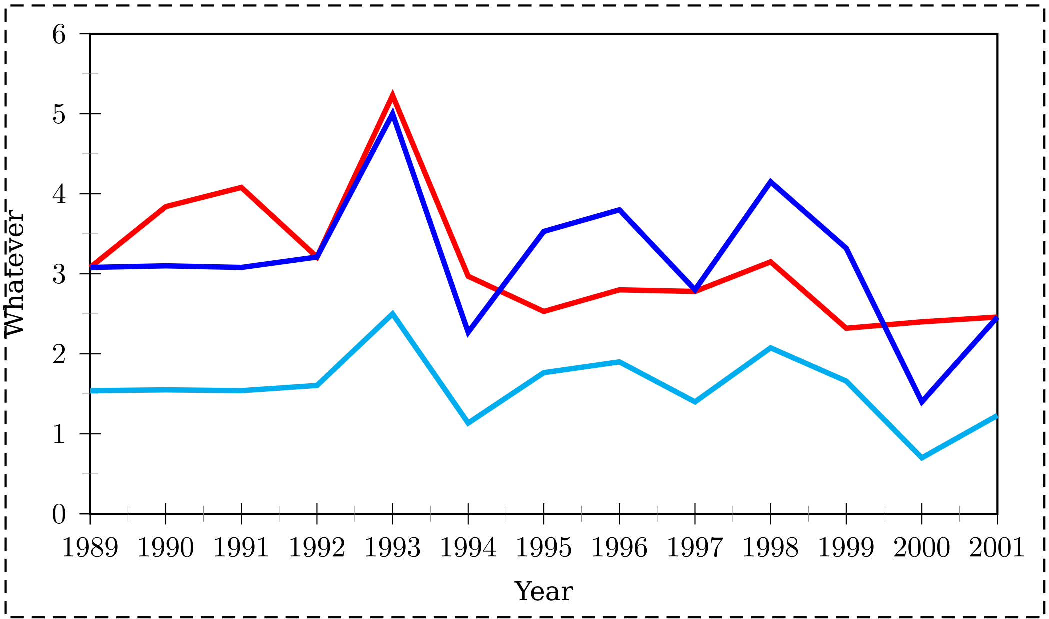



```
\def\data{0 0 1 4 1.5 1.75 2.25 4 2.75 7 3 9}
\psset{lly=-0.5cm}
\begin{psgraph}[axesstyle=none,ticks=none](0,0)(3.0,9.0){12cm}{5cm}
  \pscustom[fillstyle=solid,fillcolor=red!40,linestyle=none]{%
    \listplot{\data}
    \psline(3,9)(3,0)}
  \pscustom[fillstyle=solid,fillcolor=blue!40,linestyle=none]{%
    \listplot{\data}
    \psline(3,9)(0,9)}
  \listplot[linewidth=2pt]{\data}
  \psaxes[axesstyle=frame,ticksize=0 5pt,xsubticks=20,ysubticks=4,
    tickstyle=inner,dy=2,Dy=2,tickwidth=1.5pt,subtickcolor=black](0,0)(3,9)
  \rput*(2.5,3){level 1}\rput*(1,7){level 2}
\end{psgraph}
```

30.1. 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

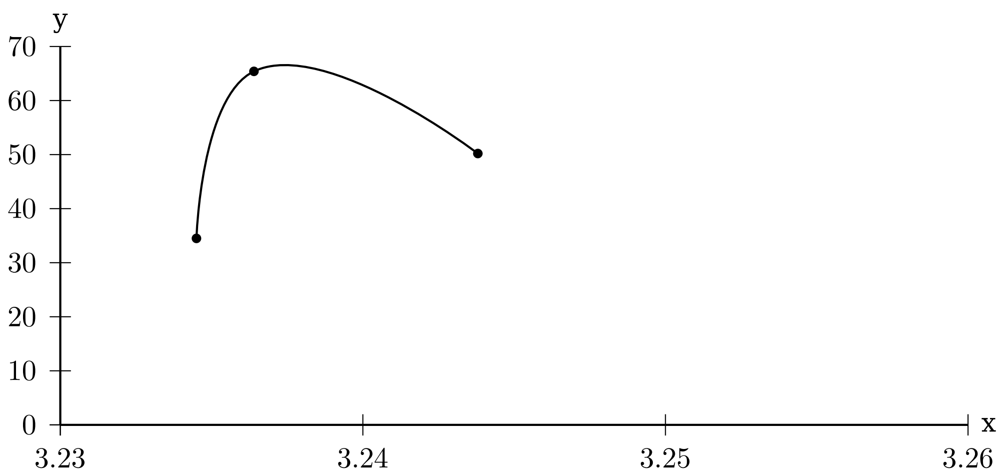
There is one restriction in using the trim parameters, they must been set **before** `\psgraph` is called. They are redundant when used as parameters of `\psgraph` itself. The `?AxisLabelPos` options can use the letter `c` for centering an x-axis or y-axis label. The `c` is a replacement for the x or y value. When using values with units, the position is always measured from the origin of the coordinate system, which can be outside the visible `pspicture` environment




```
\readdata{\data}{demo2.data}%
\readdata{\dataII}{demo3.data}%
\psset{llx=-1cm,lly=-1.25cm,urx=0.5cm,ury=0.1in,xAxisLabel=Year,%
  yAxisLabel=Whatever,xAxisLabelPos={c,-0.4in},%
  yAxisLabelPos={-0.4in,c}}
\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
  }{2.5in}%
  \listplot[linecolor=red,linewidth=2pt]{\data}%
  \listplot[linecolor=blue,linewidth=2pt]{\dataII}%
  \listplot[linecolor=cyan,linewidth=2pt,yunit=0.5]{\dataII}%
\end{psgraph}%
}
```

30.2. Problems

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




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

\psset{lly=-0.5cm,llx=-1cm}
\readdata{\data}{test.data}
\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 on the linewidth of the document.

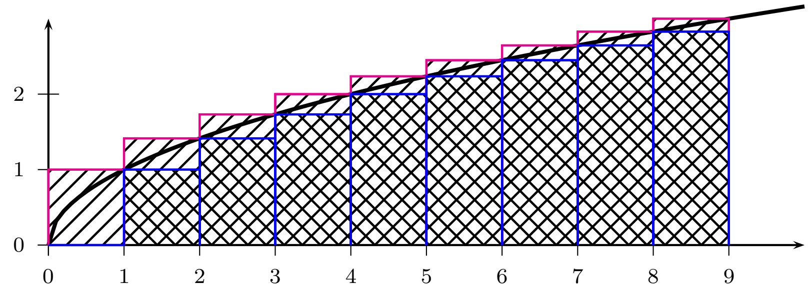
31. \psStep

`\psStep` calculates a step function for the upper or lower sum or the max/min of the Riemann integral definition of a given function. The available option is

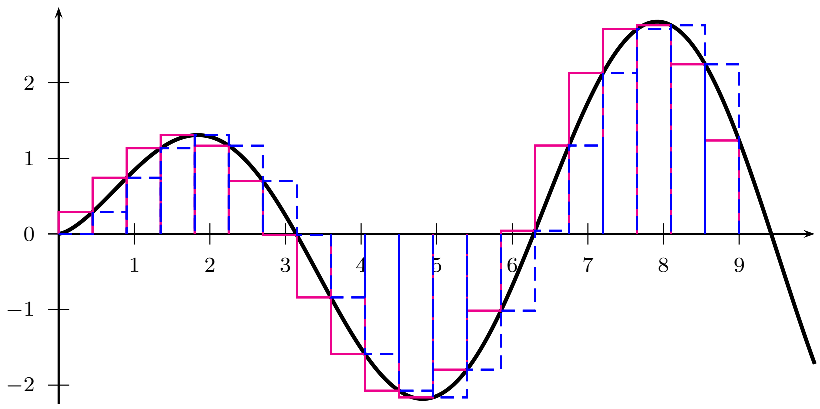
`StepType=lower |upper|Riemann|infimum|supremum` or alternative `StepType=l |u|R|i|s` with `lower` as the default setting. The syntax of the function is

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

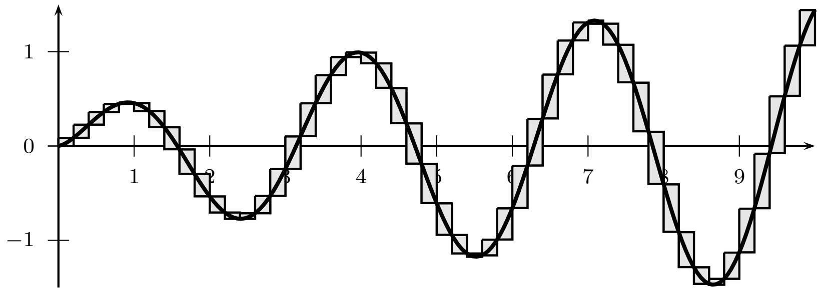
(x1,x2) is the given interval for the step wise calculated function, n is the number of the rectangles and *function* is the mathematical function in postfix or algebraic notation (with `algebraic=true`).



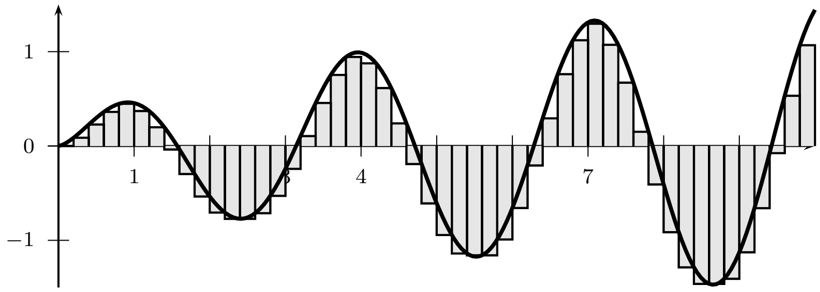
```
\begin{pspicture}(-0.5,-0.5)(10,3)
 \psaxes[labelFontSize=\scriptstyle]{->}(10,3)
 \psplot[plotpoints=100,linewidth=1.5pt,algebraic]{0}{10}{sqrt(x)}
 \psStep[linecolor=magenta,StepType=upper,fillstyle=hlines](0,9){9}{x sqrt}
 \psStep[linecolor=blue,fillstyle=vlines](0,9){9}{x sqrt }
\end{pspicture}
```



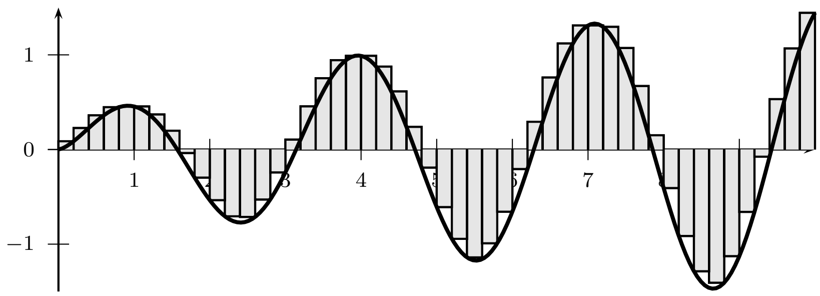
```
\psset{plotpoints=200}
\begin{pspicture}(-0.5,-2.25)(10,3)
  \psaxes[labelFontSize=\scriptstyle]{->}(0,0)(0,-2.25)(10,3)
 \psplot[linewidth=1.5pt,algebraic]{0}{10}{sqrt(x)*sin(x)}
 \psStep[algebraic,linecolor=magenta,StepType=upper](0,9){20}{sqrt(x)*sin(x)
   }
 \psStep[linecolor=blue,linestyle=dashed](0,9){20}{x sqrt x RadtoDeg sin mul
   }
\end{pspicture}
```



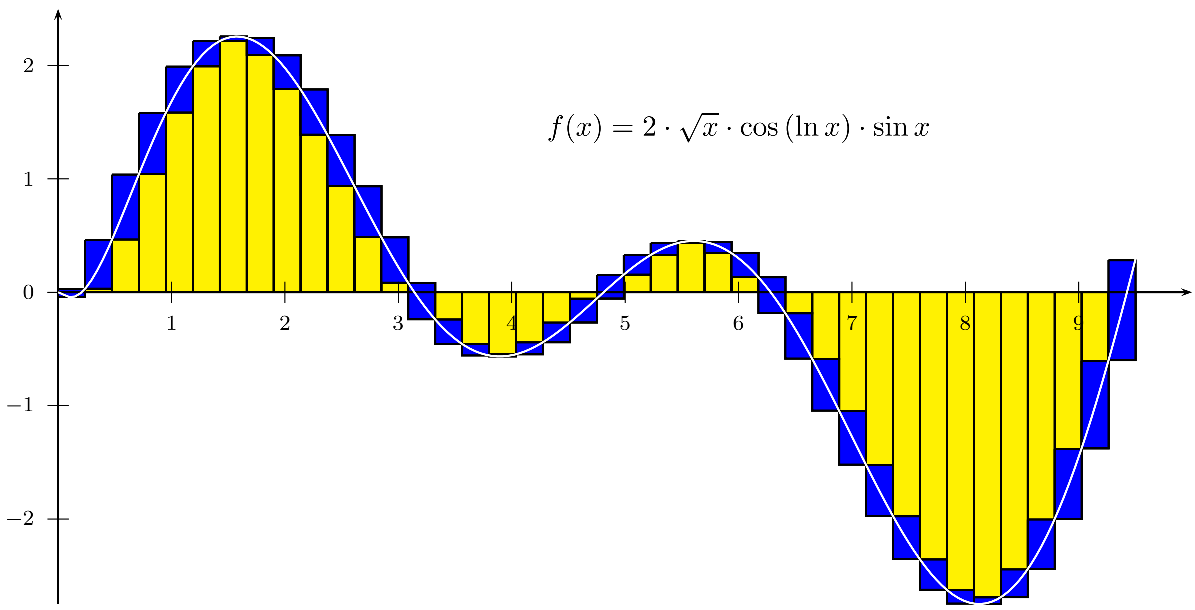

```
\psset{yunit=1.25cm,plotpoints=200}
\begin{pspicture}(-0.5,-1.5)(10,1.5)
 \psaxes[labelFontSize=\scriptstyle]{->}(0,0)(0,-1.5)(10,1.5)
 \psStep[algebraic,StepType=Riemann,fillstyle=solid,fillcolor=black
   !10](0,10){50}%
   {sqrt(x)*cos(x)*sin(x)}
 \psplot[linewidth=1.5pt,algebraic]{0}{10}{sqrt(x)*cos(x)*sin(x)}
\end{pspicture}
```




```
\psset{yunit=1.25cm,plotpoints=200}
\begin{pspicture}(-0.5,-1.5)(10,1.5)
 \psaxes[labelFontSize=\scriptstyle]{->}(0,0)(0,-1.5)(10,1.5)
 \psStep[algebraic,StepType=infimum,fillstyle=solid,fillcolor=black
   !10](0,10){50}%
   {sqrt(x)*cos(x)*sin(x)}
 \psplot[linewidth=1.5pt,algebraic]{0}{10}{sqrt(x)*cos(x)*sin(x)}
\end{pspicture}
```




```
\psset{yunit=1.25cm,plotpoints=200}
\begin{pspicture}(-0.5,-1.5)(10,1.5)
 \psaxes[labelFontSize=\scriptstyle]{->}(0,0)(0,-1.5)(10,1.5)
 \psStep[algebraic,StepType=supremum,fillstyle=solid,fillcolor=black!10](0,10){50}%
   {sqrt(x)*cos(x)*sin(x)}
 \psplot[linewidth=1.5pt,algebraic]{0}{10}{sqrt(x)*cos(x)*sin(x)}
\end{pspicture}
```




```
\psset{unit=1.5cm,plotpoints=200}
\begin{pspicture}[plotpoints=200](-0.5,-3)(10,2.5)
  \psStep[algebraic,fillstyle=solid,fillcolor=yellow](0.001,9.5){40}{2*sqrt(x)*cos(ln(x))*sin(x)}
  \psStep[algebraic,StepType=Riemann,fillstyle=solid,fillcolor=blue](0.001,9.5){40}{2*sqrt(x)*cos(ln(x))*sin(x)}
  \psaxes[labelFontSize=\scriptstyle]{->}(0,0)(0,-2.75)(10,2.5)
  \psplot[algebraic,linecolor=white]{0.001}{9.75}{2*sqrt(x)*cos(ln(x))*sin(x)}
  \uput[90](6,1.2){$f(x)=2\cdot\sqrt{x}\cdot\cos{(\ln{x})}\cdot\sin{x}$}
\end{pspicture}
```

32. \psplotTangent and option Tnormal

There is an additional option, named `Derive` for an alternative function (see following example) to calculate the slope of the tangent. This will be in general the first derivative, 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 may be problematic in some cases of special functions or x values, then it may be appropriate to use the Derive option.

`\psplotTangent` `*` `[Options]` $\{x\}\{dx\}\{function\}$

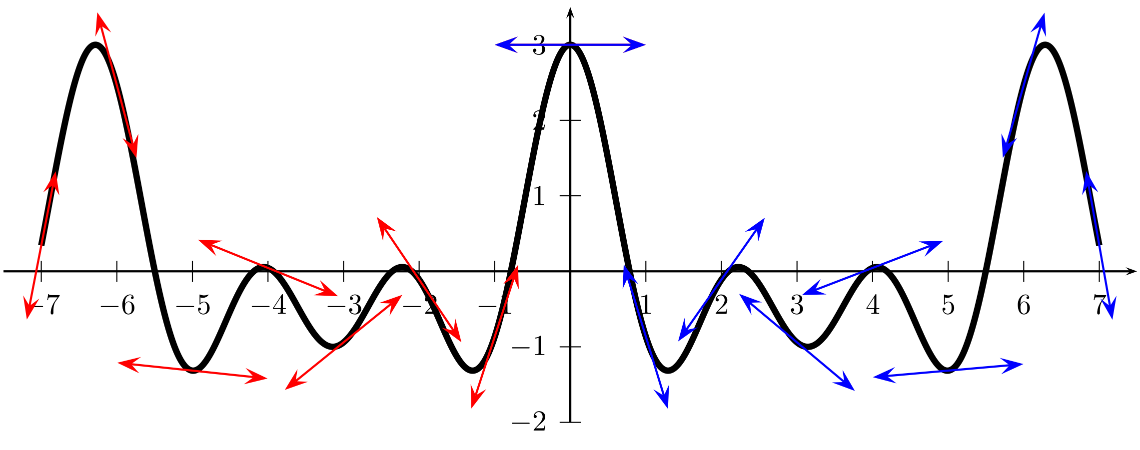
The macro expects three parameters:

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

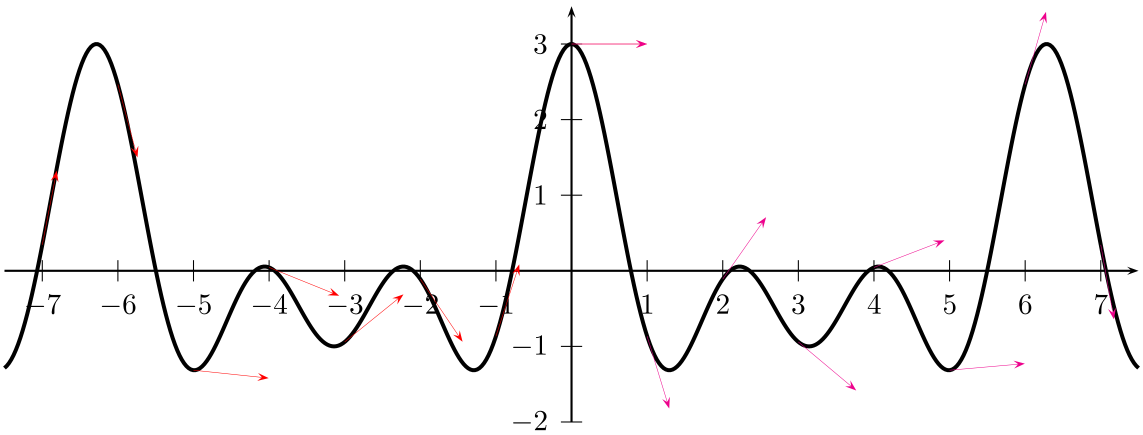
The following examples show the use of the algebraic option together with the Derive option. Remember that using the `algebraic` option implies that the angles have to be in the radian unit!




```
\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}}
```

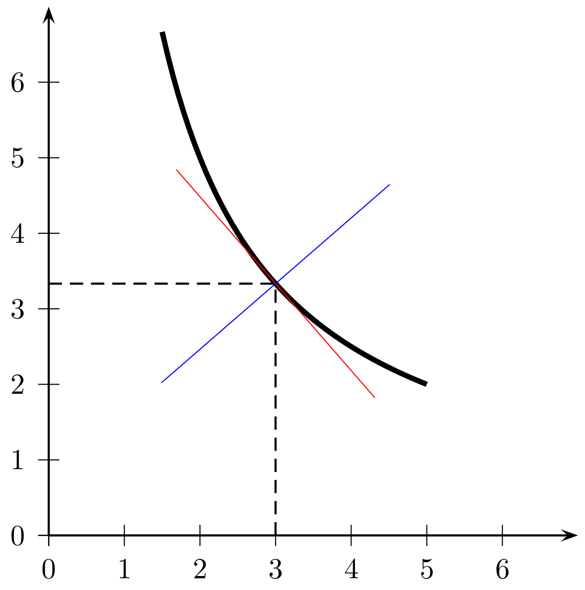
```
  \psset{linecolor=magenta, arrows=<->, arrowscale=2}%
  \multido{\n=0+1}{8}{\psplotTangent[linecolor=blue, Derive=\Fp]{\n}{1}{\F}}
\end{pspicture}
```

The star version plots only the tangent line in the positive x-direction:



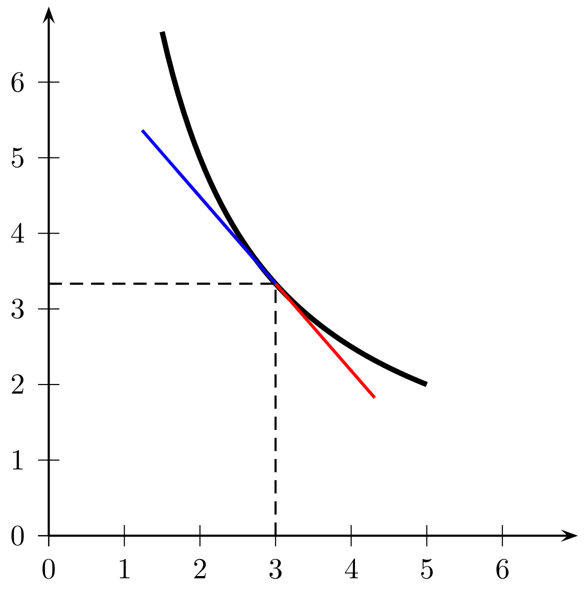

```
\def\Falg{cos(x)+cos(2*x)+cos(3*x)} \def\Fpalg{-sin(x)-2*sin(2*x)-3*sin(3*x)
  }
\begin{pspicture}(-7.5,-2.5)(7.5,4)%\psgrid
  \psaxes{->}(0,0)(-7.5,-2)(7.5,3.5)
  \psplot[linewidth=1.5pt,algebraic,plotpoints=500]{-7.5}{7.5}{\Falg}
  \multido{\n=-7+1}{8}{\psplotTangent*[linecolor=red,arrows=->,arrowscale=2,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

By setting the optional argument `Tnormal` one can plot the normal of the tangent line. It always starts at the given point.



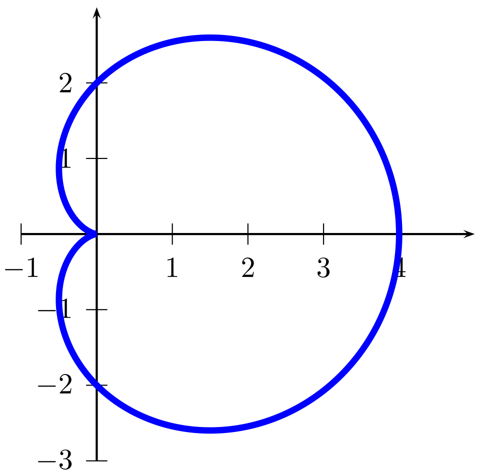

```
\begin{pspicture}(-0.5,-0.5)(7.25,7.25)
 \def\Func{10 x div}
 \psaxes[arrowscale=1.5]{->}(7,7)
 \psplot[linewidth=2pt]{1.5}{5}{\Func}
 \psplotTangent[linewidth=1.5\pslinewidth,
   linecolor=red]{3}{2}{\Func}
 \psplotTangent[linewidth=1.5\pslinewidth,
   linecolor=blue,Tnormal]{3}{2}{\Func}
 \psline[linestyle=dashed](!0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

32.1. A polarplot example

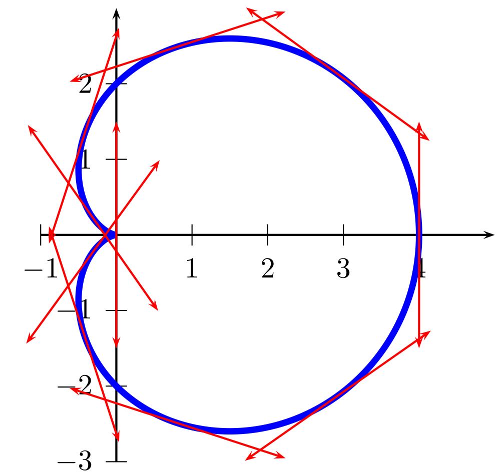
Let's work with the classical cardioid: $r = 2(1+\cos(\theta))$ and $\dfrac{dr}{d\theta} = -2\sin(\theta)$. The `Derive` option always expects the $\frac{dr}{d\theta}$ value and uses internally the equation for the derivative of implicitly defined functions:

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

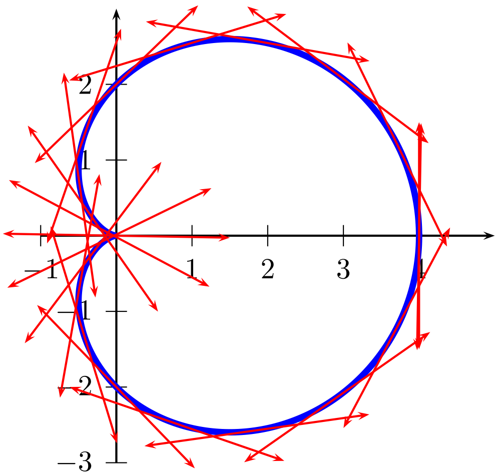
where $x = r \cdot \cos\theta$ and $y = r \cdot \sin\theta$




```
\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}
```

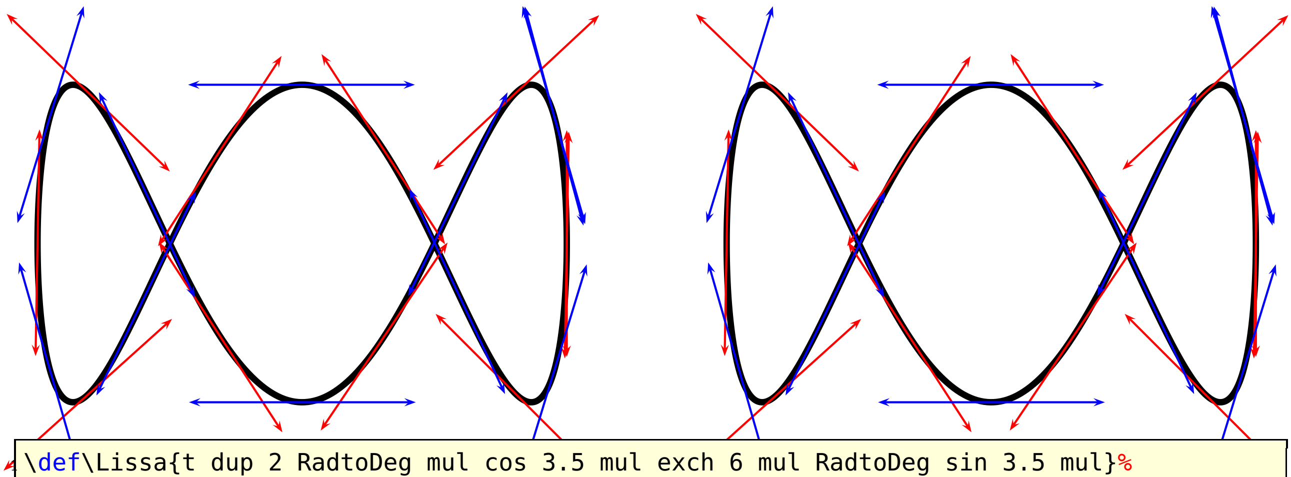



```
\begin{pspicture}(-1,-3)(5,3)%\psgrid[subgridcolor=lightgray]
 \psaxes{->}(0,0)(-1,-3)(5,3)
 \psplot[polarplot,linewidth=3\pslinewidth,linecolor=blue,algebraic,plotpoints=500]{0}{6.289}{2*(1+cos(x))}
 \multido{\r=0.000+0.314}{21}{%
   \psplotTangent[polarplot,Derive=-2*sin(x),algebraic,linecolor=red,arrows=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

32.2. 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 you must separate 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}
```

33. Successive derivatives of a function

The new PostScript function `Derive` has been added for plotting successive derivatives of a function. It must be used with the `algebraic` option. This function has two arguments:

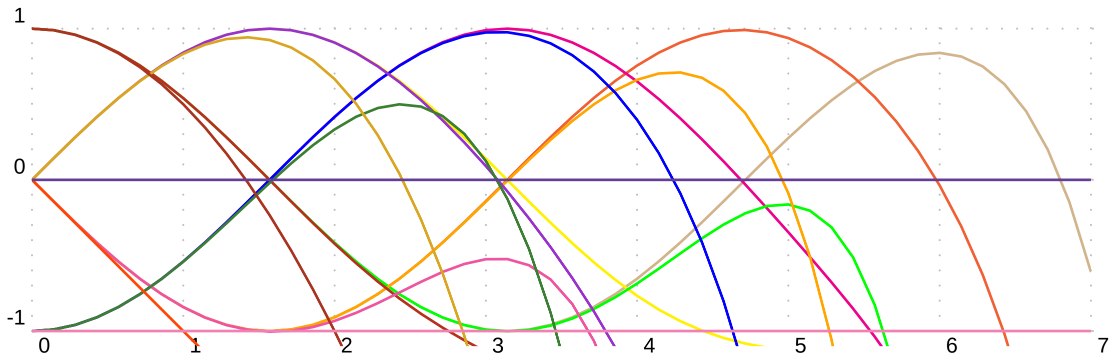
1. a positive integer which defines the order of the derivative; obviously 0 means the function itself!
2. a function of variable x which can be any function using common operators,

Do not think that the derivative is approximated, the internal PostScript engine will compute the real derivative using a formal derivative engine.

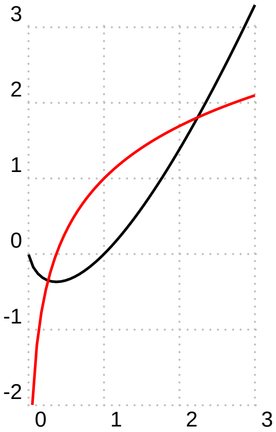
The following diagram contains the plot of the polynomial:

$$f(x)=\sum_{i=0}^{14}\frac{(-1)^i x^{2i}}{i!}=1-\frac{x^2}{2}+\frac{x^4}{4!}-\frac{x^6}{6!}+\frac{x^8}{8!}-\frac{x^{10}}{10!}+\frac{x^{12}}{12!}-\frac{x^{14}}{14!}$$

and of its first 15 derivatives. It is the sequence definition of the cosine.




```
\psset{unit=2}
\def\getColor#1{\ifcase#1 Tan\or RedOrange\or magenta\or yellow\or green\or Orange\or
  blue\or
  DarkOrchid\or BrickRed\or Rhodamine\or OliveGreen\or Goldenrod\or Mahogany\or
  OrangeRed\or CarnationPink\or RoyalPurple\or Lavender\fi}
\begin{pspicture}[showgrid=true](0,-1.2)(7,1.5)
  \psclip{\psframe[linestyle=none](0,-1.1)(7,1.1)}
  \multido{\in=0+1}{16}{%
    \psplot[linewidth=1pt,algebraic,linecolor=\getColor{\in}]{0}{7}
     {Derive(\in,1-x^2/2+x^4/24-x^6/720+x^8/40320-x^10/3628800+x^12/479001600-x
       ^14/87178291200)}}
  \endpsclip
\end{pspicture}
```




```
\begin{pspicture}[shift=-2.5,showgrid=true,linewidth=1pt](0,-2)(3,3)
  \psplot[algebraic]{.001}{3}{x*ln(x)} % f(x)
  \psplot[algebraic,linecolor=red]{.05}{3}{Derive(1,x*ln(x))} % f'(x)=1+ln(x)
\end{pspicture}
```

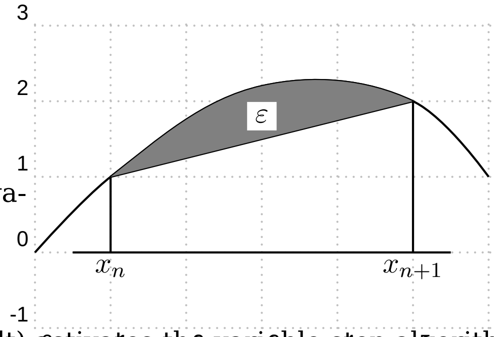
34. Variable step for plotting a curve

34.1. Theory

As you know with the `\psplot` macro, the curve is plotted using a piece-wise linear curve. The step is given by the parameter `plotpoints`. For each step between x_i and x_{i+1}, the area defined between the curve and its approximation (a segment) is majored by this formula :

$$|\varepsilon| \leq \frac{M_2(f)(x_{i+1}-x_i)^3}{12}$$

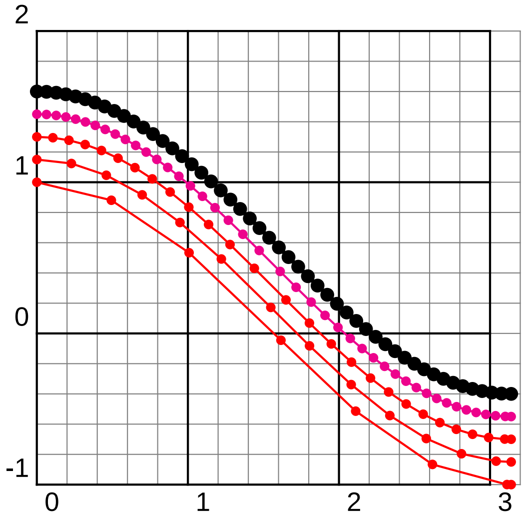
$M_2(f)$ is a majorant of the second derivative of f in the interval $[x_i; x_{i+1}]$.



The parameter `VarStep` (`false` by default) activates the variable step algorithm. It is set to a tolerance defined by the parameter `VarStepEpsilon` (`default` by default, accept real value). If this parameter is not set by the user, then it is automatically computed using the default first step given by the parameter `plotpoints`. Then, for each step, $f''(x_n)$ and $f''(x_{n+1})$ are computed and the smaller is used as $M_2(f)$, and then the step is approximated. This means that the step is constant for second order polynomials.

34.2. The cosine

Different value for the tolerance from 0.01 to $0.000\,1$, a factor 10 between each of them. In black, there is the classic `\psplot` behavior, and in magenta the default variable step behavior.

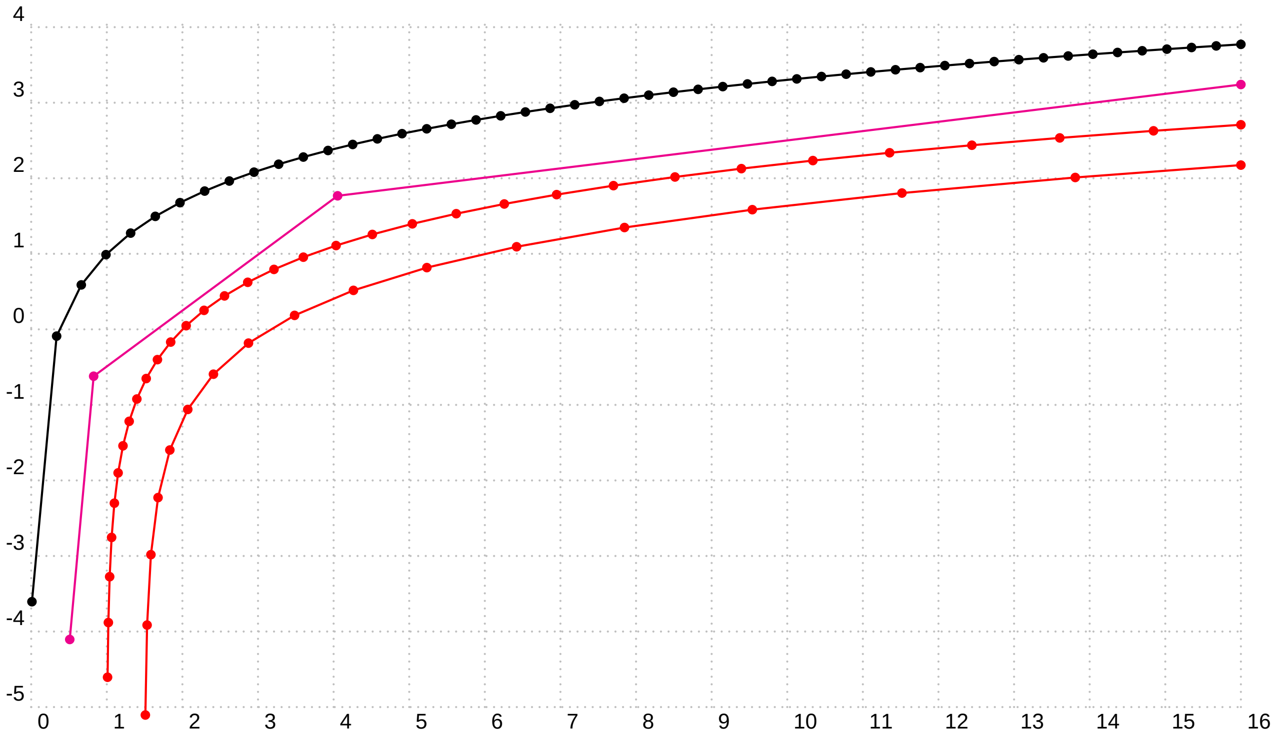



```
\psset{algebraic, VarStep=true, unit=2, showpoints=true, linecolor=red}
\begin{pspicture}[showgrid=true](-0,-1)(3.14,2)
  \psplot[VarStepEpsilon=.01]{0}{3.14}{cos(x)}
  \psplot[VarStepEpsilon=.001]{0}{3.14}{cos(x)+.15}
```

```
  \psplot[VarStepEpsilon=.0001]{0}{3.14}{cos(x)+.3}
  \psplot[linecolor=magenta]{0}{3.14}{cos(x)+.45}
  \psplot[VarStep=false,linewidth=1pt,linecolor=black]{-0}{3.14}{cos(x)+.6}
\end{pspicture}
```

34.3. The Napierian Logarithm

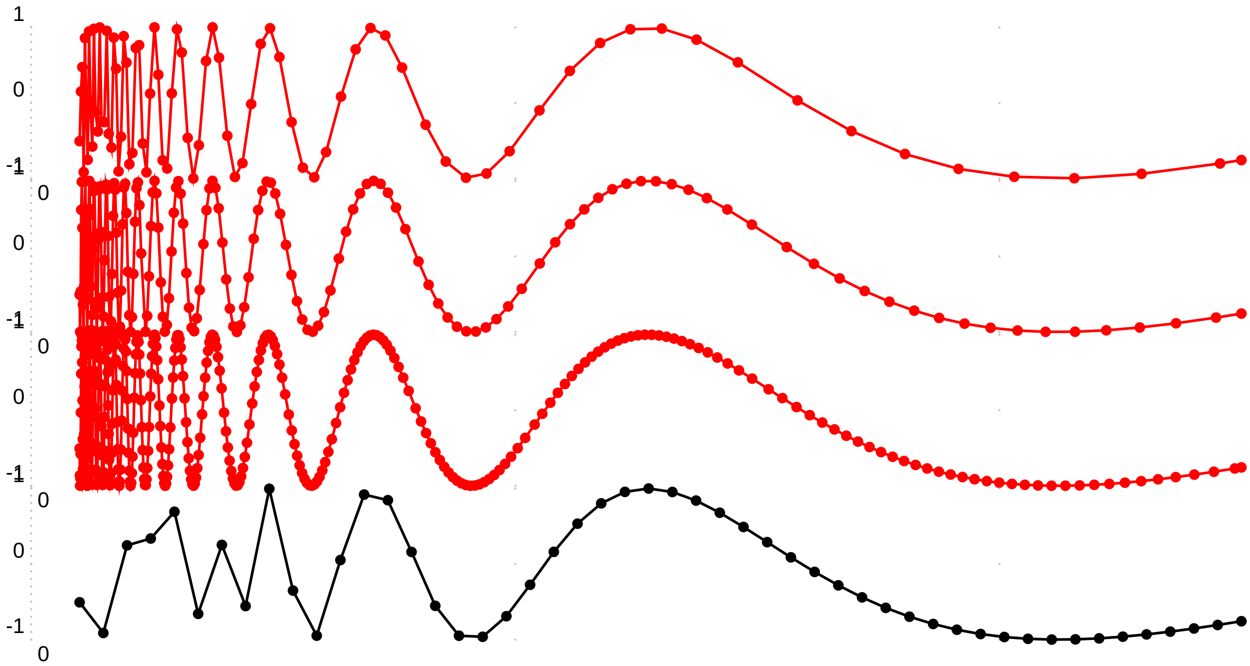
A really classic example which gives a bad beginning, the tolerance is set to 0.001.




```
\psset{algebraic, VarStep=true, linecolor=red, showpoints=true}
\begin{pspicture}[showgrid=true](0,-5)(16,4)
  \psplot[VarStep=false, linecolor=black]{.01}{16}{ln(x)+1}
  \psplot[linecolor=magenta]{.51}{16}{ln(x-1/2)+1/2}
  \psplot[VarStepEpsilon=.001]{1.01}{16}{ln(x-1)}
  \psplot[VarStepEpsilon=.01]{1.51}{16}{ln(x-1.5)-100/200}
\end{pspicture}
```

34.4. Sine of the inverse of x

Impossible to draw, but let's try!

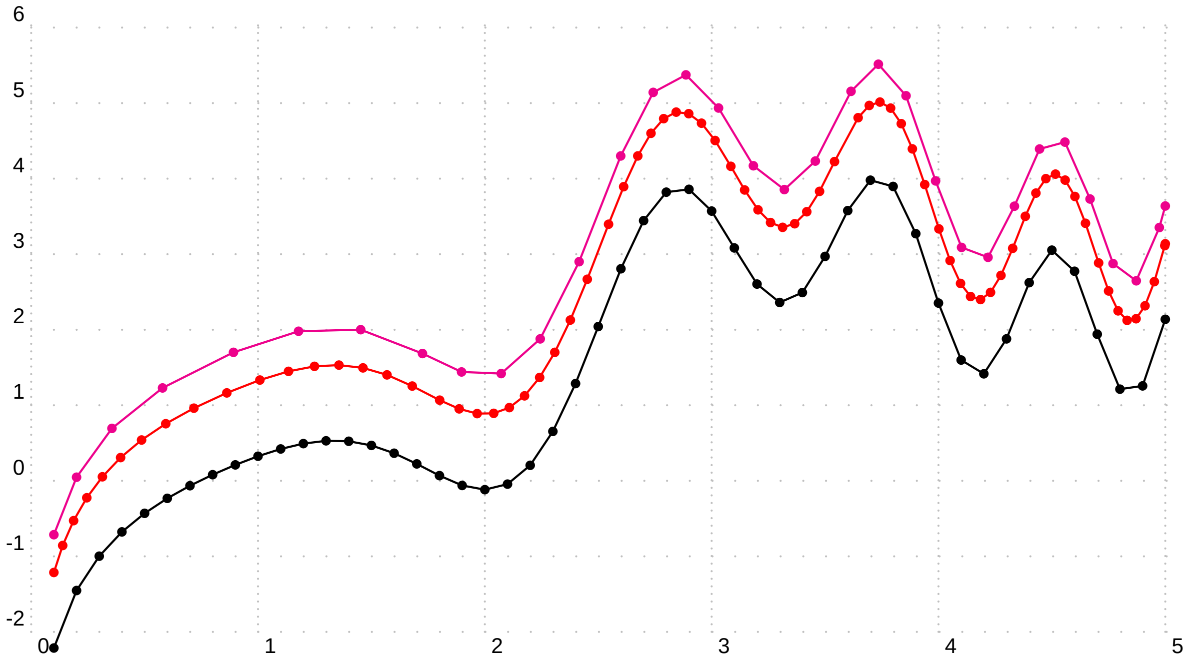


```
\psset{xunit=64,algebraic,VarStep,linecolor=red,showpoints=true,linewidth=1pt}
\begin{pspicture}[showgrid=true](0,-1)(.5,1)
  \psplot[VarStepEpsilon=.0001]{.01}{.25}{sin(1/x)}
\end{pspicture}\\
\begin{pspicture}[showgrid=true](0,-1)(.5,1)
  \psplot[VarStepEpsilon=.00001]{.01}{.25}{sin(1/x)}
\end{pspicture}\\
\begin{pspicture}[showgrid=true](0,-1)(.5,1)
  \psplot[VarStepEpsilon=.000001]{.01}{.25}{sin(1/x)}
\end{pspicture}\\
\begin{pspicture}[showgrid=true](0,-1)(.5,1)
  \psplot[VarStep=false, linecolor=black]{.01}{.25}{sin(1/x)}
\end{pspicture}
```

34.5. A really complecated function

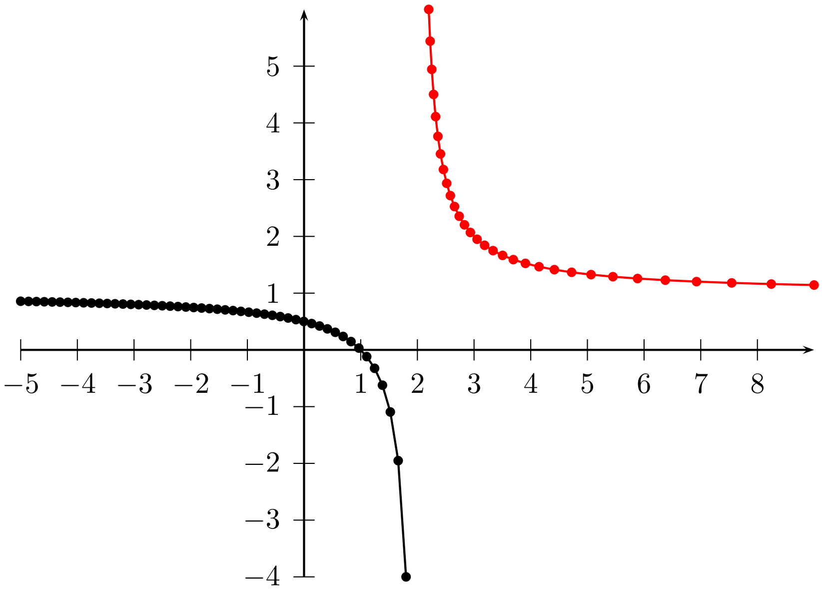
Just appreciate the difference between the normal behavior and the plotting with the `varStep` option. The function is:

$$f(x) = x - \frac{x^2}{10} + \ln(x) + \cos(2x) + \sin(x^2) - 1$$



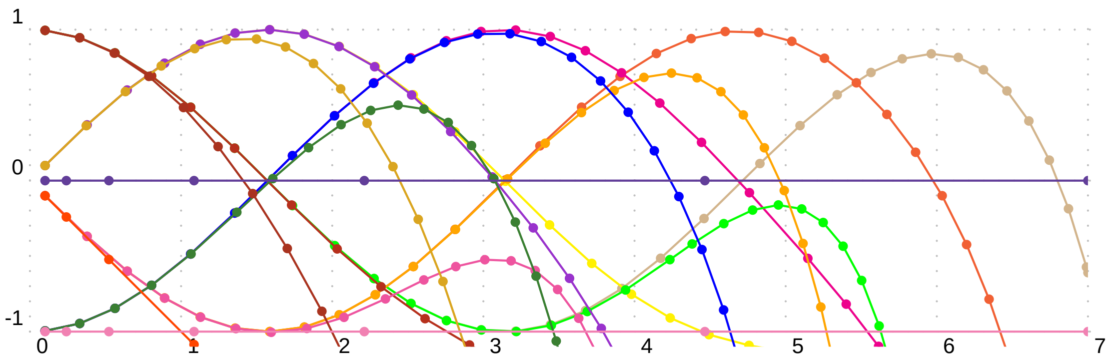

```
\psset{xunit=3, algebraic, VarStep, showpoints=true}
\begin{pspicture}[showgrid=true](0,-2)(5,6)
  \psplot[VarStepEpsilon=.0005, linecolor=red]{.1}{5}{x-x^2/10+ln(x)+cos(2*x
    )+sin(x^2)}
  \psplot[linecolor=magenta]{.1}{5}{x-x^2/10+ln(x)+cos(2*x)+sin(x^2)+.5}
  \psplot[VarStep=false]{.1}{5}{x-x^2/10+ln(x)+cos(2*x)+sin(x^2)-1}
\end{pspicture}
```

34.6. A hyperbola




```
\psset{algebraic, showpoints=true, unit=0.75}
\begin{pspicture}(-5,-4)(9,6)
  \psplot[linecolor=black]{-5}{1.8}{(x-1)/(x-2)}
  \psplot[VarStep=true, VarStepEpsilon=.001, linecolor=red]{2.2}{9}{(x-1)/(x
    -2)}
  \psaxes{->}(0,0)(-5,-4)(9,6)
\end{pspicture}
```

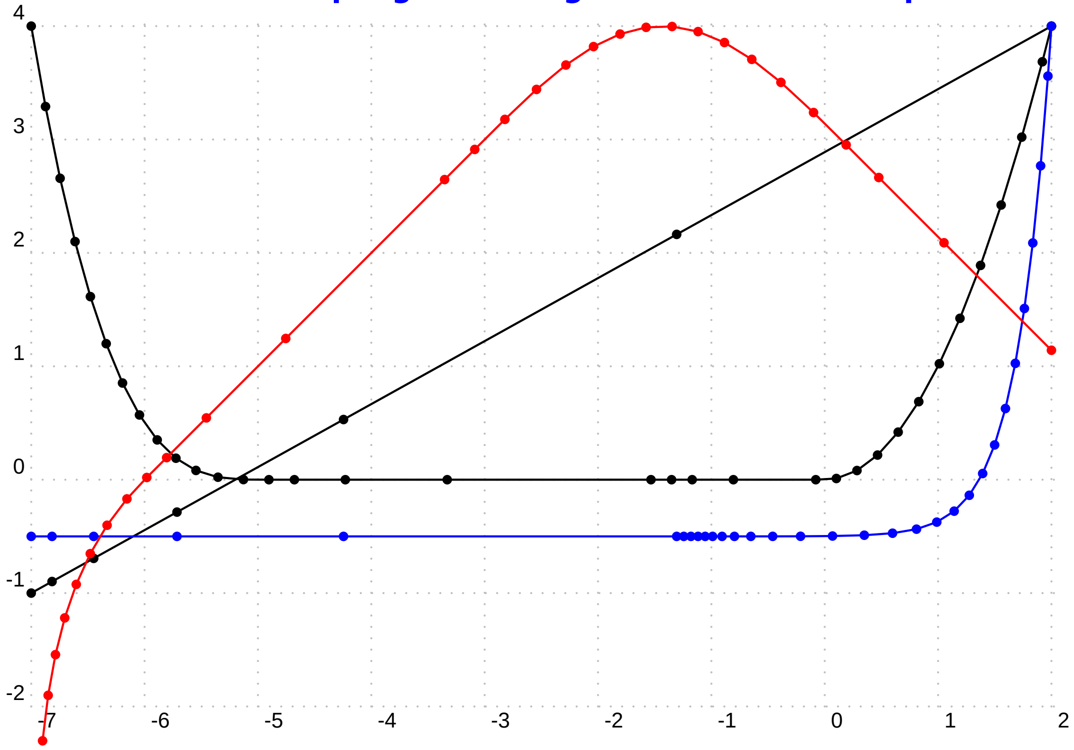
34.7. Successive derivatives of a polynomial




```
\psset{unit=2, algebraic=true, VarStep=true, showpoints=true, VarStepEpsilon
  =.001}
\def\getColor#1{\ifcase#1 Tan\or RedOrange\or magenta\or yellow\or green\or
  Orange\or blue\or
  DarkOrchid\or BrickRed\or Rhodamine\or OliveGreen\or Goldenrod\or Mahogany
    \or
```

```
  OrangeRed\or CarnationPink\or RoyalPurple\or Lavender\fi}
\begin{pspicture}[showgrid=true](0,-1.2)(7,1.5)
  \psclip{\psframe[linestyle=none](0,-1.1)(7,1.1)}
  \multido{\in=0+1}{16}{%
    \psplot[algebraic=true, linecolor=\getColor{\in}]{0.1}{7}
    {Derive(\in,Sum(i,0,1,7,(-1)^i*x^(2*i)/Fact(2*i)))}}
  \endpsclip
\end{pspicture}
```

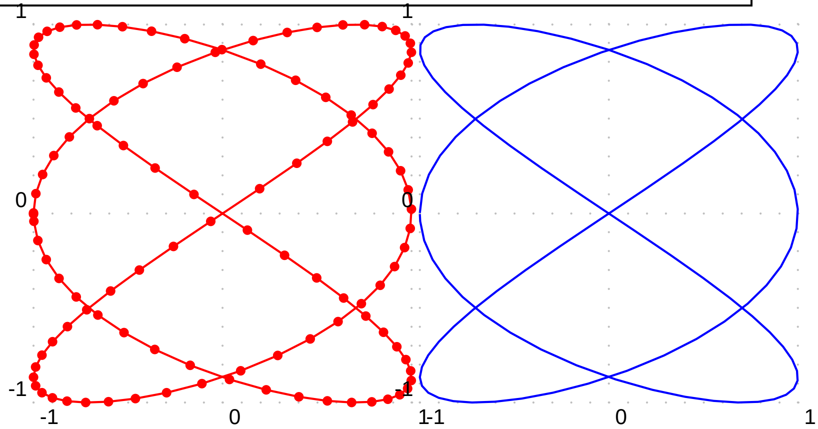
34.8. The variable step algorithm together with the IfTE primitive



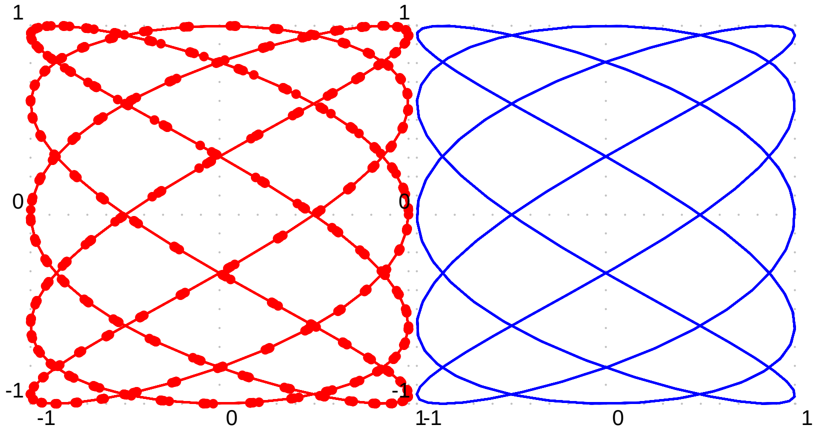

```
\psset{unit=1.5, algebraic, VarStep, showpoints=true, VarStepEpsilon=.001}
\begin{pspicture}[showgrid=true](-7,-2)(2,4)
  \psplot{-7}{2}{IfTE(x<-5,-(x+5)^3/2,IfTE(x<0,0,x^2))}
  \psplot{-7}{2}{5*x/9+26/9}
  \psplot[linecolor=blue]{-7}{2}{(x+7)^30/9^30*4.5-1/2}
  \psplot[linecolor=red]{-6.9}{2}
     {IfTE(x<-6,ln(x+7),IfTE(x<-3,x+6,IfTE(x<0.1415926,sin(x+3)+3,3.1415926-
       x)))}
\end{pspicture}
```

34.9. Using `\parametricplot`

`\parametricplot [Options] {t0}{t1} [PS commands] {x(t) y(t)}`

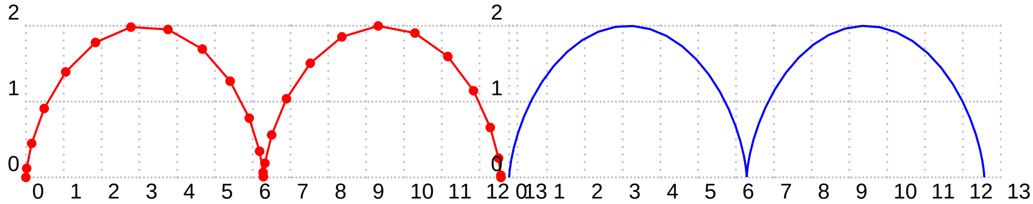



```
\psset{unit=3}
\begin{pspicture}[showgrid=true](-1,-1)(1,1)
\parametricplot[algebraic=true,linecolor=red,VarStep=true, showpoints=true,
            VarStepEpsilon=.0001]
            {-3.14}{3.14}{cos(3*t)|sin(2*t)}
\end{pspicture}
\begin{pspicture}[showgrid=true](-1,-1)(1,1)
\parametricplot[algebraic=true,linecolor=blue,VarStep=true, showpoints=false
  ,
            VarStepEpsilon=.0001]
            {-3.14}{3.14}{cos(3*t)|sin(2*t)}
\end{pspicture}
```




```
\psset{unit=2.5}
\begin{pspicture}[showgrid=true](-1,-1)(1,1)
\parametricplot[algebraic=true,linecolor=red,VarStep=true, showpoints=true,
            VarStepEpsilon=.0001]
            {0}{47.115}{cos(5*t)|sin(3*t)}
\end{pspicture}
\begin{pspicture}[showgrid=true](-1,-1)(1,1)
```

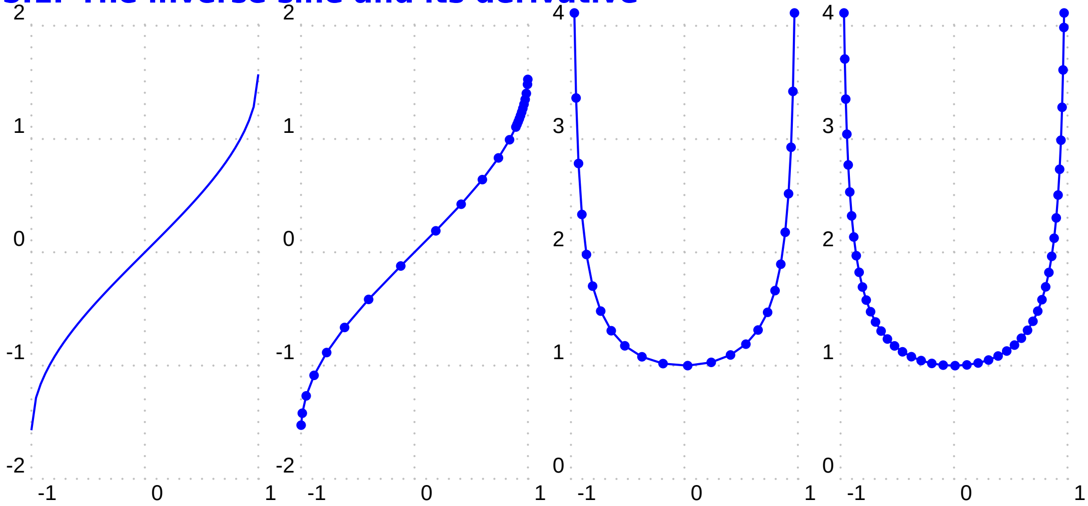
```
\parametricplot[algebraic=true,linecolor=blue,VarStep=true, showpoints=false
  ,
            VarStepEpsilon=.0001]
            {0}{47.115}{cos(5*t)|sin(3*t)}
\end{pspicture}
```




```
\psset{xunit=.5}
\begin{pspicture}[showgrid=true](0,0)(12.566,2)
\parametricplot[algebraic,linecolor=red,VarStep, showpoints=true,
      VarStepEpsilon=.01]{0}{12.566}{t+cos(-t-Pi/2)|1+sin(-t-Pi/2)}
\end{pspicture}
%
\begin{pspicture}[showgrid=true](0,0)(12.566,2)
\parametricplot[algebraic,linecolor=blue,VarStep, showpoints=false,
      VarStepEpsilon=.001]{0}{12.566}{t+cos(-t-Pi/2)|1+sin(-t-Pi/2)}
\end{pspicture}
```

35. New math functions and their derivatives

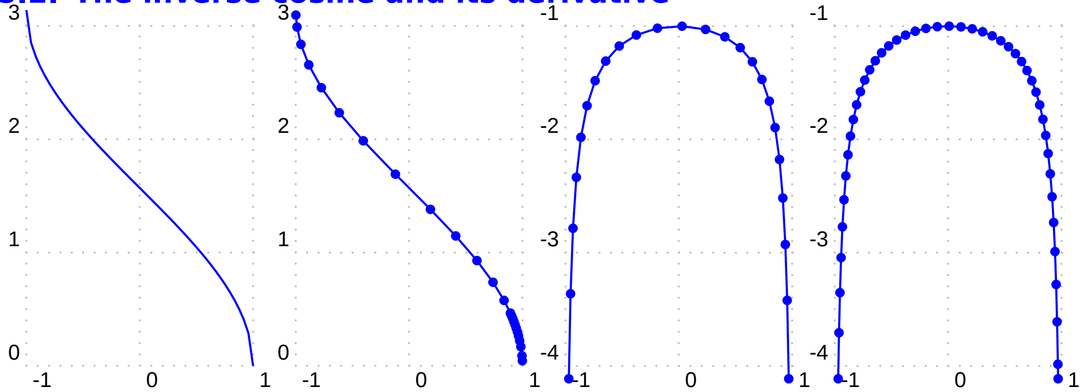
35.1. The inverse sine and its derivative




```
\psset{unit=1.5}
\begin{pspicture}[showgrid=true](-1,-2)(1,2)
  \psplot[linecolor=blue,algebraic]{-1}{1}{asin(x)}
\end{pspicture}
\hspace{1em}
\psset{algebraic, VarStep, VarStepEpsilon=.001, showpoints=true}
\begin{pspicture}[showgrid=true](-1,-2)(1,2)
```

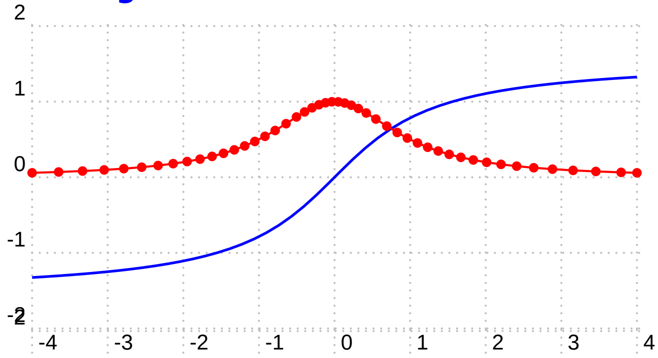
```
  \psplot[linecolor=blue]{-.999}{.999}{asin(x)}
\end{pspicture}
\hspace{1em}
\begin{pspicture}[showgrid=true](-1,0)(1,4)
  \psplot[linecolor=red]{-.97}{.97}{Derive(1,asin(x))}
\end{pspicture}
\hspace{1em}
\psset{algebraic, VarStep, VarStepEpsilon=.0001, showpoints=true}
\begin{pspicture}[showgrid=true](-1,0)(1,4)
  \psplot[linecolor=red]{-.97}{.97}{Derive(1,asin(x))}
\end{pspicture}
```

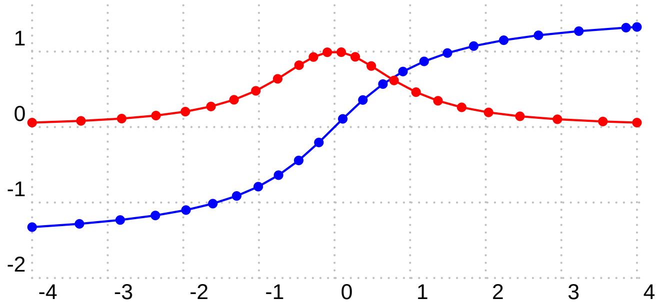
35.2. The inverse cosine and its derivative



```
\psset{unit=1.5}
\begin{pspicture}[showgrid=true](-1,0)(1,3)
  \psplot[linecolor=blue,algebraic]{-1}{1}{acos(x)}
\end{pspicture}
\hspace{1em}
\psset{algebraic, VarStep, VarStepEpsilon=.001, showpoints=true}
\begin{pspicture}[showgrid=true](-1,0)(1,3)
  \psplot[linecolor=blue]{-.999}{.999}{acos(x)}
\end{pspicture}
\hspace{1em}
\begin{pspicture}[showgrid=true](-1,-4)(1,-1)
  \psplot[linecolor=red]{-.97}{.97}{Derive(1,acos(x))}
\end{pspicture}
\hspace{1em}
\psset{algebraic, VarStep, VarStepEpsilon=.0001, showpoints=true}
\begin{pspicture}[showgrid=true](-1,-4)(1,-1)
  \psplot[linecolor=red]{-.97}{.97}{Derive(1,acos(x))}
\end{pspicture}
```

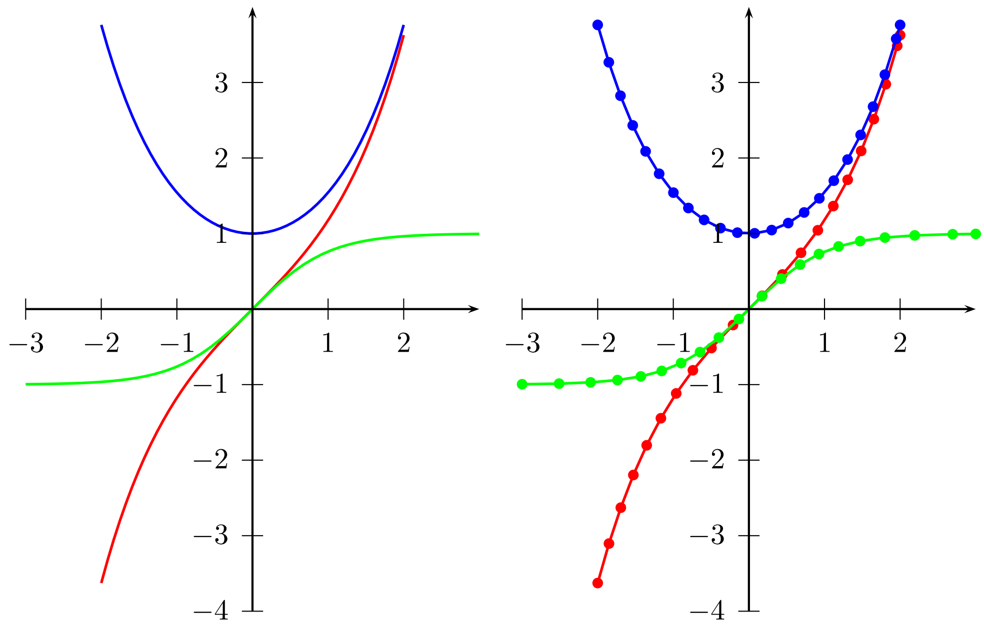
35.3. The inverse tangent and its derivative



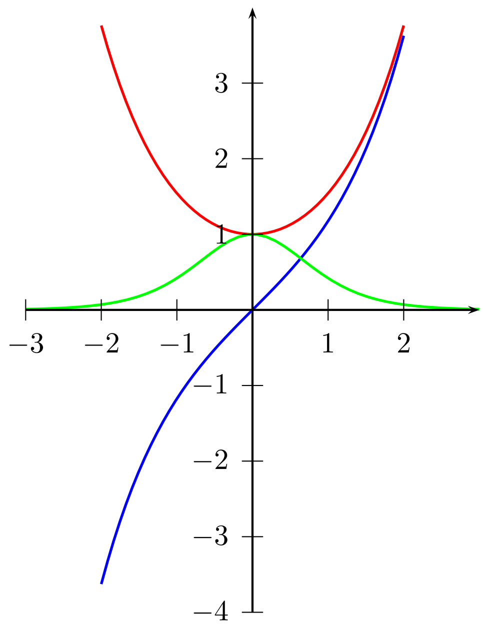


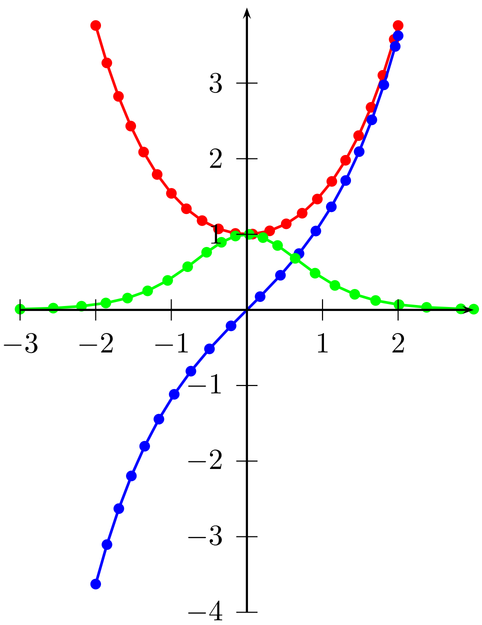

```
\begin{pspicture}[showgrid=true](-4,-2)(4,2)
\psset{algebraic=true}
  \psplot[linecolor=blue,linewidth=1pt]{-4}{4}{atg(x)}
  \psplot[linecolor=red,VarStep, VarStepEpsilon=.0001, showpoints=true
    ]{-4}{4}{Derive(1,atg(x))}
\end{pspicture}
\hspace{1em}
\begin{pspicture}[showgrid=true](-4,-2)(4,2)
\psset{algebraic, VarStep, VarStepEpsilon=.001, showpoints=true}
  \psplot[linecolor=blue]{-4}{4}{atg(x)}
  \psplot[linecolor=red]{-4}{4}{Derive(1,atg(x))}
\end{pspicture}
```

35.4. Hyperbolic functions

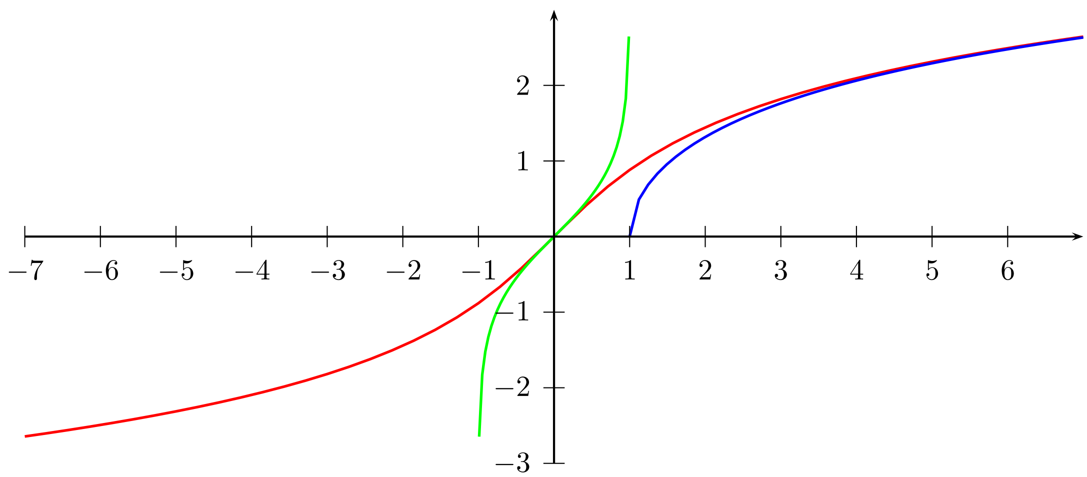


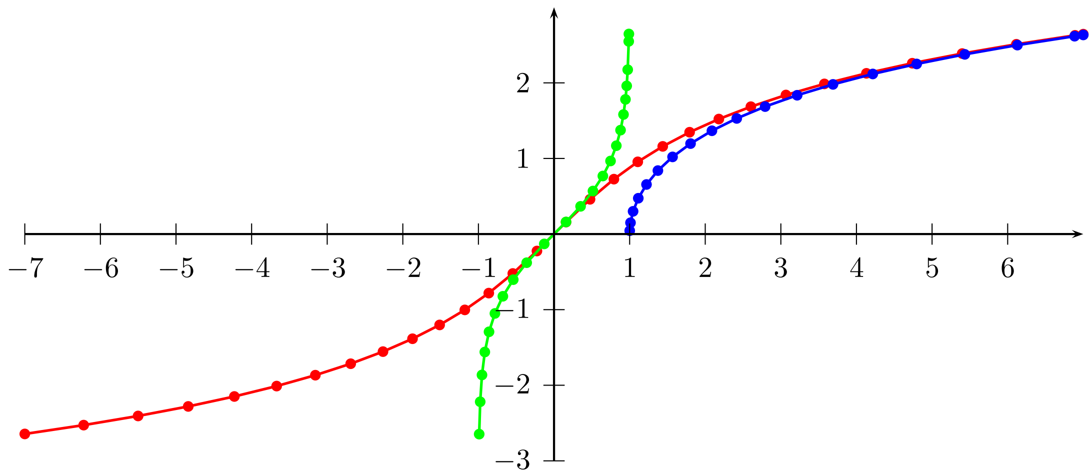

```
\begin{pspicture}(-3,-4)(3,4)
\psset{algebraic=true}
  \psplot[linecolor=red,linewidth=1pt]{-2}{2}{sh(x)}
  \psplot[linecolor=blue,linewidth=1pt]{-2}{2}{ch(x)}
  \psplot[linecolor=green,linewidth=1pt]{-3}{3}{th(x)}
  \psaxes{->}(0,0)(-3,-4)(3,4)
\end{pspicture}
\hspace{1em}
\begin{pspicture}(-3,-4)(3,4)
\psset{algebraic=true, VarStep=true, VarStepEpsilon=.001, showpoints=true}
  \psplot[linecolor=red,linewidth=1pt]{-2}{2}{sh(x)}
  \psplot[linecolor=blue,linewidth=1pt]{-2}{2}{ch(x)}
  \psplot[linecolor=green,linewidth=1pt]{-3}{3}{th(x)}
  \psaxes{->}(0,0)(-3,-4)(3,4)
\end{pspicture}
```



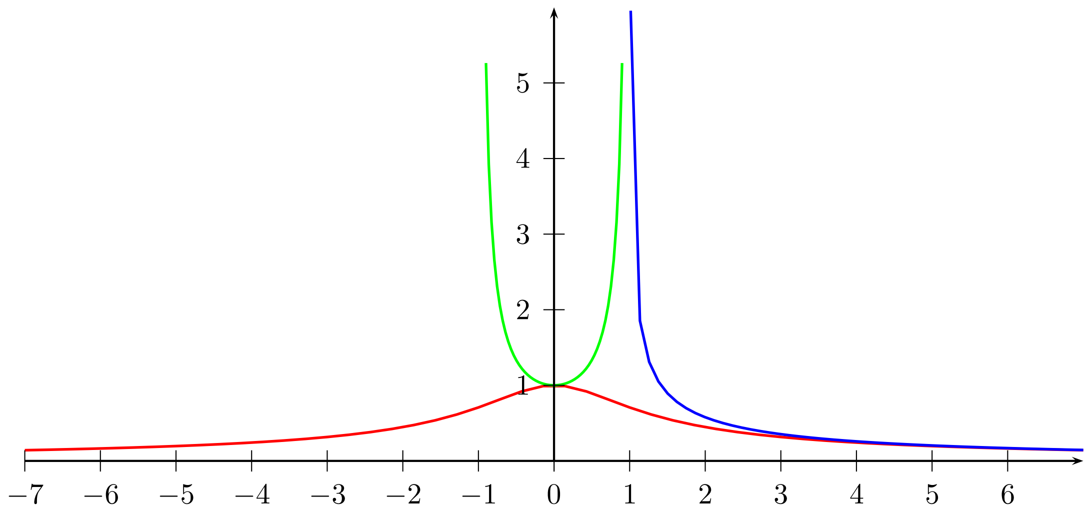


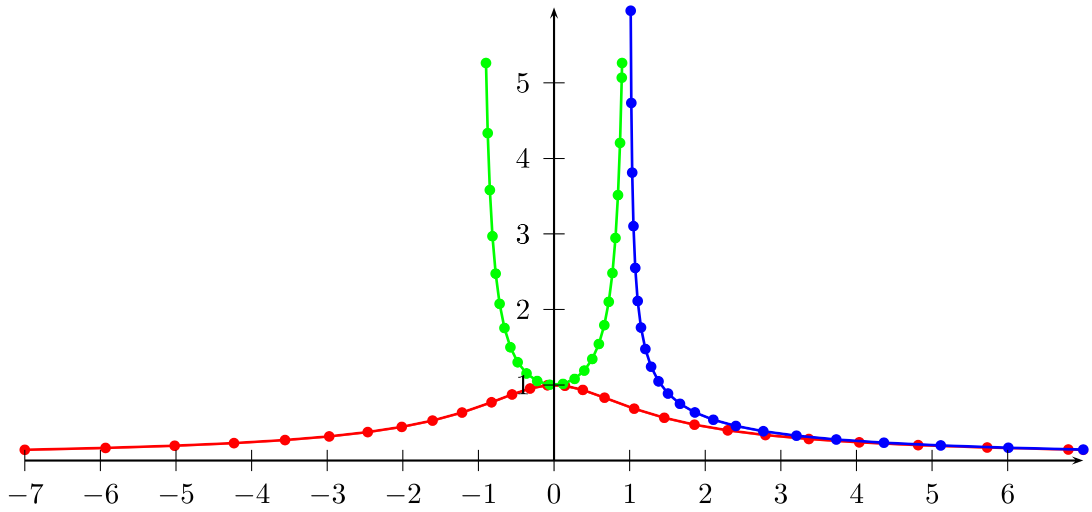

```
\begin{pspicture}(-3,-4)(3,4)
\psset{algebraic=true,linewidth=1pt}
  \psplot[linecolor=red,linewidth=1pt]{-2}{2}{Derive(1,sh(x))}
  \psplot[linecolor=blue,linewidth=1pt]{-2}{2}{Derive(1,ch(x))}
  \psplot[linecolor=green,linewidth=1pt]{-3}{3}{Derive(1,th(x))}
  \psaxes{->}(0,0)(-3,-4)(3,4)
\end{pspicture}
\hspace{1em}
\begin{pspicture}(-3,-4)(3,4)
\psset{algebraic=true, VarStep=true, VarStepEpsilon=.001, showpoints=true}
  \psplot[linecolor=red,linewidth=1pt]{-2}{2}{Derive(1,sh(x))}
  \psplot[linecolor=blue,linewidth=1pt]{-2}{2}{Derive(1,ch(x))}
  \psplot[linecolor=green,linewidth=1pt]{-3}{3}{Derive(1,th(x))}
  \psaxes{->}(0,0)(-3,-4)(3,4)
\end{pspicture}
```






```
\begin{pspicture}(-7,-3)(7,3)
\psset{algebraic=true}
  \psplot[linecolor=red,linewidth=1pt]{-7}{7}{Argsh(x)}
  \psplot[linecolor=blue,linewidth=1pt]{1}{7}{Argch(x)}
  \psplot[linecolor=green,linewidth=1pt]{-.99}{.99}{Argth(x)}
  \psaxes{->}(0,0)(-7,-3)(7,3)
\end{pspicture}\\[\baselineskip]
\begin{pspicture}(-7,-3)(7,3)
  \psset{algebraic, VarStep, VarStepEpsilon=.001, showpoints=true}
  \psplot[linecolor=red,linewidth=1pt]{-7}{7}{Argsh(x)}
  \psplot[linecolor=blue,linewidth=1pt]{1.001}{7}{Argch(x)}
  \psplot[linecolor=green,linewidth=1pt]{-.99}{.99}{Argth(x)}
  \psaxes{->}(0,0)(-7,-3)(7,3)
\end{pspicture}
```






```
\begin{pspicture}(-7,-0.5)(7,6)
\psset{algebraic=true}
  \psplot[linecolor=red,linewidth=1pt]{-7}{7}{Derive(1,Argsh(x))}
  \psplot[linecolor=blue,linewidth=1pt]{1.014}{7}{Derive(1,Argch(x))}
  \psplot[linecolor=green,linewidth=1pt]{-.9}{.9}{Derive(1,Argth(x))}
  \psaxes{->}(0,0)(-7,0)(7,6)
\end{pspicture}\\[\baselineskip]
\begin{pspicture}(-7,-0.5)(7,6)
\psset{algebraic=true}
  \psset{algebraic=true, VarStep=true, VarStepEpsilon=.001, showpoints=true}
  \psplot[linecolor=red,linewidth=1pt]{-7}{7}{Derive(1,Argsh(x))}
  \psplot[linecolor=blue,linewidth=1pt]{1.014}{7}{Derive(1,Argch(x))}
  \psplot[linecolor=green,linewidth=1pt]{-.9}{.9}{Derive(1,Argth(x))}
  \psaxes{->}(0,0)(-7,0)(7,6)
\end{pspicture}
```

36. \psplotDiffEqn – solving diffential equations

A differential equation of first order is like

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

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

`\psplotDiffEqn [Options] {x0}{x1}{y0}{f(x,y,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 4^{th} 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 the ordinate;
- `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;
- `algebraic`: algebraic description for f, `buildvector` parameter is useless when activating this option.

36.1. Variable step for differential equations

A new algorithm has been added for adjusting the step according to the variations of the curve. The parameter `method` has a new possible value : `varrkiv` to activate the Runge-Kutta method with variable step, then the parameter `varsteptol` (real value; `.01` by default) can control the tolerance of the algortihm.

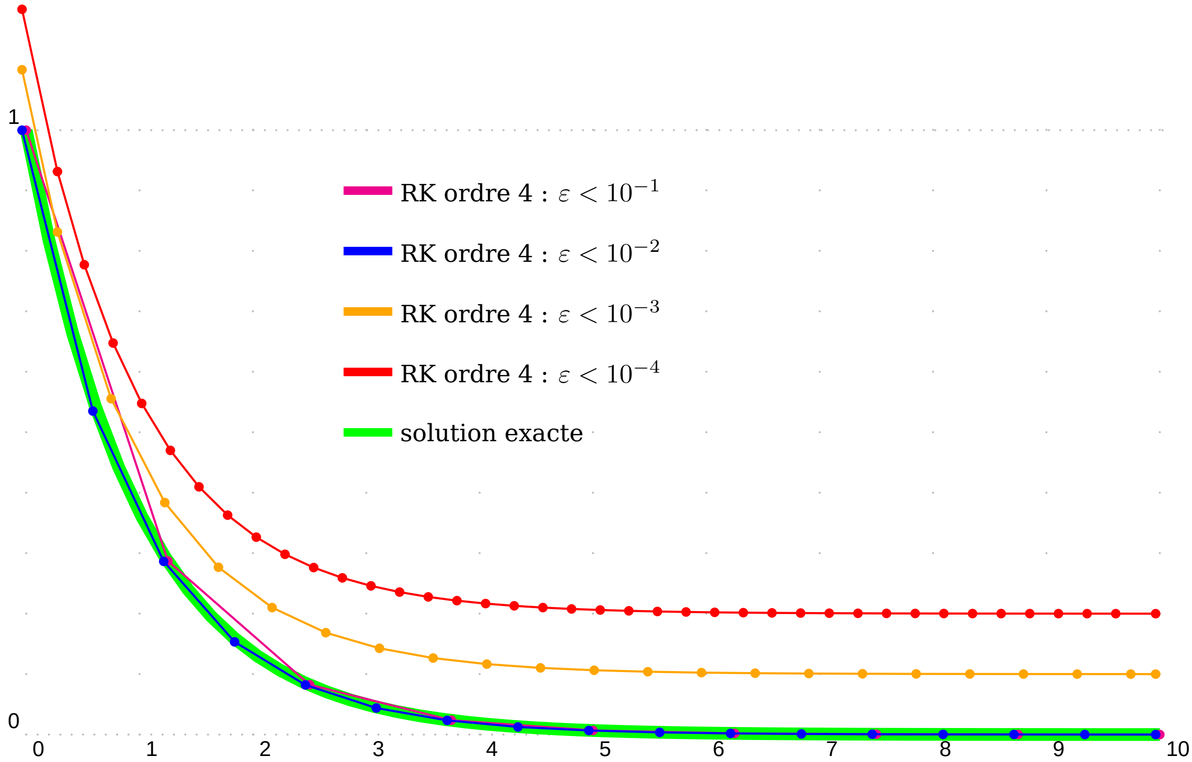


Figure 1: Equation $y' = -y$ with $y_0 = 1$.

```
\def\Funct{neg}\def\FunctAlg{-y[0]}
\psset{xunit=1.5, yunit=8, showpoints=true}
\begin{pspicture}[showgrid=true](0,0)(10,1.2)
  \psplot[linewidth=6\pslinewidth, linecolor=green, showpoints=false
    ]{0}{10}{Euler x neg exp}
  \psplotDiffEqn[linecolor=magenta, method=varrkiv, varsteptol=.1,
    plotpoints=2]{0}{10}{1}{\Funct}
  \rput(0,.0){\psplotDiffEqn[linecolor=blue, method=varrkiv, varsteptol=.01,
     plotpoints=2]{0}{10}{1}{\Funct}}
  \rput(0,.1){\psplotDiffEqn[linecolor=Orange, method=varrkiv, varsteptol
    =.001, plotpoints=2]{0}{10}{1}{\Funct}}
  \rput(0,.2){\psplotDiffEqn[linecolor=red, method=varrkiv, varsteptol
    =.0001, plotpoints=2]{0}{10}{1}{\Funct}}
  \psset{linewidth=4\pslinewidth,showpoints=false}
  \rput*(3.3,.9){\psline[linecolor=magenta](-.75cm,0)}
  \rput*[l](3.3,.9){\small RK ordre 4 : $\varepsilon<10^{-1}$}
  \rput*(3.3,.8){\psline[linecolor=blue](-.75cm,0)}
  \rput*[l](3.3,.8){\small RK ordre 4 : $\varepsilon<10^{-2}$}
  \rput*(3.3,.7){\psline[linecolor=Orange](-.75cm,0)}
```

```
  \rput*[l](3.3,.7){\small RK ordre 4 : $\varepsilon<10^{-3}$}
  \rput*(3.3,.6){\psline[linecolor=red](-.75cm,0)}
  \rput*[l](3.3,.6){\small RK ordre 4 : $\varepsilon<10^{-4}$}
  \rput*(3.3,.5){\psline[linecolor=green](-.75cm,0)}
  \rput*[l](3.3,.5){\small solution exacte}
\end{pspicture}
```

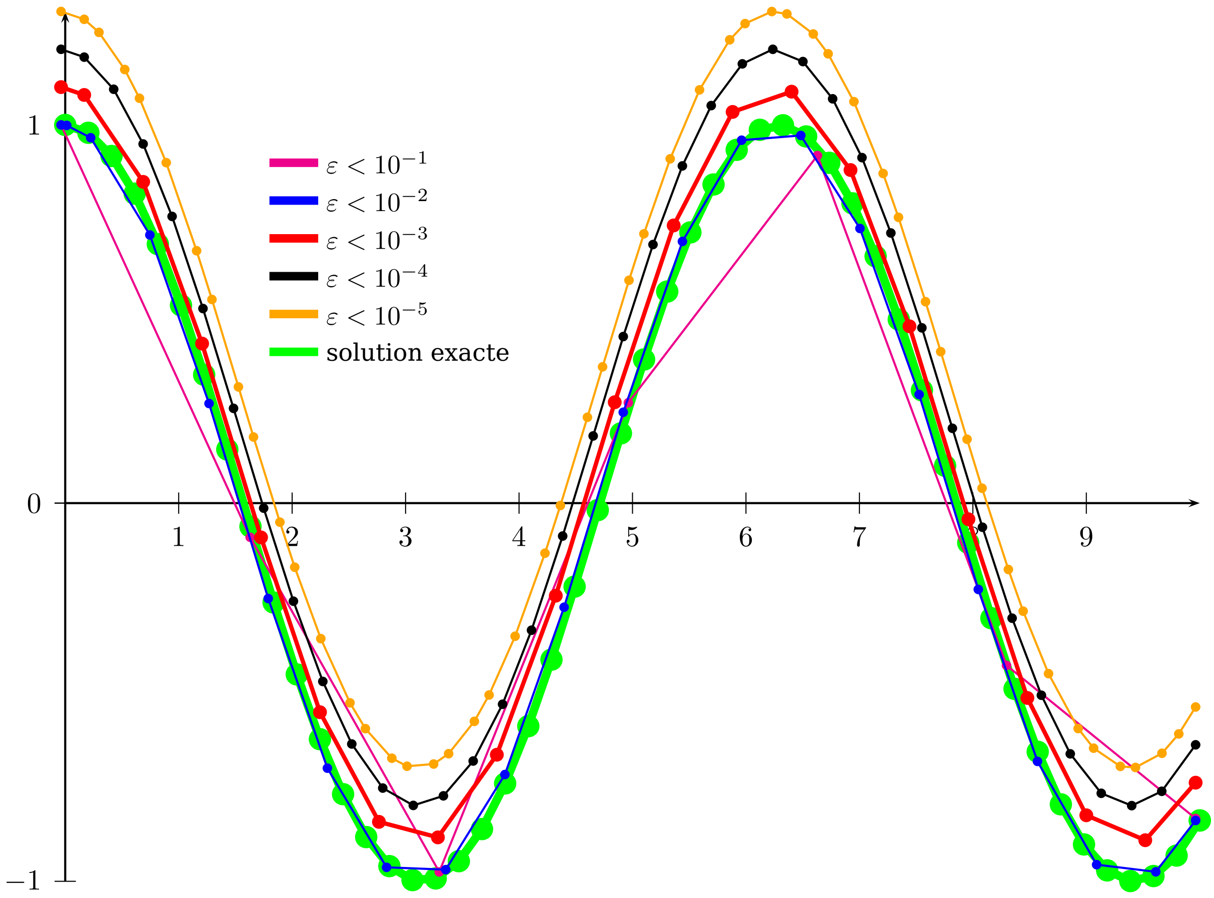


Figure 2: Equation $y'' = -y$

```
\def\Funct{exch neg}
\psset{xunit=1.5, yunit=5, method=varrkiv, showpoints=true}%%
\def\quatrepi{12.5663706144}
\begin{pspicture}(0,-1)(10,1.3)
  \psaxes{->}(0,0)(0,-1)(10,1.3)
  \psplot[linewidth=4\pslinewidth, linecolor=green, algebraic=true]{0}{10}{
    cos(x)}
  \rput(0,.0){\psplotDiffEqn[linecolor=magenta, plotpoints=7, varsteptol
    =.1]{0}{10}{1 0}{\Funct}}
  \rput(0,.0){\psplotDiffEqn[linecolor=blue, plotpoints=201, varsteptol
    =.01]{0}{10}{1 0}{\Funct}}
  \rput(0,.1){\psplotDiffEqn[linewidth=2\pslinewidth, linecolor=red,
    varsteptol=.001]{0}{10}{1 0}{\Funct}}
```

```
  \rput(0,.2){\psplotDiffEqn[linecolor=black, varsteptol=.0001]{0}{10}{10}{\Funct}}
  \rput(0,.3){\psplotDiffEqn[linecolor=Orange, varsteptol=.00001]{0}{10}{10}{\Funct}}
  \psset{linewidth=4\pslinewidth,showpoints=false}
  \rput*(2.3,.9){\psline[linecolor=magenta](-.75cm,0)}
  \rput*[l](2.3,.9){\small $\varepsilon<10^{-1}$}
  \rput*(2.3,.8){\psline[linecolor=blue](-.75cm,0)}
  \rput*[l](2.3,.8){\small $\varepsilon<10^{-2}$}
  \rput*(2.3,.7){\psline[linecolor=red](-.75cm,0)}
  \rput*[l](2.3,.7){\small $\varepsilon<10^{-3}$}
  \rput*(2.3,.6){\psline[linecolor=black](-.75cm,0)}
  \rput*[l](2.3,.6){\small $\varepsilon<10^{-4}$}
  \rput*(2.3,.5){\psline[linecolor=Orange](-.75cm,0)}
  \rput*[l](2.3,.5){\small $\varepsilon<10^{-5}$}
  \rput*(2.3,.4){\psline[linecolor=green](-.75cm,0)}
  \rput*[l](2.3,.4){\small solution exacte}
\end{pspicture}
```

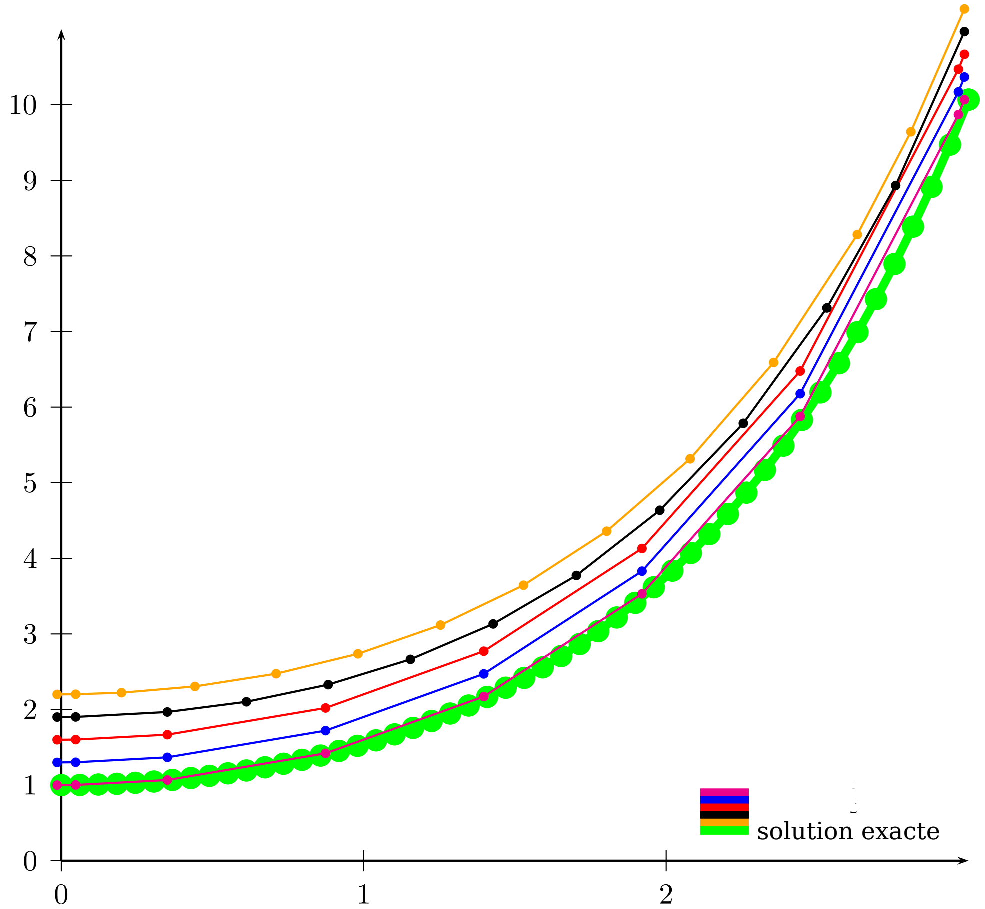


Figure 3: Equation $y'' = y$

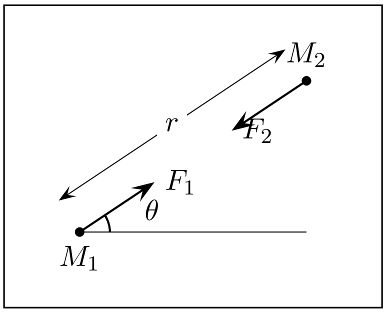
```
\def\Funct{exch}
\psset{xunit=4, yunit=1, method=varrkiv, showpoints=true}%%
```

```
\def\quatrepi{12.5663706144}
\begin{pspicture}(0,-0.5)(3,11)
  \psaxes{->}(0,0)(3,11)
  \psplot[linewidth=4\pslinewidth, linecolor=green, algebraic=true]{0}{3}{ch
    (x)}
  \rput(0,.0){\psplotDiffEqn[linecolor=magenta, varsteptol=.1]{0}{3}{1 0}{\
    Funct}}
  \rput(0,.3){\psplotDiffEqn[linecolor=blue, varsteptol=.01]{0}{3}{1 0}{\
    Funct}}
  \rput(0,.6){\psplotDiffEqn[linecolor=red, varsteptol=.001]{0}{3}{1 0}{\
    Funct}}
  \rput(0,.9){\psplotDiffEqn[linecolor=black, varsteptol=.0001]{0}{3}{1 0}{\
    Funct}}
  \rput(0,1.2){\psplotDiffEqn[linecolor=Orange, varsteptol=.00001]{0}{3}{1
    0}{\Funct}}
  \psset{linewidth=4\pslinewidth,showpoints=false}
  \rput*(2.3,.9){\psline[linecolor=magenta](-.75cm,0)}
  \rput*[l](2.3,.9){\small $\varepsilon<10^{-1}$}
  \rput*(2.3,.8){\psline[linecolor=blue](-.75cm,0)}
  \rput*[l](2.3,.8){\small $\varepsilon<10^{-2}$}
  \rput*(2.3,.7){\psline[linecolor=red](-.75cm,0)}
  \rput*[l](2.3,.7){\small $\varepsilon<10^{-3}$}
  \rput*(2.3,.6){\psline[linecolor=black](-.75cm,0)}
  \rput*[l](2.3,.6){\small $\varepsilon<10^{-4}$}
  \rput*(2.3,.5){\psline[linecolor=Orange](-.75cm,0)}
  \rput*[l](2.3,.5){\small $\varepsilon<10^{-5}$}
  \rput*(2.3,.4){\psline[linecolor=green](-.75cm,0)}
  \rput*[l](2.3,.4){\small solution exacte}
\end{pspicture}
```

36.2. Equation of second order

Here is the traditional simulation of two stars attracting each other according to the classical gravitation law in $\frac{1}{r^2}$. In 2-Dimensions, the system to be solved is composed of four second order differential equations. In order to be described, each of them gives two first order equations, then we obtain a 8 sized vectorial equation. In the following example the masses of the stars are 1 and 20.

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

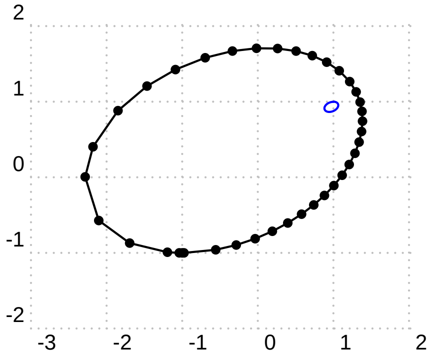



```
                                                        %% x1 y1 x'1 y'1 x2 y2 x'2 y'2
/yp2 exch def /xp2 exch def /ay2 exch def /ax2 exch def %% mise en variables
/yp1 exch def /xp1 exch def /ay1 exch def /ax1 exch def %% mise en variables
/ro2 ax2 ax1 sub dup mul ay2 ay1 sub dup mul add def    %% 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
```

Table 3: `PostScript` source code for the gravitational interaction

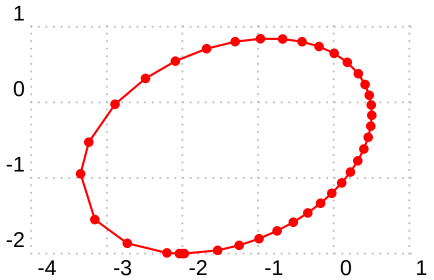
```
y[2]|                                         %% y'[0]
y[3]|                                         %% y'[1]
(y[4]-y[0])/((y[4]-y[0])^2+(y[5]-y[1])^2)^1.5|    %% y'[2]=y"[0]
(y[5]-y[1])/((y[4]-y[0])^2+(y[5]-y[1])^2)^1.5|    %% y'[3]=y"[1]
y[6]|                                         %% y'[4]
y[7]|                                         %% y'[5]
20*(y[0]-y[4])/((y[4]-y[0])^2+(y[5]-y[1])^2)^1.5| %% y'[6]=y"[4]
20*(y[1]-y[5])/((y[4]-y[0])^2+(y[5]-y[1])^2)^1.5  %% y'[7]=y"[5]
```

Table 4: Algebraic description for the gravitational interaction




```
\def\InitCond{ 1 1 .1 0 -1 -1 -2 0}
\begin{pspicture}[shift=-2,showgrid=true](-3,-1.75)(2,1.5)
  \psplotDiffEqn[whichabs=0, whichord=1, linecolor=blue,
    method=rk4, plotpoints=100]{0}{3.95}{\InitCond}{\Grav}
  \psset{showpoints=true,whichabs=4, whichord=5}
  \psplotDiffEqn[linecolor=black, method=varrkiv,
    varsteptol=.0001, plotpoints=200]{0}{3.9}{\InitCond}{\
    Grav}
\end{pspicture}
```

Figure 4: Gravitational interaction: fixed landmark, trajectory of the stars




```
\def\InitCond{ 1 1 .1 0 -1 -1 -2 0}
\begin{pspicture}[shift=-1.5,showgrid=true](-4,-1.75)(1,1)
  \psplotDiffEqn[linecolor=red, plotpoints=200,method=
    varrkiv, varsteptol=.0001, showpoints=true,
     plotfuncx=y dup 4 get exch 0 get sub,
     plotfuncy=dup 5 get exch 1 get sub ]{0}{3.9}{\InitCond
       }{\Grav}
\end{pspicture}
```

Figure 5: Gravitational interaction : landmark defined by one star

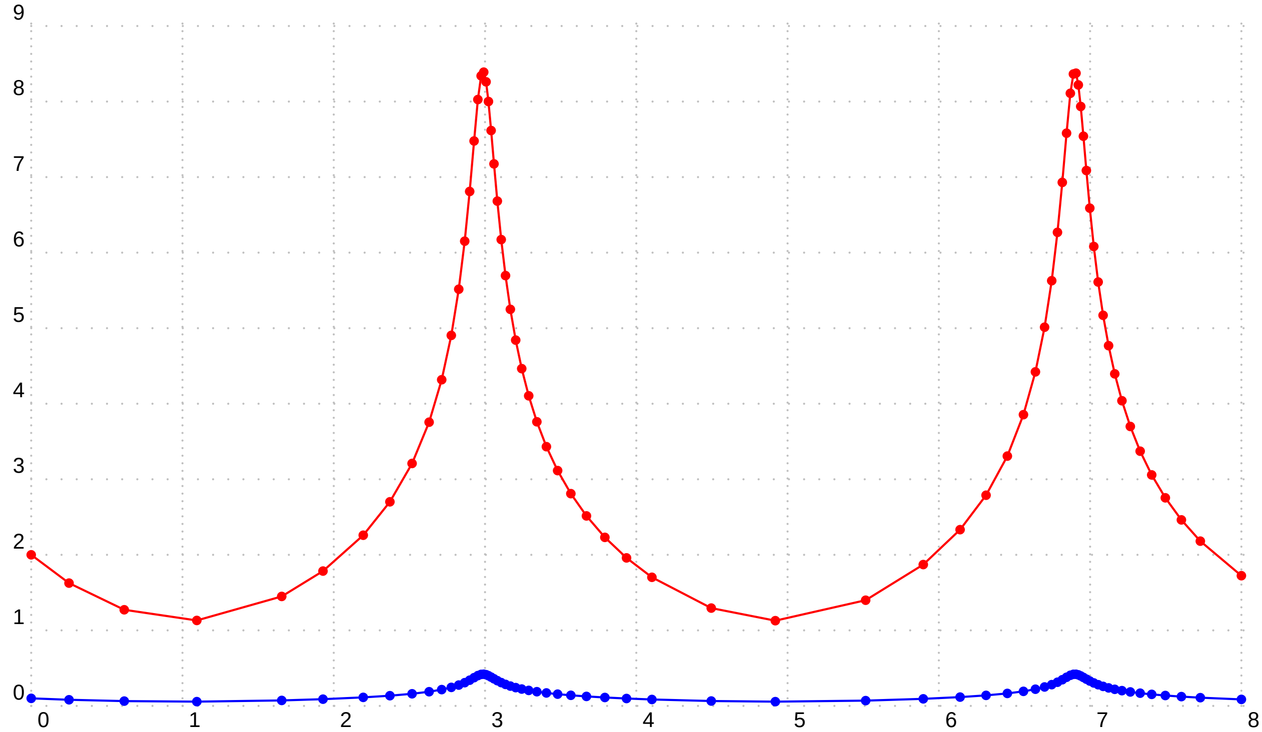
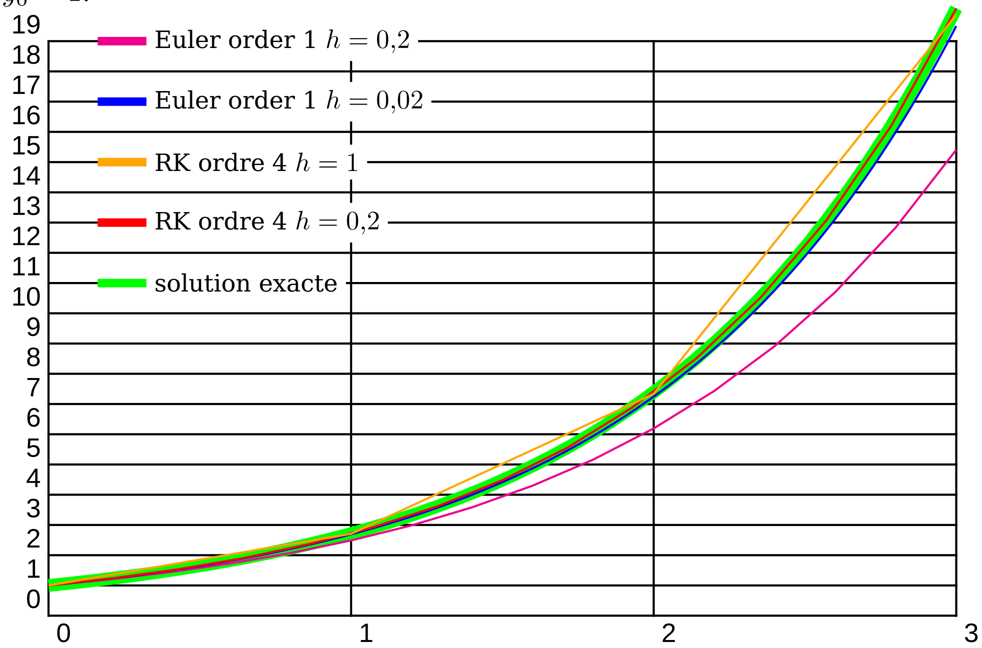


Figure 6: Gravitational interaction : speeds of the stars

```
\psset{xunit=2}
\begin{pspicture}[showgrid=true](0,0)(8,9)
 \psset{showpoints=true}
 \psplotDiffEqn[linecolor=red, method=varrkiv, plotpoints=2, varsteptol=.0001,
    plotfuncy=dup 6 get dup mul exch 7 get dup mul add sqrt]{0}{8}{\InitCond}{\Grav}
 \psplotDiffEqn[linecolor=blue, method=varrkiv, plotpoints=2, varsteptol=.0001,
    plotfuncy=dup 2 get dup mul exch 3 get dup mul add sqrt]{0}{8}{\InitCond}{\Grav}
\end{pspicture}
```

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}
```

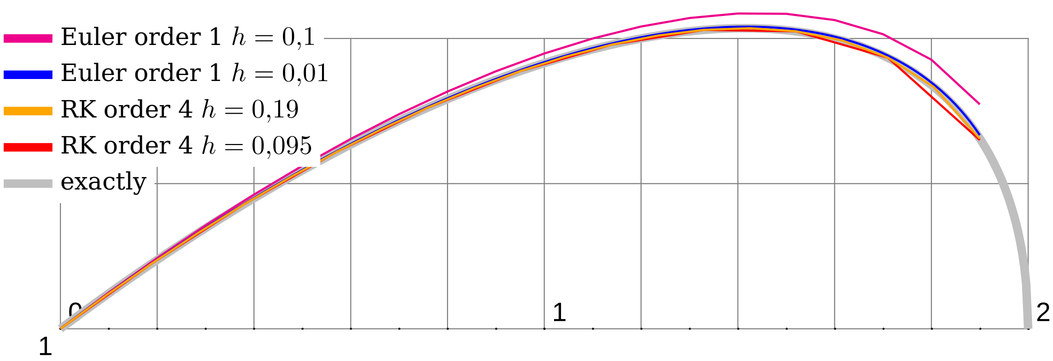
$$y' = \frac{2-ty}{4-t^2}$$

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 is 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)}
```

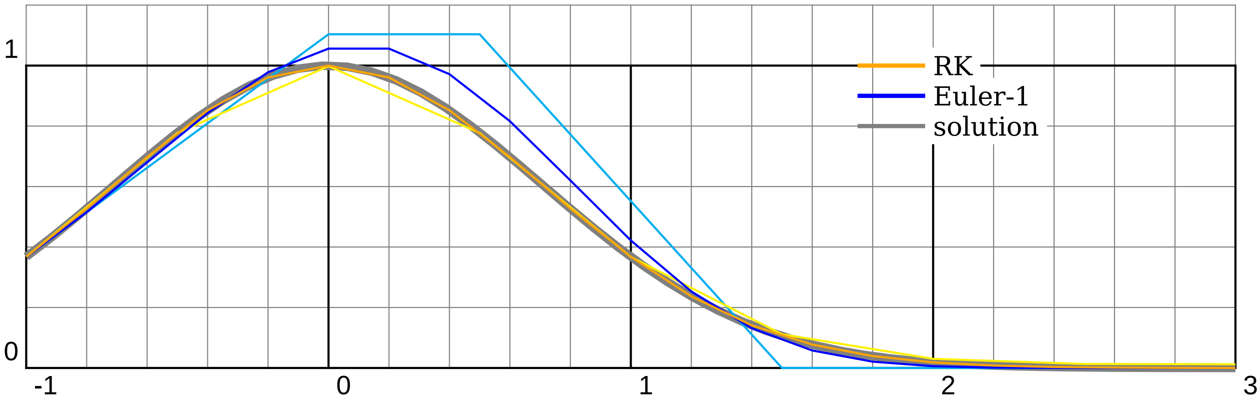



```
\psset{xunit=6.4, yunit=9.6, showpoints=false}
\begin{pspicture}(0,1)(2,1.7) \psgrid[subgriddiv=5]
 { \psset{linewidth=4\pslinewidth,linecolor=lightgray}
 \psplot{0}{1.8}{x dup dup mul 4 exch sub sqrt add 2 div}
 \psplot{1.8}{2}{x dup dup mul 4 exch sub sqrt add 2 div} }
 \def\InitCond{1}
 \def\Func{x 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}
```

$y' = -2xy$

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}
```

Spiral of Cornu

The integrals of Fresnel:

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

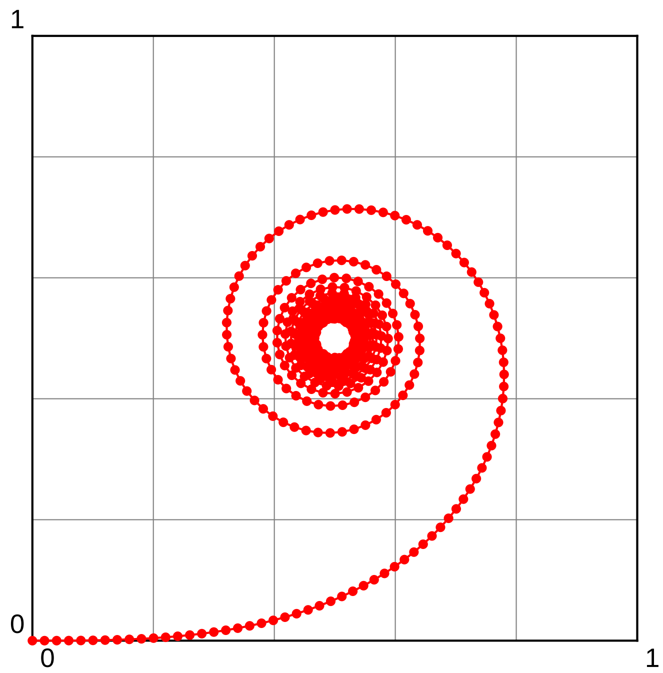
$$y = \int_0^t \sin \frac{\pi t^2}{2} \mathrm{d}t \tag{3}$$

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}
```



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:

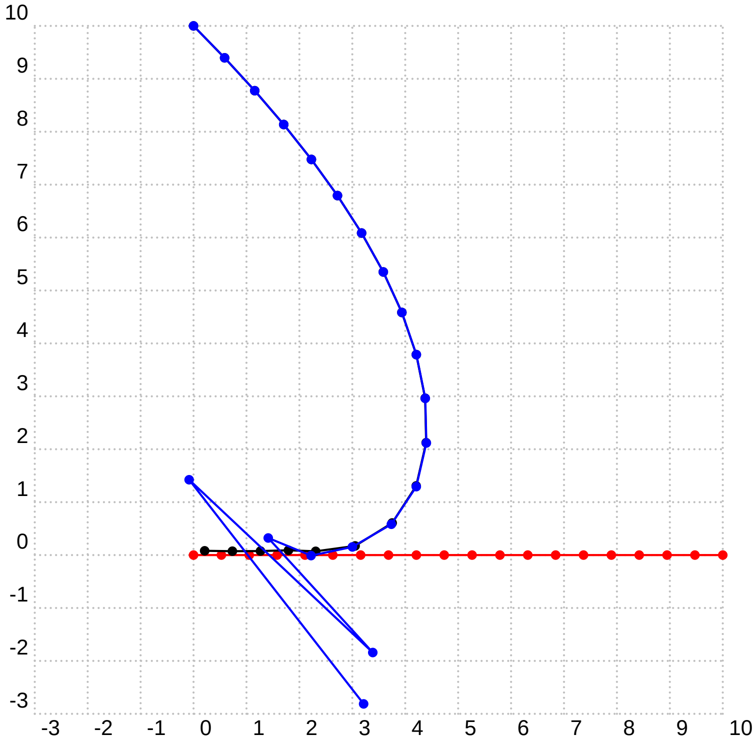
a is the natural growth rate of rabbits in the absence of predation,

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

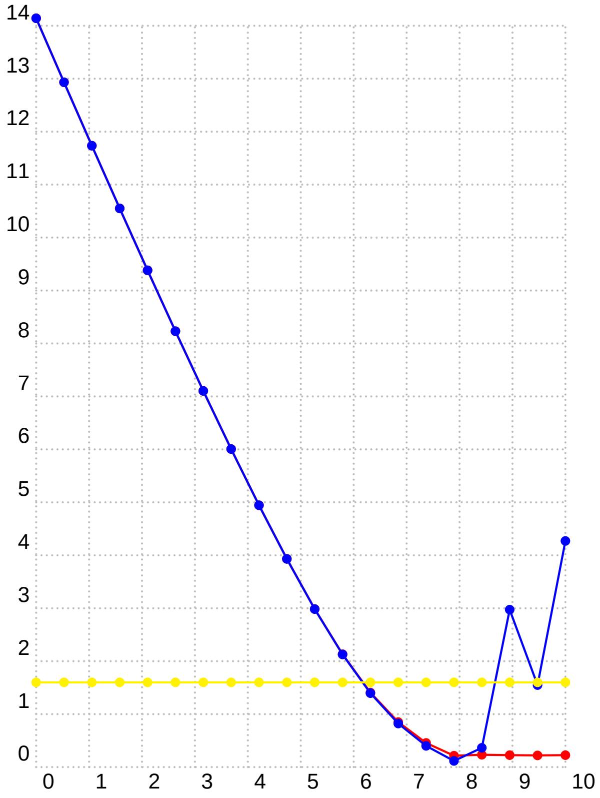
b is the death rate per encounter of rabbits due to predation,

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}[showgrid=true](-3,-3)(10,10)
 \psplotDiffEqn[plotfuncy=pop 0,whichabs=2,linecolor=red]{0}{10}{\InitCond}{\
   Faiglelapin}
 \psplotDiffEqn[whichabs=0,whichord=1,linecolor=black,method=rk4]{0}{10}{\InitCond}{\
   Faiglelapin}
  \psplotDiffEqn[whichabs=0,whichord=1,linecolor=blue]{0}{10}{\InitCond}{\Faiglelapin
   }
\end{pspicture}
```

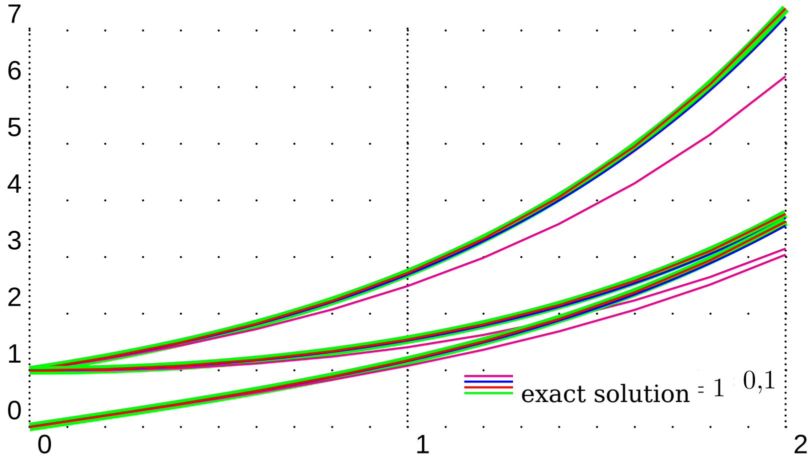



```
\def\InitCond{ 0 10 10}%% xa ya xl
\def\Faiglelapin{\Vaigle*(y[2]-y[0])/sqrt(y[1]^2+(y[2]-y[0])^2)|%
             -\Vaigle*y[1]/sqrt(y[1]^2+(y[2]-y[0])^2)|%
             -\Vlapin}
\def\Vlapin{1} \def\Vaigle{1.6}
\psset{unit=.7,subgriddiv=0,gridcolor=lightgray,method=adams,algebraic,%
  plotpoints=20,showpoints=true}
\begin{pspicture}[showgrid=true](10,12)
 \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}
```

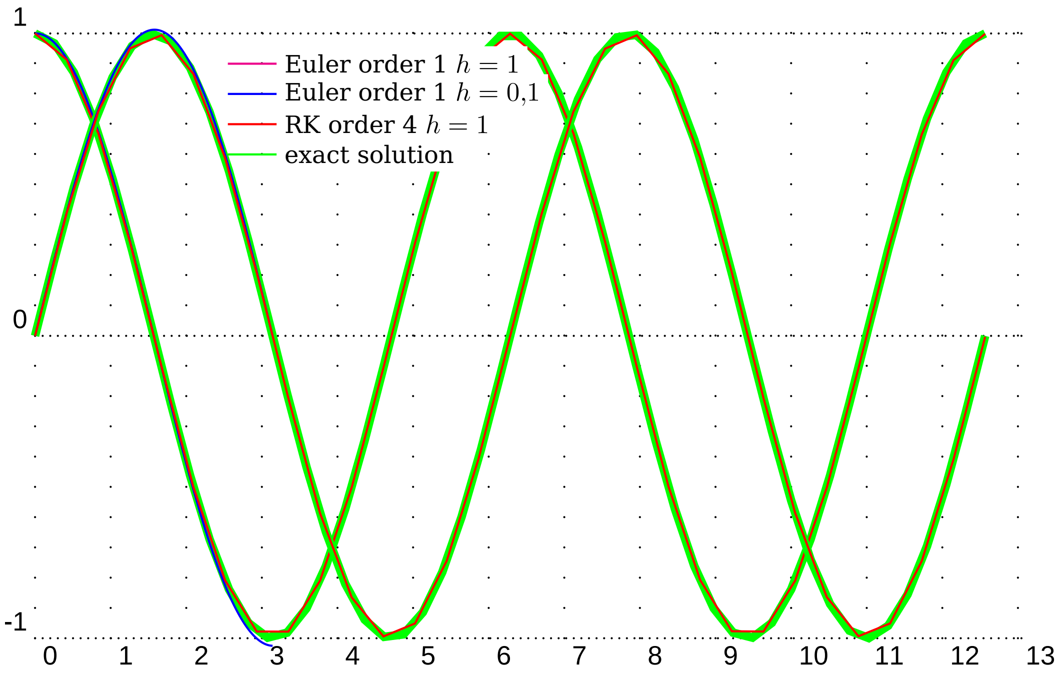
$y'' = y$

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




```
\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} order 1 $h=1$}
 \rput*(1.3,.8){\psline[linecolor=blue](-.75cm,0)}\rput*[l](1.3,.8){\small\textsc{
  Euler} order 1 $h=0{,}1$}
 \rput*(1.3,.7){\psline[linecolor=red](-.75cm,0)}\rput*[l](1.3,.7){\small RK order 4
  $h=1$}
 \rput*(1.3,.6){\psline[linecolor=green](-.75cm,0)}\rput*[l](1.3,.6){\small exact
  solution}
\end{pspicture}
```

$y'' = -y$




```
\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 order 4 $h=1$}
 \rput*(3.3,.6){\psline[linecolor=green](-.75cm,0)}\rput*[l](3.3,.6){\small exact solution}
\end{pspicture}
```

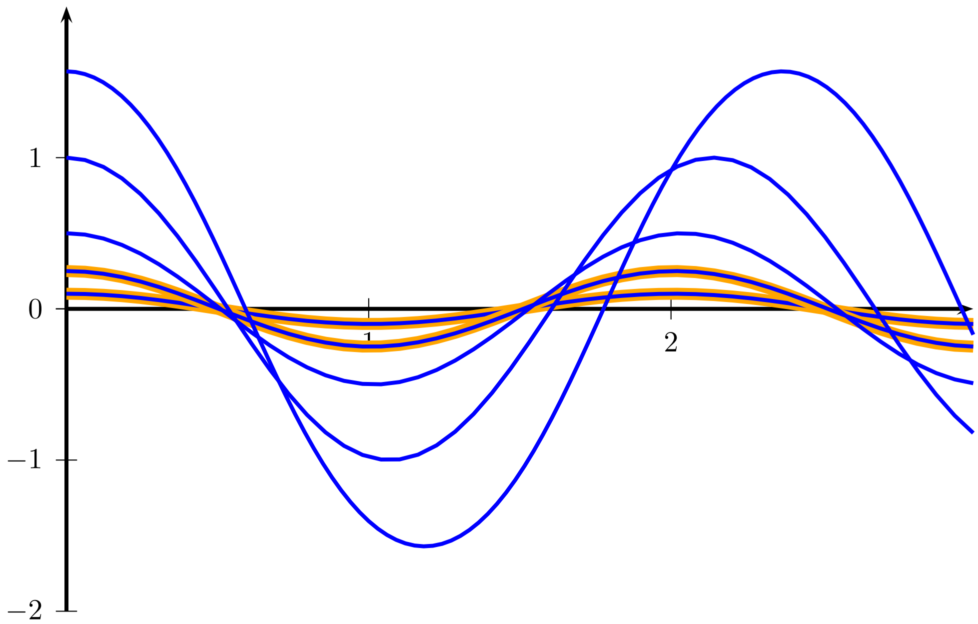
The mechanical pendulum: $y'' = -\frac{g}{l}\sin(y)$

For small oscillations $\sin(y) \simeq y$:

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

The function f is written in PostScript code:

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

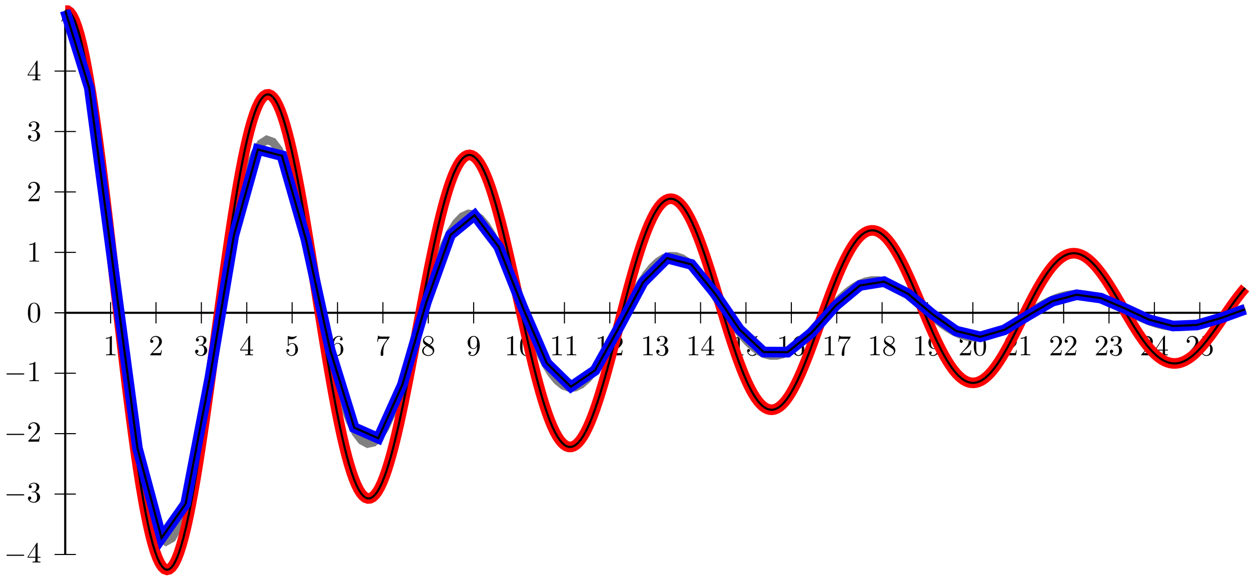



```
\def\Func{y[1]|-9.8*sin(y[0])}
\psset{yunit=2,xunit=4,algebraic=true,linewidth=1.5pt}
\begin{pspicture}(0,-2.25)(3,2.25)
  \psaxes{->}(0,0)(0,-2)(3,2)
  \psplot[linewidth=3\pslinewidth, linecolor=Orange]{0}{3}{.1*cos(sqrt(9.8)*x)}
  \psset{method=rk4,plotpoints=50,linecolor=blue}
  \psplotDiffEqn{0}{3}{.1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

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

For $y_0 = 5$ and $y'_0 = 0$ the solution is:

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

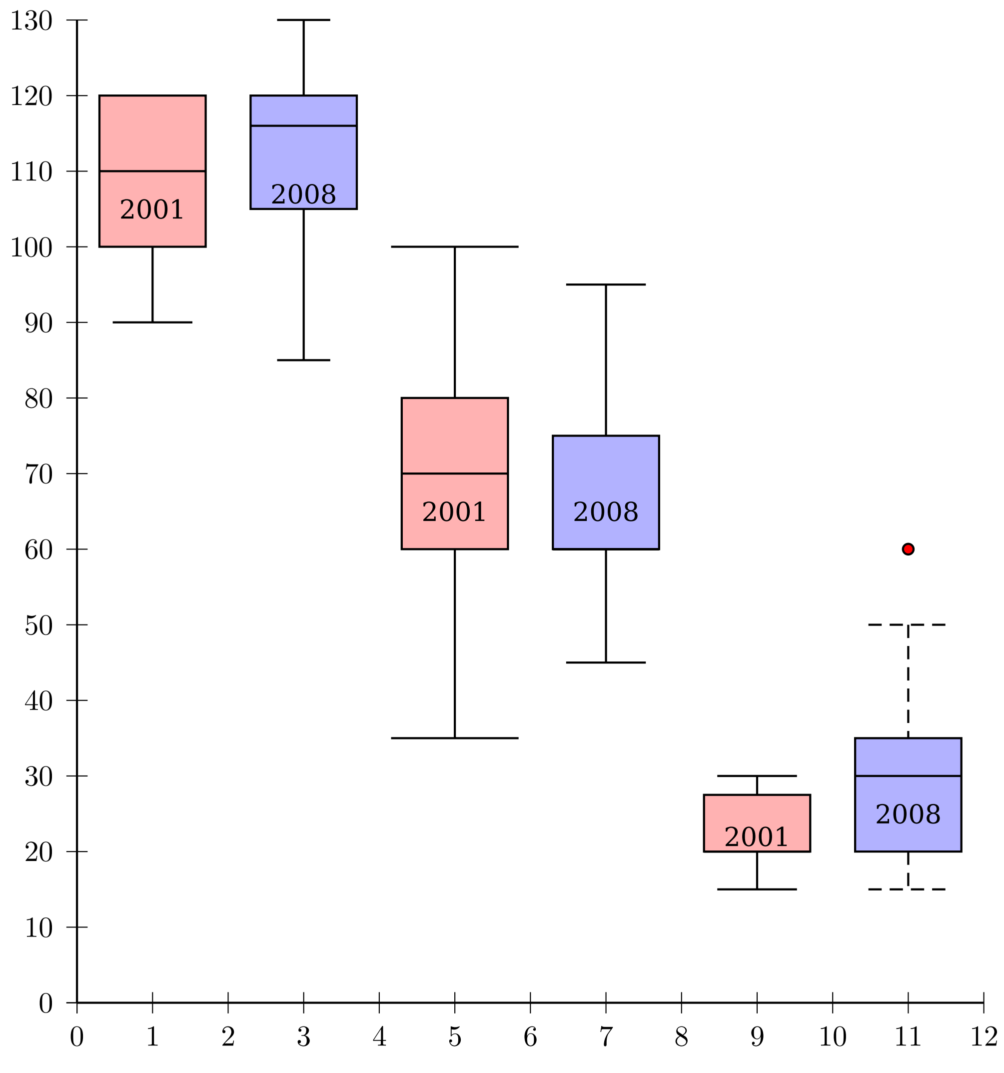



```
\psset{xunit=.6,yunit=0.8,plotpoints=500}
\begin{pspicture}(0,-4.25)(26,5.25)
  \psaxes{->}(0,0)(0,-4)(26,5)
  \psplot[plotpoints=200,linewidth=4\pslinewidth,linecolor=gray]{0}{26}{%
     Euler x -8 div exp x 127 sqrt 8 div mul RadtoDeg dup cos 5 mul exch sin
       127 sqrt div 5 mul add mul}
  \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}
```

37. \psBoxplot

A box-and-whisker plot (often called simply a box plot) is a histogram-like method of displaying data, invented by John. Tukey. The box-and-whisker plot is a box with ends at the quartiles Q_1 and Q_3 and has a statistical median M as a horizontal line in the box. The "‘whiskers"* are lines to the farthest points that are not outliers (i.e., that are within 3/2 times the interquartile range of Q_1 and Q_3). Then, for every point more than 3/2 times the interquartile range from the end of a box, is a dot.

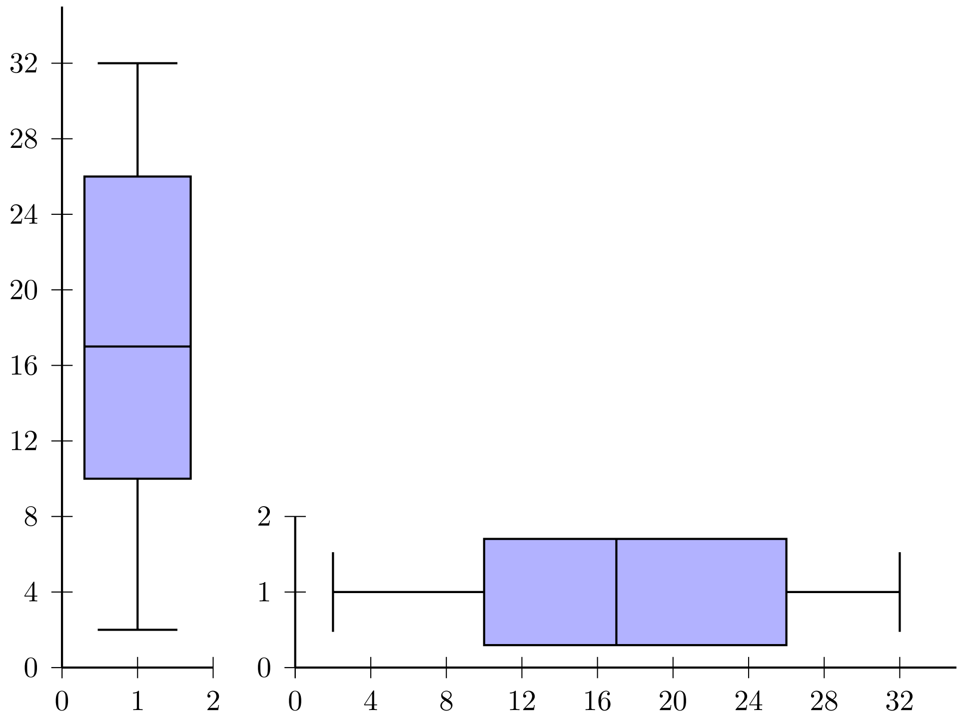
The only special optional arguments, beside all other which are valid for drawing lines and filling areas, are `IQLfactor`, `barwidth`, and `arrowlength`, where the latter is a factor which is multiplied with the barwidth for the line ends. The `IQLfactor`, preset to 1.5, defines the area for the outliers.




```
\begin{pspicture}(-1,-1)(12,14)
\psset{yunit=0.1,fillstyle=solid}
\savedata{\data}[100 90 120 115 120 110 100 110 100 90 100 100 120 120 120]
```

```
\rput(1,0){\psBoxplot[fillcolor=red!30]{\data}}
\rput(1,105){2001}
\savedata{\data}[90 120 115 116 115 110 90 130 120 120 120 85 100 130 130]
\rput(3,0){\psBoxplot[arrowlength=0.5,fillcolor=blue!30]{\data}}
\rput(3,107){2008}
\savedata{\data}[35 70 90 60 100 60 60 80 80 60 50 55 90 70 70]
\rput(5,0){\psBoxplot[barwidth=40pt,arrowlength=1.2,fillcolor=red!30]{\data
  }}
\rput(5,65){2001}
\savedata{\data}[60 65 60 75 75 60 50 90 95 60 65 45 45 60 90]
\rput(7,0){\psBoxplot[barwidth=40pt,fillcolor=blue!30]{\data}}
\rput(7,65){2008}
\savedata{\data}[20 20 25 20 15 20 20 25 30 20 20 20 30 30 30]
\rput(9,0){\psBoxplot[fillcolor=red!30]{\data}}
\rput(9,22){2001}
\savedata{\data}[20 30 20 35 35 20 20 60 50 20 35 15 30 20 40]
\rput(11,0){\psBoxplot[fillcolor=blue!30,linestyle=dashed]{\data}}
\rput(11,25){2008}
\psaxes[dy=1cm,Dy=10](0,0)(12,130)
\end{pspicture}
```

The next example uses an external file for the data, which must first be read by the macro `\readdata`. The next one creates a horizontal boxplot by rotating the output with -90 degrees.




```
\readdata{\data}{boxplot.data}
\begin{pspicture}(-1,-1)(2,10)
```

```
\psset{yunit=0.25,fillstyle=solid}
\savedata{\data}[2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32]
\rput(1,0){\psBoxplot[fillcolor=blue!30]{\data}}
\psaxes[dy=1cm,Dy=4](0,0)(2,35)
\end{pspicture}
%
\begin{pspicture}(-1,-1)(11,2)
\psset{xunit=0.25,fillstyle=solid}
\savedata{\data}[2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32]
\rput{-90}(0,1){\psBoxplot[yunit=0.25,fillcolor=blue!30]{\data}}
\psaxes[dx=1cm,Dx=4](0,0)(35,2)
\end{pspicture}
```

38. \psMatrixPlot

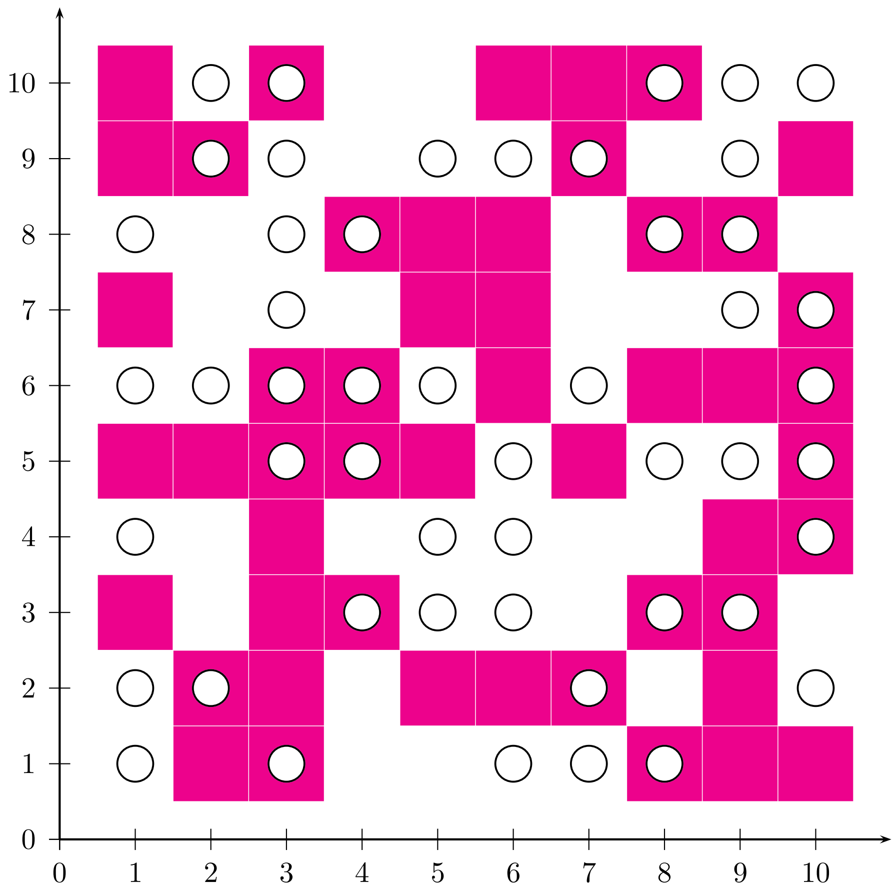
This macro allows you to visualize a matrix. The datafile must be defined as a PostScript matrix named dotmatrix:

```
/dotmatrix [ % <------------ important line
0 1 1 0 0 0 0 1 1 1
0 1 1 0 1 1 1 0 1 0
1 0 1 1 0 0 0 1 1 0
0 0 1 0 0 0 0 0 1 1
1 1 1 1 1 0 1 0 0 1
0 0 1 1 0 1 0 1 1 1
1 0 0 0 1 1 0 0 0 1
0 0 0 1 1 1 0 1 1 0
1 1 0 0 0 0 1 0 0 1
1 0 1 0 0 1 1 1 0 0
] def    % <------------ important line
```

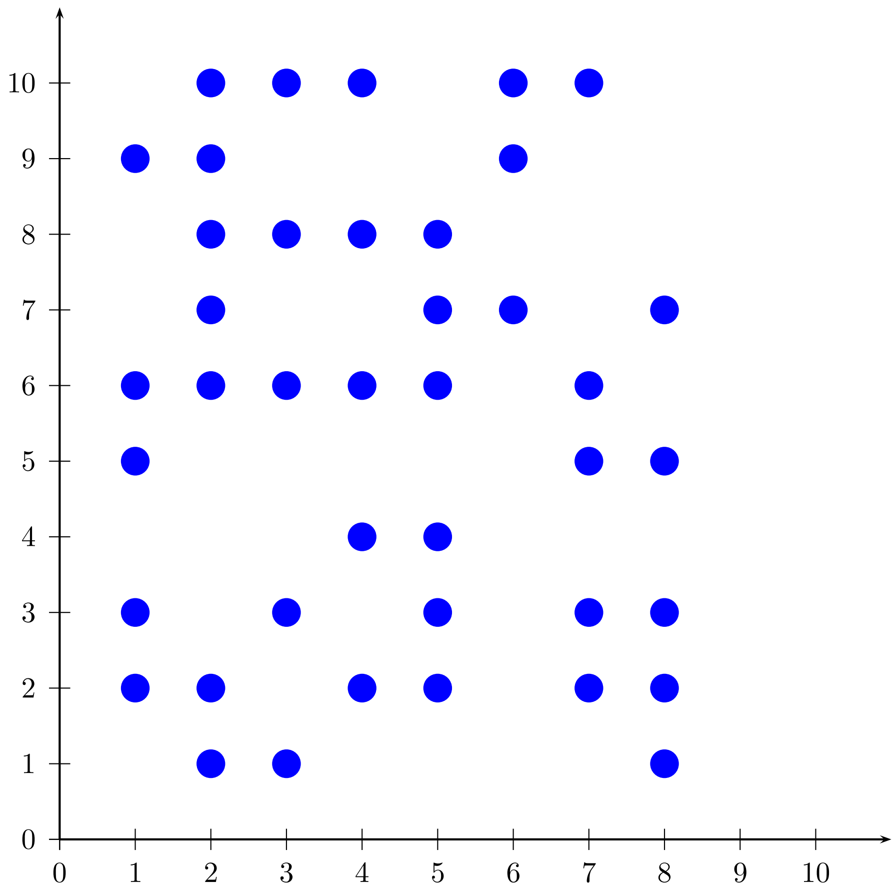
Only the value 0 is important, in which case nothing happens, and for all other cases a dot is printed. The syntax of the macro is:

\psMatrixPlot [Options] {*rows*}{*columns*}{*data file*}

The matrix is scanned line by line from the the first one to the last. In general it appears as a bottom-to-top version of the above listed matrix, the first row $0\,1\,1\,0\,0\,0\,0\,1\,1\,1$ is the first plotted line ($y = 1$). With the option ChangeOrder=true it looks exactly like the above view.




```
\begin{pspicture}(-0.5,-0.75)(11,11)
  \psaxes{->}(11,11)
  \psMatrixPlot[dotsize=1.1cm,dotstyle=square*,linecolor=magenta]%
    {10}{10}{matrix.data}
  \psMatrixPlot[dotsize=.5cm,dotstyle=o,ChangeOrder]{10}{10}{matrix.data}
\end{pspicture}
```



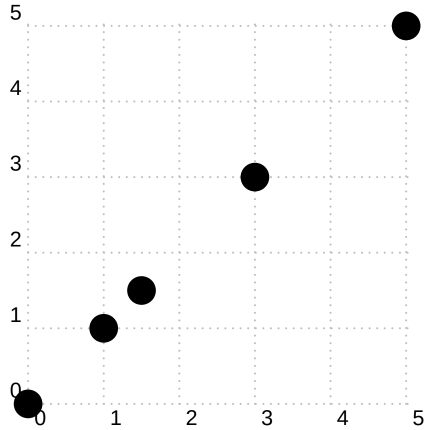

```
\begin{pspicture}(-0.5,-0.75)(11,11)
  \psaxes{->}(11,11)
  \psMatrixPlot[dotscale=3,dotstyle=*,linecolor=blue]{10}{8}{matrix.data}
\end{pspicture}
```

39. \psforeach and \psForeach

The macro `\psforeach` allows a loop with an individual increment.

`\psforeach{`*variable*`}{`*value list*`}{`*action*`}`
`\psForeach{`*variable*`}{`*value list*`}{`*action*`}`

With `\psforeach` the *action* is done inside a group and for `\psForeach` not. This maybe useful when using the macro to create tabular cells, which are alread grouped itself.




```
\begin{pspicture}[showgrid=true](5,5)
 \psforeach{\nA}{0, 1, 1.5, 3, 5}{%
   \psdot[dotscale=3](\nA,\nA)}
\end{pspicture}
```

$y = 2^x$

2	4	6	8	10	12
4.0	16.0	64.0	256.0	1024.0	4096.0

```
%\usepackage{pst-func}
\makeatletter
\newcommand*\InitToks{\toks@={}}
\newcommand\AddToks[1]{\toks@=\expandafter{\the\toks@ #1}}
\newcommand*\PrintToks{\the\toks@}
\newcommand*{\makeTable}[4][5mm]{%
  \begingroup
    \InitToks%
    \AddToks{\begin{tabular}{|*{#2}{>{\RaggedLeft}p{#1}|}@{}l@{}}\cline{1-#2}}
    \psForeach{\iA}{#3}{\expandafter\AddToks\expandafter{\iA & }}
    \AddToks{\\\cline{1-#2}}%
    \psForeach{\iA}{#3}{\expandafter\AddToks\expandafter{\expandafter%
      \psPrintValue\expandafter{\iA\space /x ED #4} & }}
    \AddToks{\\\cline{1-#2}\end{tabular}}%
    \PrintToks
  \endgroup
}
\makeatother

\sffamily
\psset{decimals=2,valuewidth=7,xShift=-20}
$y=2^x$\\
\makeTable[1cm]{6}{2,4,6,8,10,12}{2 x exp}
```

40. \resetOptions

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

A. PostScript

PostScript uses the stack system and the LIFO system, "'Last In, First Out"'.

Table 5: Some primitive PostScript macros

Function	*Meaning* on stack before		*after*
`add`	$x \quad y$	$\rightarrow$	$x + y$
`sub`	$x \quad y$	$\rightarrow$	$x - y$
`mul`	$x \quad y$	$\rightarrow$	$x \times y$
`div`	$x \quad y$	$\rightarrow$	$x \div y$
`sqrt`	x	$\rightarrow$	$\sqrt{x}$
`abs`	x	$\rightarrow$	$\lvert x \rvert$
`neg`	x	$\rightarrow$	$-x$
`cos`	x	$\rightarrow$	$\cos(x)$ (`x in degrees`)
`sin`	x	$\rightarrow$	$\sin(x)$ (`x in degrees`)
`tan`	x	$\rightarrow$	$\tan(x)$ (`x in degrees`)
`atan`	$y \quad x$	$\rightarrow$	$\angle(\vec{Ox};\vec{OM})$ (`in degrees of` $M(x,y)$)
`ln`	x	$\rightarrow$	$\ln(x)$
`log`	x	$\rightarrow$	$\log(x)$
`array`	n	$\rightarrow$	v (of dimension n)
`aload`	v	$\rightarrow$	$x_1 \quad x_2 \quad \cdots \quad x_n \quad v$
`astore`	$x_1 \quad x_2 \quad \cdots \quad x_n \quad v$	$\rightarrow$	$[v]$
`pop`	x	$\rightarrow$	-
`dup`	x	$\rightarrow$	$x \quad x$

B. List of all optional arguments for pstricks-add

Key	Type	Default
CMYK	boolean	true
fsAngle	ordinary	137.50775
fsOrigin	ordinary	[none]
maxdashes	ordinary	11
intSeparator	ordinary	[none]
braceWidth	ordinary	[none]
bracePos	ordinary	[none]
braceWidthInner	ordinary	[none]
braceWidthOuter	ordinary	[none]
veearrowlength	ordinary	3mm
veearrowangle	ordinary	30
veearrowlinewidth	ordinary	0.35mm
filledveearrowlength	ordinary	3mm
filledveearrowangle	ordinary	15
filledveearrowlinewidth	ordinary	0.35mm
arrowLW	ordinary	[none]
tickarrowlength	ordinary	1.5mm
tickarrowlinewidth	ordinary	0.35mm
hooklength	ordinary	3mm
hookwidth	ordinary	1mm
ArrowFill	boolean	true
nArrowsA	ordinary	2
nArrowsB	ordinary	2
nArrows	ordinary	2
ArrowInside	ordinary	[none]
ArrowInsidePos	ordinary	0.5
ArrowInsideNo	ordinary	1
ArrowInsideOffset	ordinary	0
dashNo	ordinary	[none]
linecap	ordinary	[none]
randomPoints	ordinary	1000
color	boolean	true
lineAngle	ordinary	0
trueAngle	boolean	true
blName	command	
bcName	command	
brName	command	
clName	command	
ccName	command	
crName	command	
tlName	command	
tcName	command	

Continued on next page

Continued from previous page

Key	Type	Default
trName	command	
yMaxValue	ordinary	-1.0
labelFontSize	ordinary	
mathLabel	boolean	true
decimalSeparator	ordinary	.
comma	boolean	true
xAxis	boolean	true
yAxis	boolean	true
xyAxes	boolean	true
xlabelPos	ordinary	b
ylabelPos	ordinary	l
xyDecimals	ordinary	
xDecimals	ordinary	
yDecimals	ordinary	
xlogBase	ordinary	
ylogBase	ordinary	
xylogBase	ordinary	
trigLabelBase	ordinary	0
trigLabelsSimplify	boolean	true
trigLabels	boolean	true
logLines	ordinary	none
ylabelFactor	ordinary	\relax
xlabelFactor	ordinary	\relax
showOriginTick	boolean	true
xticksize	ordinary	[none]
yticksize	ordinary	[none]
subticks	ordinary	1
xsubticks	ordinary	1
ysubticks	ordinary	1
subticksize	ordinary	0.75
xsubticksize	ordinary	0.75
ysubticksize	ordinary	0.75
tickwidth	ordinary	0.5\pslinewidth
xtickwidth	ordinary	0.5\pslinewidth
ytickwidth	ordinary	0.5\pslinewidth
subtickwidth	ordinary	0.25\pslinewidth
xsubtickwidth	ordinary	0.25\pslinewidth
ysubtickwidth	ordinary	0.25\pslinewidth
tickcolor	ordinary	black
xtickcolor	ordinary	black
ytickcolor	ordinary	black
subtickcolor	ordinary	gray
xsubtickcolor	ordinary	gray
ysubtickcolor	ordinary	gray

Continued on next page

Continued from previous page

Key	Type	Default
xticklinestyle	ordinary	solid
xsubticklinestyle	ordinary	solid
yticklinestyle	ordinary	solid
ysubticklinestyle	ordinary	solid
ticklinestyle	ordinary	solid
subticklinestyle	ordinary	solid
nStep	ordinary	1
nStart	ordinary	0
nEnd	ordinary	
xStep	ordinary	0
yStep	ordinary	0
xStart	ordinary	
xEnd	ordinary	
yStart	ordinary	
yEnd	ordinary	
plotNo	ordinary	1
plotNoMax	ordinary	1
xyValues	boolean	true
ChangeOrder	boolean	true
xAxisLabel	ordinary	x
yAxisLabel	ordinary	y
xAxisLabelPos	ordinary	
yAxisLabelPos	ordinary	
llx	ordinary	\z@
lly	ordinary	\z@
urx	ordinary	\z@
ury	ordinary	\z@
box	ordinary	true
ignoreLines	ordinary	[none]
polarplot	boolean	true
algebraic	boolean	true
method	ordinary	[none]
whichabs	ordinary	[none]
whichord	ordinary	[none]
plotfuncx	ordinary	[none]
plotfuncy	ordinary	[none]
expression	ordinary	[none]
buildvector	boolean	true
VarStep	boolean	true
PlotDerivative	ordinary	[none]
VarStepEpsilon	ordinary	[none]
varsteptol	ordinary	[none]
adamsorder	ordinary	[none]
barwidth	ordinary	[none]

Continued on next page

Continued from previous page

Key	Type	Default
`PSfont`	`ordinary`	`Times-Roman`
`valuewidth`	`ordinary`	`10`
`fontscale`	`ordinary`	`10`
`decimals`	`ordinary`	`-1`
`StepType`	`ordinary`	`[none]`
`Derive`	`ordinary`	`[none]`
`Tnormal`	`boolean`	`true`
`filename`	`ordinary`	`[none]`
`saveData`	`boolean`	`true`
`IQLfactor`	`ordinary`	`[none]`
`chartStyle`	`ordinary`	`[none]`
`chartColor`	`ordinary`	`[none]`
`chartSep`	`ordinary`	`[none]`
`chartStack`	`ordinary`	`[none]`
`chartStackDepth`	`ordinary`	`[none]`
`chartStackWidth`	`ordinary`	`[none]`
`chartHeight`	`ordinary`	`[none]`
`userColor`	`ordinary`	`[none]`
`chartNodeI`	`ordinary`	`[none]`
`chartNodeO`	`ordinary`	`[none]`

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, Sebastian Rahtz, Denis Roegel, and Herbert Voß. *The LaTeX Graphics Companion*. Addison-Wesley Publishing Company, Reading, Mass., 2007.

[4] Alan Hoenig. *TeX Unbound: LaTeX & TeX Strategies, Fonts, Graphics, and More*. Oxford University Press, London, 1998.

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

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

[7] Frank Mittelbach and Michel Goosens et al. *The LaTeX Companion*. Addison-Wesley Publishing Company, Boston, second edition, 2004.

[8] Frank Mittelbach and Michel Goosens et al. *Der LaTeX Begleiter*. Pearson Education, München, zweite edition, 2005.

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

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

[11] Herbert Voß. *`PSTricks` Grafik für TEX und LATEX*. DANTE – Lob.media, Heidelberg/Hamburg, fifth edition, 2008.

[12] Herbert Voß. *Mathematiksatz in LATEX*. Lehmanns Media/DANTE, Berlin/Heidelberg, first edition, 2009.

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

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

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

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

Index

W

X

Y