pst-node

Nodes and connections*

v.2.00

Herbert Voß

May 6, 2005

Abstract

This version of `pst-node` includes all the macros which were for testing part of the `pstricks-add`-package. This documentation shows only these extensions. For the other macros have a look into the old PSTricks documentation.

`pst-node` uses the extended version of the keyval package. So be sure, that you have installed `pst-xkey` which is part of the `xkeyval`-package and that all packages, that uses the old keyval interface are loaded **before** the `xkeyval`.

*This document was written with `Kile: 1.7 (Qt: 3.1.1; KDE: 3.3;` `http://sourceforge.net/projects/kile/`) and the PDF output was build with VTeX/Free (`http://www.micropress-inc.com/linux`)

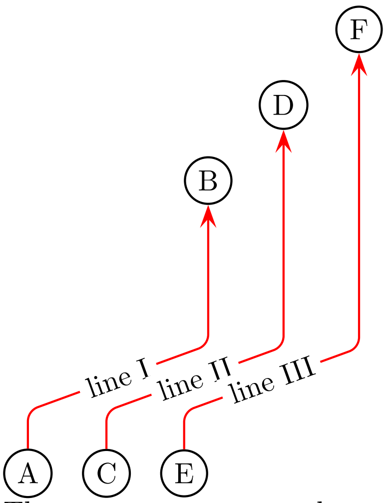
Contents

1 \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

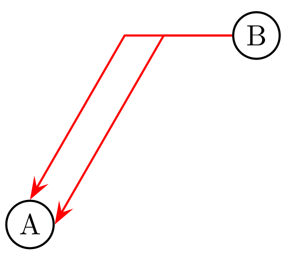
```
\ncdiag[<options>]{<Node A>}{<Node B>}
\pcdiag[<options>](<Node A>)(<Node B>)
```

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

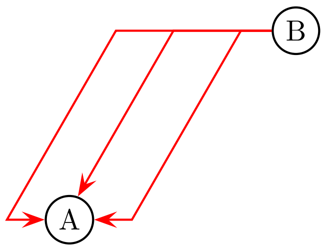



```
\begin{pspicture}(5,6)
  \circlenode{A}{A}\quad\circlenode{C}{C}%
    \quad\circlenode{E}{E}
  \rput(0,4){\circlenode{B}{B}}
  \rput(1,5){\circlenode{D}{D}}
  \rput(2,6){\circlenode{F}{F}}
  \psset{arrowscale=2,linearc=0.2,%
    linecolor=red,armA=0.5, angleA=90,angleB=-90}
  \ncdiag[lineAngle=20]{->}{A}{B}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

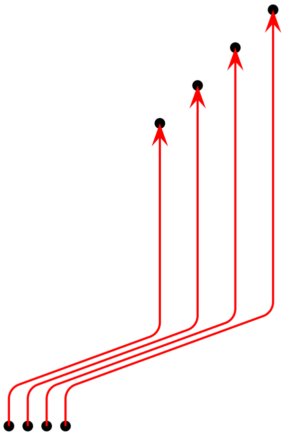
The `ncdiag` macro sets the `armB` dynamically to the calculated value. Any user setting of `armB` is overwritten by the macro. The `armA` could be set to a zero length:




```
\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}
```

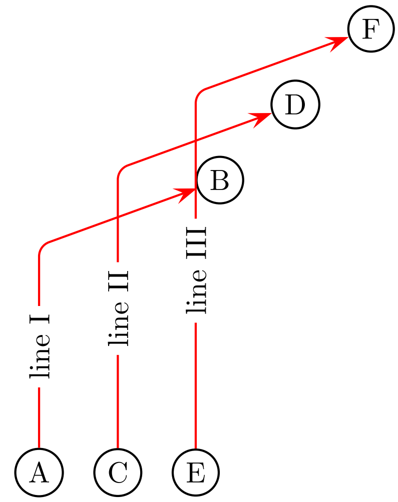


```
\begin{pspicture}(4,5.5)
  \cnode*(0,0){2pt}{A}%
  \cnode*(0.25,0){2pt}{C}%
  \cnode*(0.5,0){2pt}{E}%
  \cnode*(0.75,0){2pt}{G}%
  \cnode*(2,4){2pt}{B}%
  \cnode*(2.5,4.5){2pt}{D}%
  \cnode*(3,5){2pt}{F}%
  \cnode*(3.5,5.5){2pt}{H}%
  {\psset{arrowscale=2,linearc=0.2,%
    linecolor=red,armA=0.5,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

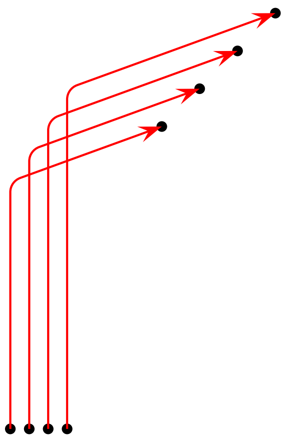
2 \ncdiagg and \pcdiagg

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

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

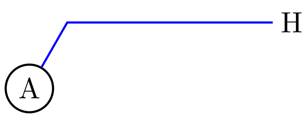



```
\begin{pspicture}(4,6)
  \psset{linecolor=black}
  \circlenode{A}{A}%
  \quad\circlenode{C}{C}%
  \quad\circlenode{E}{E}
  \rput(0,4){\circlenode{B}{B}}
  \rput(1,5){\circlenode{D}{D}}
  \rput(2,6){\circlenode{F}{F}}
  {\psset{arrowscale=2,linearc=0.2,linecolor=red,armA=0.5, angleA=90}
  \ncdiagg[lineAngle=-160]{->}{A}{B}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

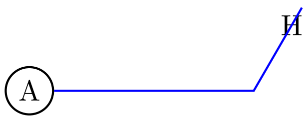


```
\begin{pspicture}(4,6)
  \psset{linecolor=black}
  \cnode*(0,0){2pt}{A}%
  \cnode*(0.25,0){2pt}{C}%
  \cnode*(0.5,0){2pt}{E}%
  \cnode*(0.75,0){2pt}{G}%
  \cnode*(2,4){2pt}{B}%
  \cnode*(2.5,4.5){2pt}{D}%
  \cnode*(3,5){2pt}{F}%
  \cnode*(3.5,5.5){2pt}{H}%
  {\psset{arrowscale=2,linearc=0.2,linecolor=red,armA=0.5,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

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




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

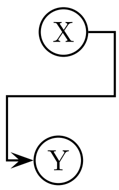



```
\begin{pspicture}(4,1.5)
  \circlenode{a}{A}
  \rput[l](3,1){\rnode{b}{H}}
  \ncdiagg[lineAngle=60,armA=.5,nodesepB=3pt,linecolor=blue]{a
    }{b}
\end{pspicture}
```

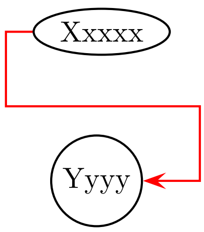

```
\begin{pspicture}(4,1.5)
  \circlenode{a}{A}
  \rput[l](3,1){\rnode{b}{H}}
  \ncdiagg[lineAngle=-120,armA=.5,nodesepB=3pt,linecolor=blue]{
    a}{b}
\end{pspicture}
```

3 \ncbarr und \pcbarr

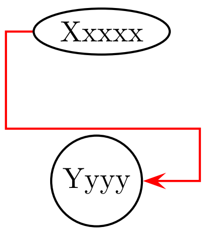
This has the same behaviour as `ncbar`, but has 5 segments and all are horizontal ones. This is the reason why `angleA` should be $0°$ or alternative $180°$. Other values are possible but don't make really sense. `AngleB` is in geenral `AngleA`180+ and cannot be set to another value. The intermediate horizontal line is symmetrical to the distance of the two nodes and can be set with the option `tpos` ($0 \leq tpos \leq 1$).



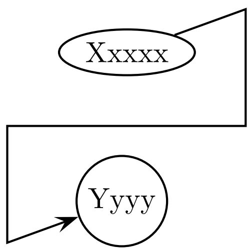

```
\psset{arrowscale=2}%
\circlenode{X}{X}\\[1cm]
\circlenode{Y}{Y}
\ncbarr[angleA=0,arrows=->,arrowscale=2]{X}{Y}
```



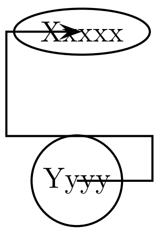

```
\psset{arrowscale=2}%
\ovalnode{X}{Xxxxx}\\[1cm]
\circlenode{Y}{Yyyy}
\ncbarr[angleA=180,arrows=->,arrowscale=2,linecolor=red,armB
  =0.75]{X}{Y}
```




```
\psset{arrowscale=2}%
\ovalnode{X}{Xxxxx}\\[1cm]
\circlenode{Y}{Yyyy}
\ncbarr[angleA=180,arrows=->,arrowscale=2,linecolor=red,armB
  =0.75,tpos=0.65]{X}{Y}
```



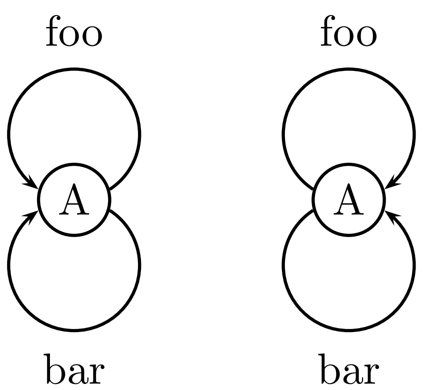

```
\psset{arrowscale=2}%
\ovalnode{X}{Xxxxx}\\[1cm]
\circlenode{Y}{Yyyy}
\ncbarr[angleA=20,arm=1cm,arrows=->,arrowscale=2]{X}{Y}
```




```
\psset{arrowscale=2}%
\ovalnode{X}{Xxxxx}\\[1cm]
\circlenode{Y}{Yyyy}
\pcbarr[arm=1cm,arrows=->,arrowscale=2,tpos=0.3](Y)(X)
```

4 \nccircle

The changes to the code of `psarc@v` now allows to draw node loops with `\pscircle` (`\pst-node`) clockwise below a node.




```
\psscalebox{1.5}{
\circlenode{A}{A}
\nccircle{->}{A}{1.5em}
\nbput{foo}
\nccircle{<-}{A}{-1.5em}
\naput{bar}}
%
\hspace*{2cm}
\psscalebox{1.5}{
\circlenode{A}{A}
\nccircle{<-}{A}{1.5em}
\nbput{foo}
\nccircle{->}{A}{-1.5em}
\naput{bar}}
```

5 \psRelLine

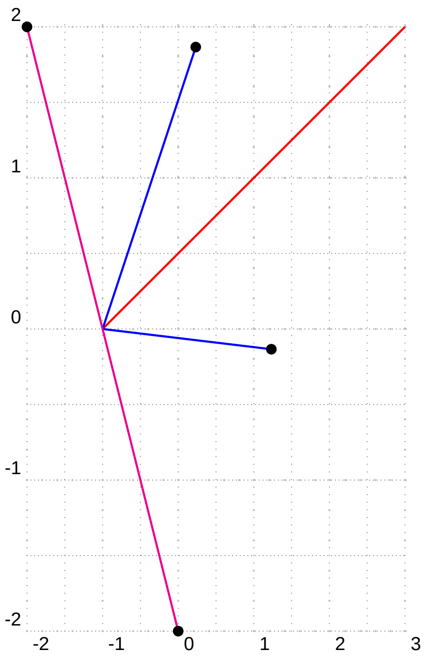
With this macro it is possible to plot lines relative to a given one. Parameter are the angle and the length factor:

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

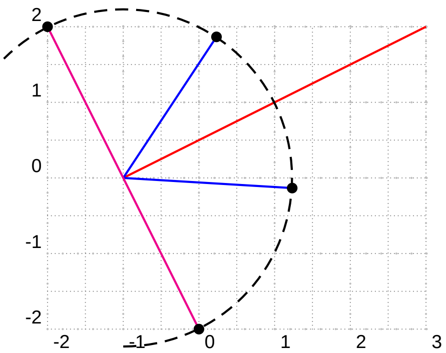
The length factor depends to the distance of $\overline{P_0P_1}$ and the end node name must be a valid nodename and shouldn't contain any of the special PostScript characters. There are two valid options:

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

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

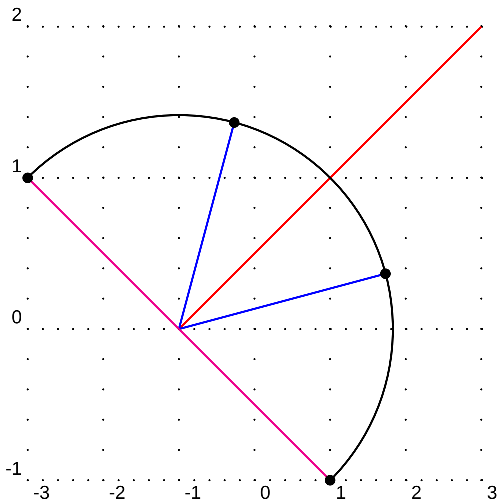



```
\psset{yunit=2,xunit=1}
\begin{pspicture}(-2,-2)(3,2)
\psgrid[subgriddiv=2,subgriddots=10,gridcolor=
  lightgray]
\pnode(-1,0){A}\pnode(3,2){B}
\psline[linecolor=red](A)(B)
\psRelLine[linecolor=blue,angle=30](-1,0)(B){0.5}{
  EndNode}
\qdisk(EndNode){2pt}
\psRelLine[linecolor=blue,angle=-30](A)(B){0.5}{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



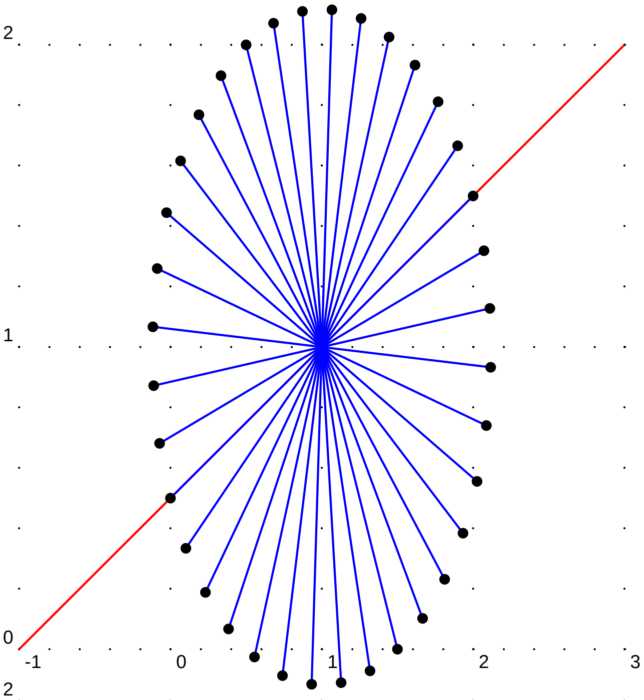

```
\begin{pspicture}(-2,-2)(3,2)
\psgrid[subgriddiv=2,subgriddots=10,gridcolor=lightgray]
\pnode(-1,0){A}\pnode(3,2){B}
\psline[linecolor=red](A)(B)
\psarc[linestyle=dashed](A){2.23}{-90}{135}
\psRelLine[linecolor=blue,angle=30](-1,0)(B){0.5}{EndNode}
\qdisk(EndNode){2pt}
\psRelLine[linecolor=blue,angle=-30](A)(B){0.5}{EndNode}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

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

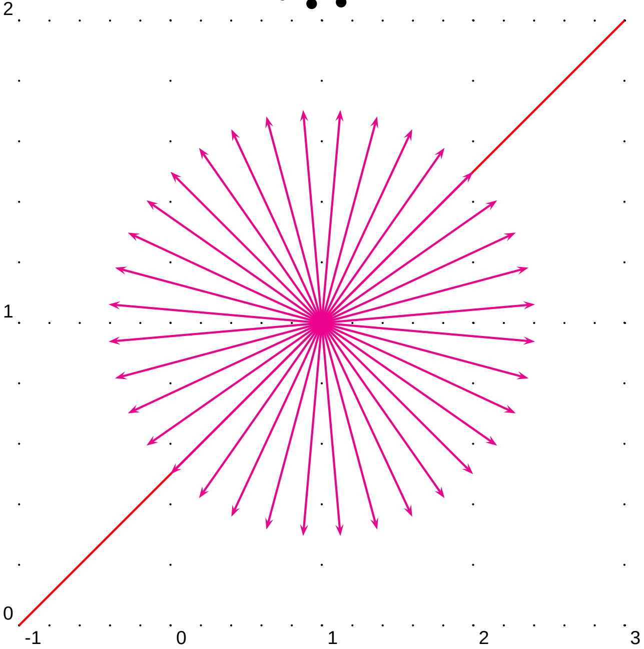



```
\psset{yunit=2,xunit=1}
\begin{pspicture}(-3,-1)(3,2)\psgrid[subgridcolor=lightgray]
\pnode(-1,0){A}\pnode(3,2){B}
\psline[linecolor=red](A)(B)
\psarc(A){2.83}{-45}{135}
\psRelLine[linecolor=blue,angle=30,trueAngle](A)(B){0.5}{EndNode}
\qdisk(EndNode){2pt}
\psRelLine[linecolor=blue,angle=-30,trueAngle](A)(B){0.5}{EndNode}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

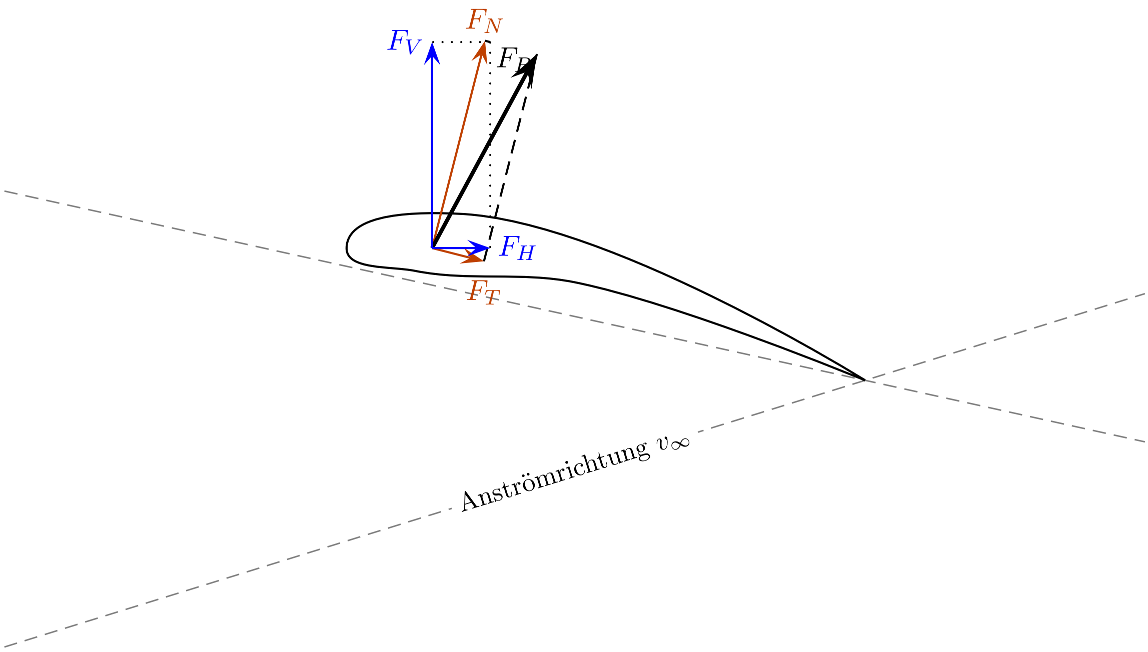
Two examples with using `\multido` to show the behaviour of the options `trueAngle` and `angle`.




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




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



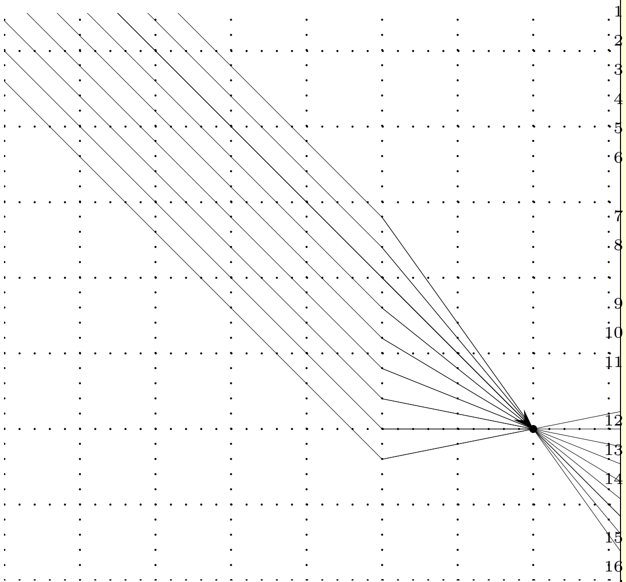

```
\psset{xunit=\linewidth,yunit=\linewidth,trueAngle}%
\begin{pspicture}(1,0.6)%\psgrid
  \pnode(.3,.35){Vk} \pnode(.375,.35){D}
  \pnode(0,.4){DST1} \pnode(1,.18){DST2}
  \pnode(0,.1){A1} \pnode(1,.31){A1}
  { \psset{linewidth=.02,linestyle=dashed,linecolor=gray}%
    \pcline(DST1)(DST2) % <- Druckseitentangente
    \pcline(A2)(A1) % <- Anströmmrichtung
    \lput*{:U}{\small Anströmrichtung $v_{\infty}$}
  }%
  \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}
```

6 \psParallelLine

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

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

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

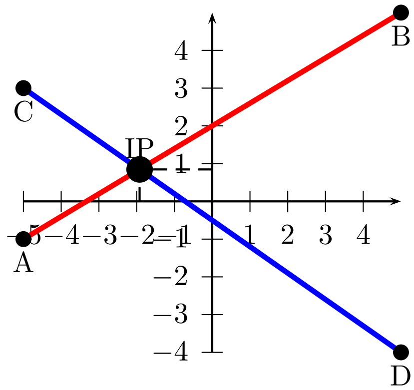


```
\begin{pspicture*}(-5,-4)(5,3.5)
  \psgrid[subgriddiv=0,griddots=5]
  \pnode(2,-2){FF}\qdisk(FF){1.5pt}
  \pnode(-5,5){A}\pnode(0,0){O}
  \multido{\nCountA=-2.4+0.4}{9}{%
    \psParallelLine[linecolor=red](O)(A)
      (0,\nCountA){9}{P1}
    \psline[linecolor=red](0,\nCountA)(FF)
    \psRelLine[linecolor=red](0,\nCountA)(
      FF){9}{P2}
  }
  \psline[linecolor=blue](A)(FF)
  \psRelLine[linecolor=blue](A)(FF){5}{
    END1}
  \rput(0,0){%
    \psline[linewidth=1pt](xLeft)(xRight)
    \psline[linewidth=2pt,arrows=->](F')(
      FF)
  }%
\end{pspicture*}
```

7 \psIntersectionPoint

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

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

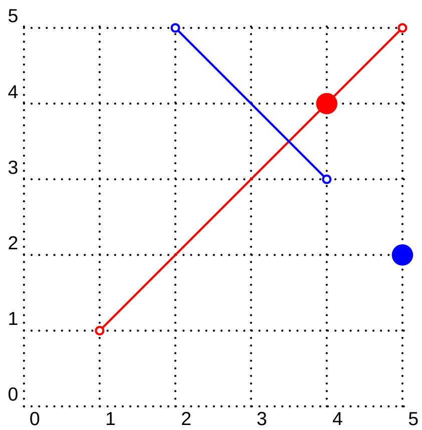



```
\psset{unit=0.5cm}
\begin{pspicture}(-5,-4)(5,5)
  \psaxes{->}(0,0)(-5,-4)(5,5)
  \psline[linecolor=red,linewidth=2pt](-5,-1)(5,5)
  \psline[linecolor=blue,linewidth=2pt](-5,3)(5,-4)
  \qdisk(-5,-1){3pt}\uput[-90](-5,-1){A}
  \qdisk(5,5){3pt}\uput[-90](5,5){B}
  \qdisk(-5,3){3pt}\uput[-90](-5,3){C}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

8 \psLNode and \psLCNode

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

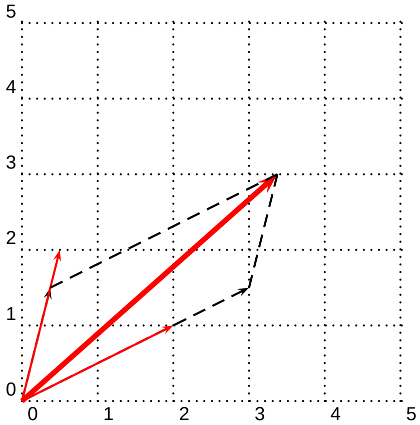
```
\setLNode(P1)(P2){value}{Node name}
```



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

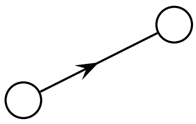
The `\psLCNode` macro builds the linear combination of the two given vectors and stores the end of the new vector as a node. All vectors start at $(0,0)$, so a `\rput` maybe appropriate. The syntax is

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

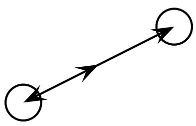



```
\begin{pspicture}(5,5)
\psgrid[subgriddiv=0,griddots=10]
\psset{linecolor=black}
\psline[linestyle=dashed]{->}(3,1.5)
\psline[linestyle=dashed]{->}(0.375,1.5)
\psset{linecolor=red}
\psline{->}(2,1)\psline{->}(0.5,2)
\setLCNode(2,1){1.5}(0.5,2){0.75}{PI}
\psline[linewidth=2pt]{->}(PI)
\psset{linecolor=black}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

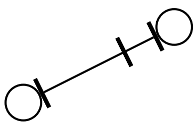
9 \ncline, \pcline and inside errors



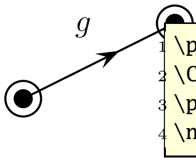
```
\begin{pspicture}(2,1)
\psset{arrowscale=2}
\Cnode(0,0){A}\Cnode(2,1){B}
\ncline[ArrowInside=->]{A}{B}
\end{pspicture}
```



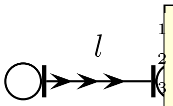
```
\begin{pspicture}(2,1)
\psset{arrowscale=2}
\Cnode(0,0){A}\Cnode(2,1){B}
\pcline[ArrowInside=->]{<->}(A)(B)
\end{pspicture}
```



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

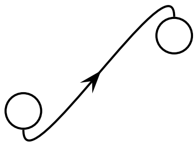



```
\psset{arrowscale=2}
\Cnode(0,0){A}\Cnode(2,1){B}
\pcline[ArrowInside=->,ArrowInsidePos=0.65]{*-*}(A)(B)
\naput[labelsep=0.3]{\large$g$}
```

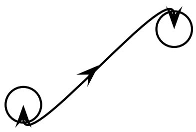



```
\psset{arrowscale=2}
\Cnode(0,0){A}\Cnode(2,0){B}
\ncline[ArrowInside=->,ArrowInsidePos=10]{|-|}{A}{B}
\naput[labelsep=0.3]{\large$l$}
```

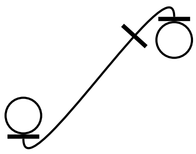
10 **\nccurve, \pccurve** and inside errors



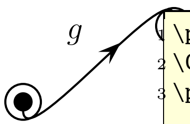
```
\begin{pspicture}(2,1)
\psset{arrowscale=2}
\Cnode(0,0){A}\Cnode(2,1){B}
\nccurve[ArrowInside=->,angleA=-90,angleB=90]{A}{B}
\end{pspicture}
```



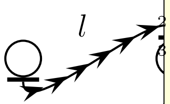
```
\begin{pspicture}(2,1)
\psset{arrowscale=2}
\Cnode(0,0){A}\Cnode(2,1){B}
\pccurve[ArrowInside=->,angleA=-90,angleB=90]{<->}(A)(B)
\end{pspicture}
```



```
\begin{pspicture}(2,1)
\psset{arrowscale=2}
\Cnode(0,0){A}\Cnode(2,1){B}
\nccurve[ArrowInside=-|,ArrowInsidePos=0.75,angleA=-90,angleB=90]{|-|}{
  A}{B}
\end{pspicture}
```




```
\psset{arrowscale=2}
\Cnode(0,0){A}\Cnode(2,1){B}
\pccurve[ArrowInside=->,ArrowInsidePos=0.65,angleA=-90,angleB=90]{*-*}(
  A)(B)
\naput[labelsep=0.3]{\large$g$}
```




```
\psset{arrowscale=2}
\Cnode(0,0){A}\Cnode(2,0){B}
\nccurve[ArrowInside=->,ArrowInsidePos=10,angleA=-90,angleB=90]{|-|}{A
  }{B}
\naput[labelsep=0.3]{\large$l$}
```

11 \resetPSTNodeOptions

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

```
\def\resetPSTNodeOptions{%
\psset[pst-node]{%
  nodealign=false,%
  href=0,%
  vref=.7ex,%
  framesize=10pt,%
  nodesep=0pt,%
  arm=10pt,%
  offset=0pt,%
  angle=0,%
  arcangle=8,%
  ncurv=.67,%
  loopsize=1cm,%
  boxsize=.4cm,%
  nrot=0,%
  npos=,%
  tpos=0.5,%
  shortput=none,%
  colsep=1.5cm,%
  rowsep=1.5cm,%
%  shortput=tablr,%%
  mcol=c,%
  mnode=R,%
  emnode=none}
}
```

12 Credits

References

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

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

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

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

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

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

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

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

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

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