

pst-node

plotting data and mathematical functions*

v.2.00

Herbert Voß

December 23, 2004

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**.

Contents

1	\ncdiag and \pcdiag	2
2	\ncdiagg and \pcdiagg	3
3	\ncbarr und \pcbarr	5
4	\psRelLine	6
5	\psParallelLine	11

*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>)

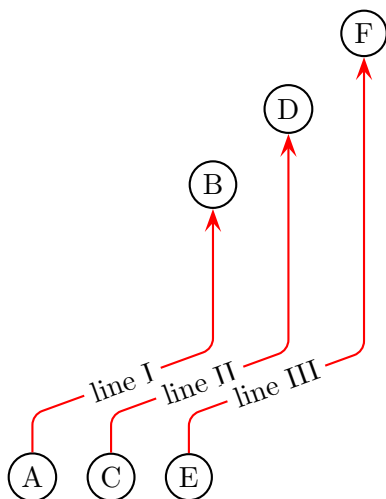
6	<code>\psIntersectionPoint</code>	11
7	<code>\psLNode</code> and <code>\psLCNode</code>	12
8	<code>\ncline</code> , <code>\pcline</code> and inside errors	13
9	<code>\nccurve</code> , <code>\pccurve</code> and inside errors	14
10	<code>\resetPSTNodeOptions</code>	15
11	Credits	15

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

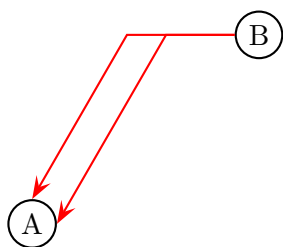
```
\ncdiag[<options>]{<Node A>}{<Node B>}
\pcdiag[<options>](<Node A>)(<Node B>)
```

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

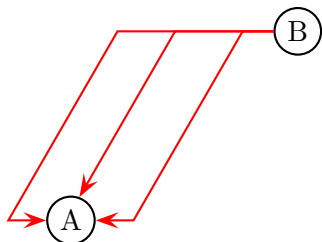


```
1 \begin{pspicture}(5,6)
2   \circnode{A}{A}\quad\circnode{C}{C}%
3   \quad\circnode{E}{E}
4   \rput(0,4){\circnode{B}{B}}
5   \rput(1,5){\circnode{D}{D}}
6   \rput(2,6){\circnode{F}{F}}
7   \psset{arrowscale=2,lineararc=0.2,%
8     linecolor=red,armA=0.5, angleA=90,angleB=-90}
9   \ncdiag[lineAngle=20]{->}{A}{B}
10  \ncput*[nrot=:U]{line I}
11  \ncdiag[lineAngle=20]{->}{C}{D}
12  \ncput*[nrot=:U]{line II}
13  \ncdiag[lineAngle=20]{->}{E}{F}
14  \ncput*[nrot=:U]{line III}
15 \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:



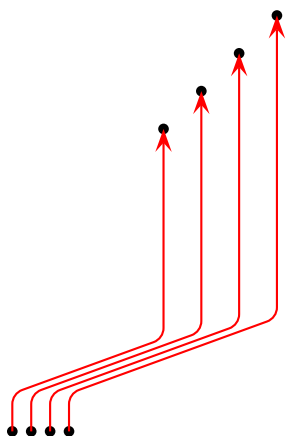
```
1 \begin{pspicture}(4,3)
2   \rput(0.5,0.5){\circnode{A}{A}}
3   \rput(3.5,3){\circnode{B}{B}}
4   {\psset{linecolor=red,arrows=<->,arrowscale=2}
5     \ncdiag[lineAngle=60,%
6       armA=0,angleA=0,angleB=180]{A}{B}
7     \ncdiag[lineAngle=60,%
8       armA=0,angleA=90,angleB=180]{A}{B}
9   }
\end{pspicture}
```



```

1 \begin{pspicture}(4,3)
2   \rput(1,0.5){\circlenode{A}{A}}
3   \rput(4,3){\circlenode{B}{B}}
4   {\psset{linecolor=red,arrows=-,arrowscale=2}
5     \ncdiag[lineAngle=60,%
6       armA=0.5,angleA=0,angleB=180]{A}{B}
7     \ncdiag[lineAngle=60,%
8       armA=0,angleA=70,angleB=180]{A}{B}
9     \ncdiag[lineAngle=60,%
10      armA=0.5,angleA=180,angleB=180]{A}{B}}
11 \end{pspicture}

```



```

1 \begin{pspicture}(4,5.5)
2   \cnode*(0,0){2pt}{A}%
3   \cnode*(0.25,0){2pt}{C}%
4   \cnode*(0.5,0){2pt}{E}%
5   \cnode*(0.75,0){2pt}{G}%
6   \cnode*(2,4){2pt}{B}%
7   \cnode*(2.5,4.5){2pt}{D}%
8   \cnode*(3,5){2pt}{F}%
9   \cnode*(3.5,5.5){2pt}{H}%
10  {\psset{arrowscale=2,linear=0.2,%
11    linecolor=red,armA=0.5, angleA=90,angleB=-90}
12    \pcdiag[lineAngle=20]{->}{A}(B)
13    \pcdiag[lineAngle=20]{->}{C}(D)
14    \pcdiag[lineAngle=20]{->}{E}(F)
15    \pcdiag[lineAngle=20]{->}{G}(H)}
16 \end{pspicture}

```

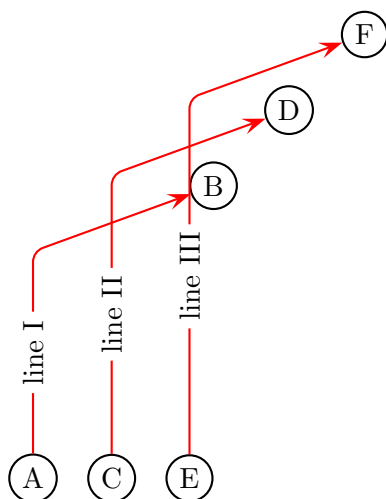
2 \ncdiagg and \pcdiagg

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

```

\ncdiagg[<options>]{<Node A>}{<Node B>}
\pcdiagg[<options>](<Node A>)(<Node B>)

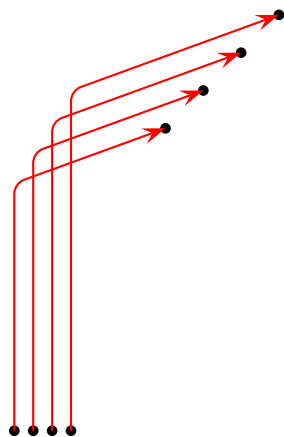
```



```

1 \begin{pspicture}(4,6)
2   \psset{linecolor=black}
3   \circulnode{A}{A}%
4   \quad\circulnode{C}{C}%
5   \quad\circulnode{E}{E}
6   \rput(0,4){\circulnode{B}{B}}
7   \rput(1,5){\circulnode{D}{D}}
8   \rput(2,6){\circulnode{F}{F}}
9   {\psset{arrowscale=2,lineararc=0.2,linecolor=red,arma
10    =0.5, angleA=90}
11   \ncdiagg[lineAngle=-160]{->}{A}{B}
12   \ncput*[nrot=:U]{line I}
13   \ncdiagg[lineAngle=-160]{->}{C}{D}
14   \ncput*[nrot=:U]{line II}
15   \ncdiagg[lineAngle=-160]{->}{E}{F}
16   \ncput*[nrot=:U]{line III}}
17 \end{pspicture}

```

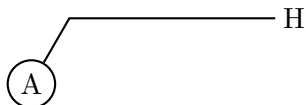


```

1 \begin{pspicture}(4,6)
2   \psset{linecolor=black}
3   \cnode*(0,0){2pt}{A}%
4   \cnode*(0.25,0){2pt}{C}%
5   \cnode*(0.5,0){2pt}{E}%
6   \cnode*(0.75,0){2pt}{G}%
7   \cnode*(2,4){2pt}{B}%
8   \cnode*(2.5,4.5){2pt}{D}%
9   \cnode*(3,5){2pt}{F}%
10  \cnode*(3.5,5.5){2pt}{H}%
11  {\psset{arrowscale=2,lineararc=0.2,linecolor=red,arma
12   =0.5, angleA=90}
13   \pcdiagg[lineAngle=20]{->}{A}{B}
14   \pcdiagg[lineAngle=20]{->}{C}{D}
15   \pcdiagg[lineAngle=20]{->}{E}{F}
16   \pcdiagg[lineAngle=20]{->}{G}{H}}
17 \end{pspicture}

```

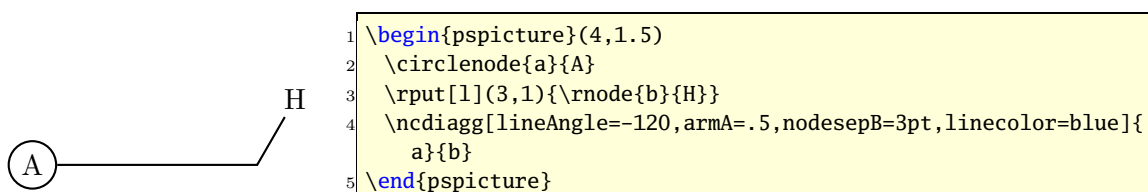
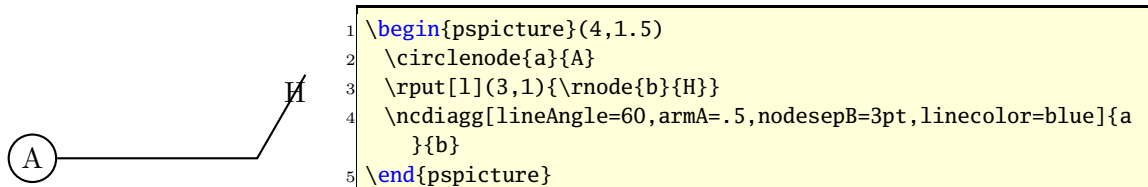
The only catch for `\ncdiagg` is, that you need the right value for `lineAngle`. If the node connection is on the wrong side of the second node, then choose the corresponding angle, e.g.: if 20 is wrong then take -160 , the corresponding to 180.



```

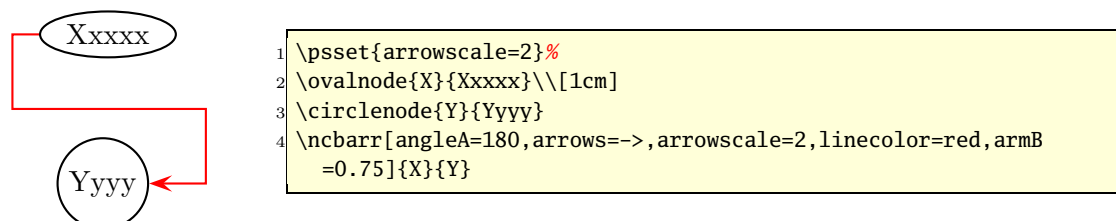
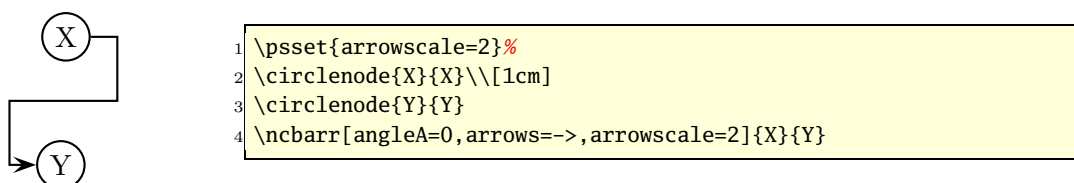
1 \begin{pspicture}(4,1.5)
2   \circulnode{a}{a}
3   \rput[1](3,1){\rnode{b}{H}}
4   \ncdiagg[lineAngle=60,angleA=180,arma=.5,nodesepA=3pt,
5   linecolor=blue]{b}{a}
6 \end{pspicture}

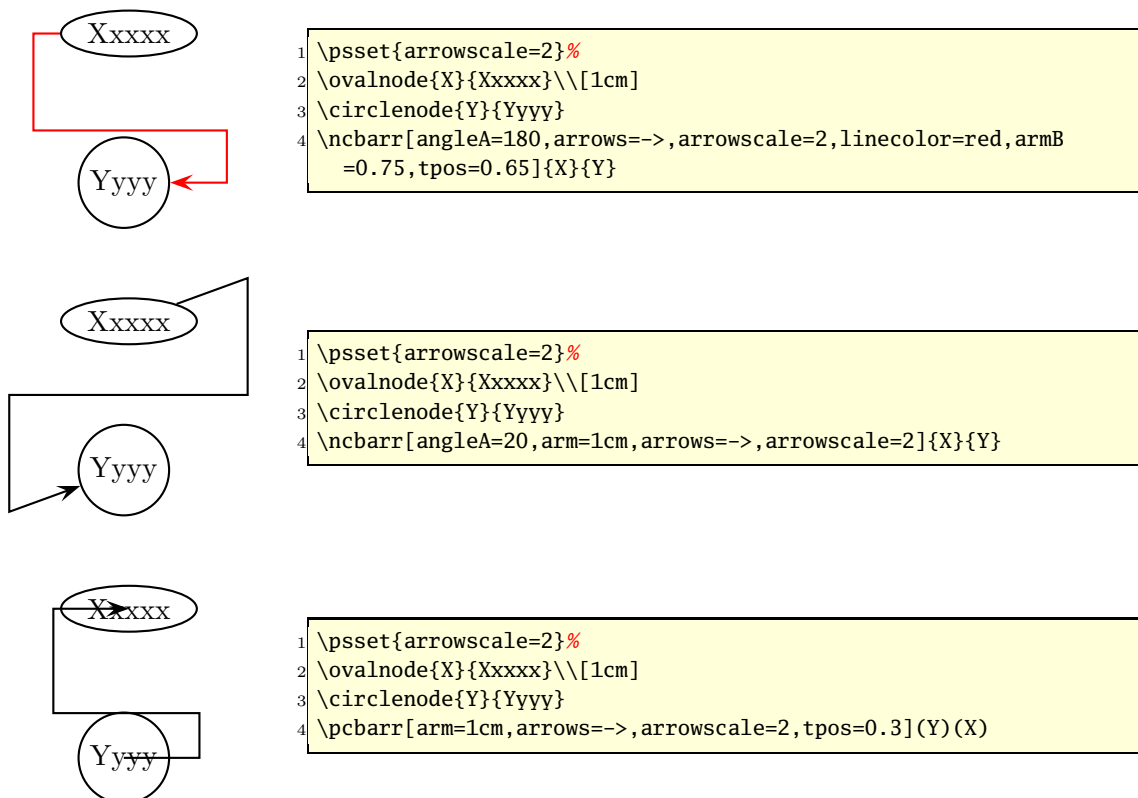
```



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 geenal `AngleA180+` 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` ($0 \leq tpos \leq 1$).





4 `\psRelLine`

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

```

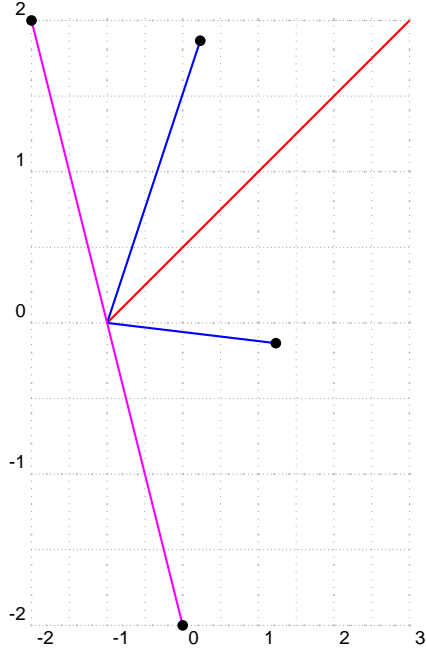
\psRelLine(<P0>)(<P1>){<length factor>}{<end node name>}
\psRelLine[<arrows>](<P0>)(<P1>){<length factor>}{<end node name>}
\psRelLine[<options>](<P0>)(<P1>){<length factor>}{<end node name>}
\psRelLine[<options>]{<arrows>}(<P0>)(<P1>){<length factor>}{<end node name>}

```

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 <code>xunit</code> and <code>yunit</code> .

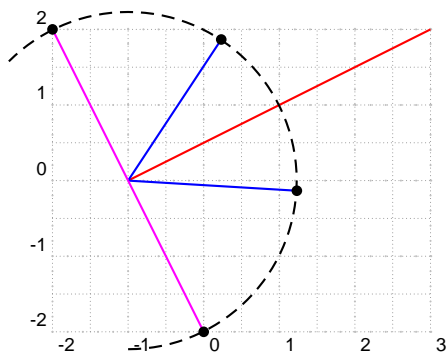
The following two figures show the same, the first one with a scaling different to 1 : 1, this is the reason why the end points are on an ellipse and not on a circle like in the second figure.



```

1 \psset{yunit=2,xunit=1}
2 \begin{pspicture}(-2,-2)(3,2)
3 \psgrid[subgriddiv=2,subgriddots=10,gridcolor=
  lightgray]
4 \pnode(-1,0){A}\pnode(3,2){B}
5 \psline[linecolor=red](A)(B)
6 \psRelLine[linecolor=blue,angle=30](-1,0)(B){0.5}{
  EndNode}
7 \qdisk(EndNode){2pt}
8 \psRelLine[linecolor=blue,angle=-30](A)(B){0.5}{
  EndNode}
9 \qdisk(EndNode){2pt}
10 \psRelLine[linecolor=magenta,angle=90](-1,0)(3,2)
    {0.5}{EndNode}
11 \qdisk(EndNode){2pt}
12 \psRelLine[linecolor=magenta,angle=-90](A)(B){0.5}{
  EndNode}
13 \qdisk(EndNode){2pt}
14 \end{pspicture}

```

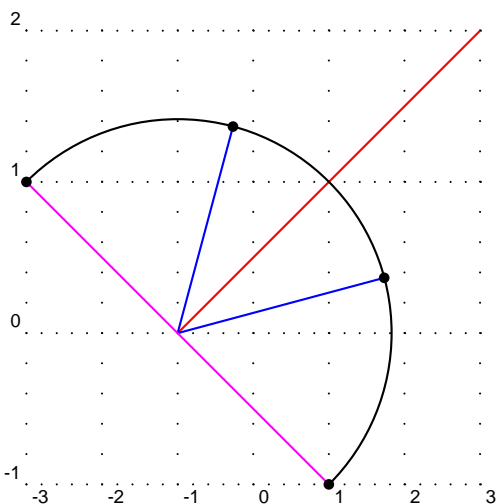



```

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

```

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

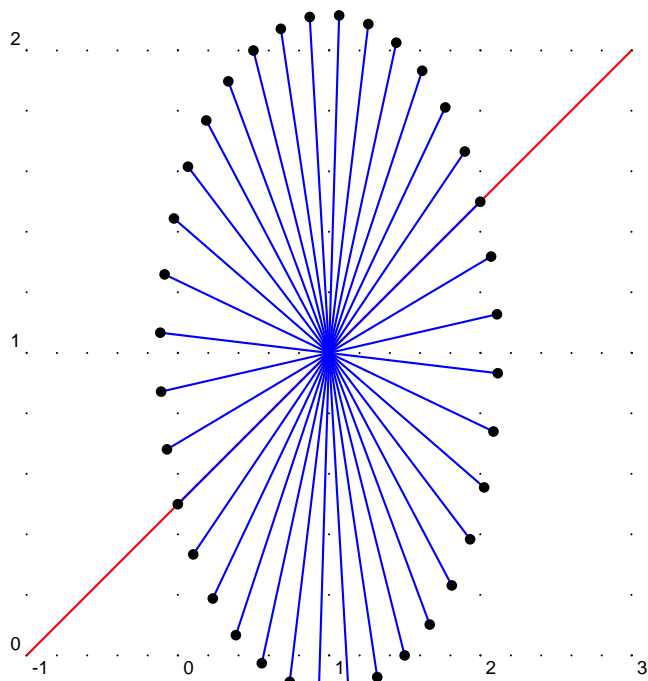


```

1 \psset{yunit=2,xunit=1}
2 \begin{pspicture}(-3,-1)(3,2)\psgrid[
   subgridcolor=lightgray]
3 \pnode(-1,0){A}\pnode(3,2){B}
4 \psline[linecolor=red](A)(B)
5 \psarc(A){2.83}{-45}{135}
6 \psRelLine[linecolor=blue,angle=30,trueAngle](A
   )(B){0.5}{EndNode}
7 \qdisk(EndNode){2pt}
8 \psRelLine[linecolor=blue,angle=-30,trueAngle](
   A)(B){0.5}{EndNode}
9 \qdisk(EndNode){2pt}
10 \psRelLine[linecolor=magenta,angle=90,trueAngle
   ](A)(B){0.5}{EndNode}
11 \qdisk(EndNode){2pt}
12 \psRelLine[linecolor=magenta,angle=-90,
   trueAngle](A)(B){0.5}{EndNode}
13 \qdisk(EndNode){2pt}
14 \end{pspicture}

```

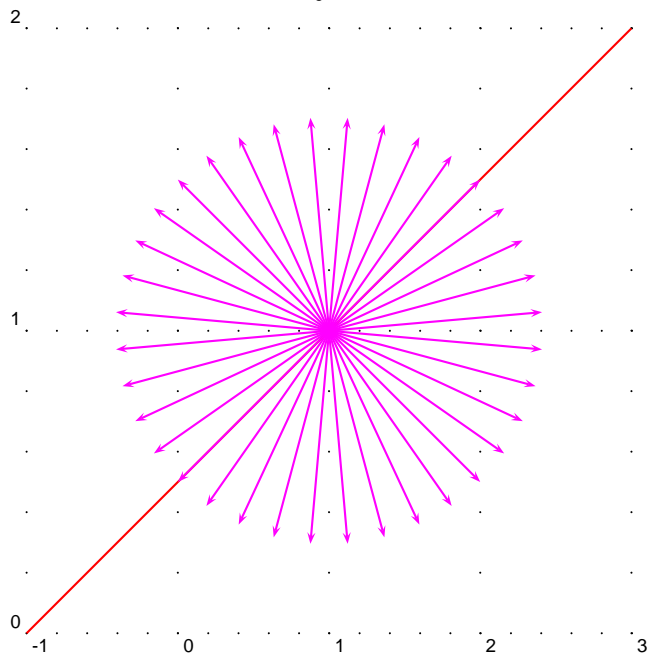
Two examples with using `\multido` to show the behaviour of the options `trueAngle` and `angle`.



```

1 \psset{yunit=4,xunit=2}
2 \begin{pspicture}(-1,0)(3,2)\psgrid[
   subgridcolor=lightgray]
3 \pnode(-1,0){A}\pnode(1,1){B}
4 \psline[linecolor=red](A)(3,2)
5 \multido{\iA=0+10}{36}{%
6   \psRelLine[linecolor=blue,angle=\iA
7     ](B)(A){-0.5}{EndNode}
8   \qdisk(EndNode){2pt}
9 }
\end{pspicture}

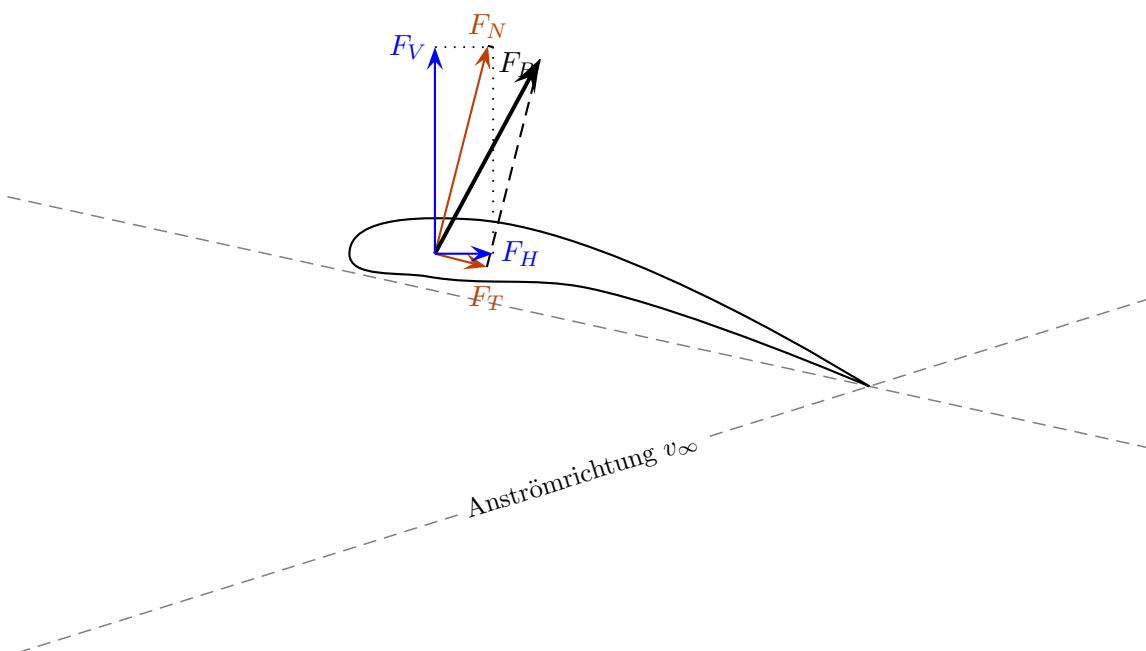
```



```

1 \psset{yunit=4,xunit=2}
2 \begin{pspicture}(-1,0)(3,2)\psgrid[
   subgridcolor=lightgray]
3 \pnode(-1,0){A}\pnode(1,1){B}
4 \psline[linecolor=red](A)(3,2)
5 \multido{\iA=0+10}{36}{%
6   \psRelLine[linecolor=magenta,angle=\
7     iA,trueAngle]{->}(B)(A){-0.5}{
8     EndNode}
9 }
\end{pspicture}

```



```

1 \psset{xunit=\linewidth,yunit=\linewidth,trueAngle}%
2 \begin{pspicture}(1,0.6)%\psgrid
3   \pnode(.3,.35){Vk} \pnode(.375,.35){D}
4   \pnode(0,.4){DST1} \pnode(1,.18){DST2}
5   \pnode(0,.1){A1} \pnode(1,.31){A1}
6   { \psset{linewidth=.02,linestyle=dashed,linecolor=gray}%
7     \pcline(DST1)(DST2) % <- Druckseitentangente
8     \pcline(A2)(A1) % <- Anströmrichtung
9     \lput*{:U}{\small Anströmrichtung $v_{\infty}$}}
10  }%
11  \psIntersectionPoint(A1)(A2)(DST1)(DST2){Hk}
12  \pscurve(Hk)(.4,.38)(Vk)(.36,.33)(.5,.32)(Hk)
13  \psParallelLine[linecolor=red!75!green,arrows=->,arrowscale=2](Vk)(Hk)(D){.1}{FtE}
14  \psRelLine[linecolor=red!75!green,arrows=->,%
15    arrowscale=2,angle=90](D)(FtE){4}{Fn}% why "4"?
16  \psParallelLine[linestyle=dashed](D)(FtE)(Fn){.1}{Fnr1}
17  \psRelLine[linestyle=dashed,angle=90](FtE)(D){-4}{Fnr2} % why "-4"?
18  \psline[linewidth=1.5pt,arrows=->,arrowscale=2](D)(Fnr2)
19  \psIntersectionPoint(D)([nodesep=2]D)(Fnr1)([offset=-4]Fnr1){Fh}
20  \psIntersectionPoint(D)([offset=2]D)(Fnr1)([nodesep=4]Fnr1){Fv}
21  \psline[linecolor=blue,arrows=->,arrowscale=2](D)(Fh)
22  \psline[linecolor=blue,arrows=->,arrowscale=2](D)(Fv)
23  \psline[linestyle=dotted](Fh)(Fnr1)
24  \psline[linestyle=dotted](Fv)(Fnr1)
25  \uput{.1}[0](Fh){\blue $F_{\text{H}}$}
26  \uput{.1}[180](Fv){\blue $F_{\text{V}}$}
27  \uput{.1}[-45](Fnr1){$F_{\text{R}}$}
28  \uput{.1}[90](Fn){\color{red!75!green}$F_{\text{N}}$}
29  \uput{.25}[-90](FtE){\color{red!75!green}$F_{\text{T}}$}
30 \end{pspicture}

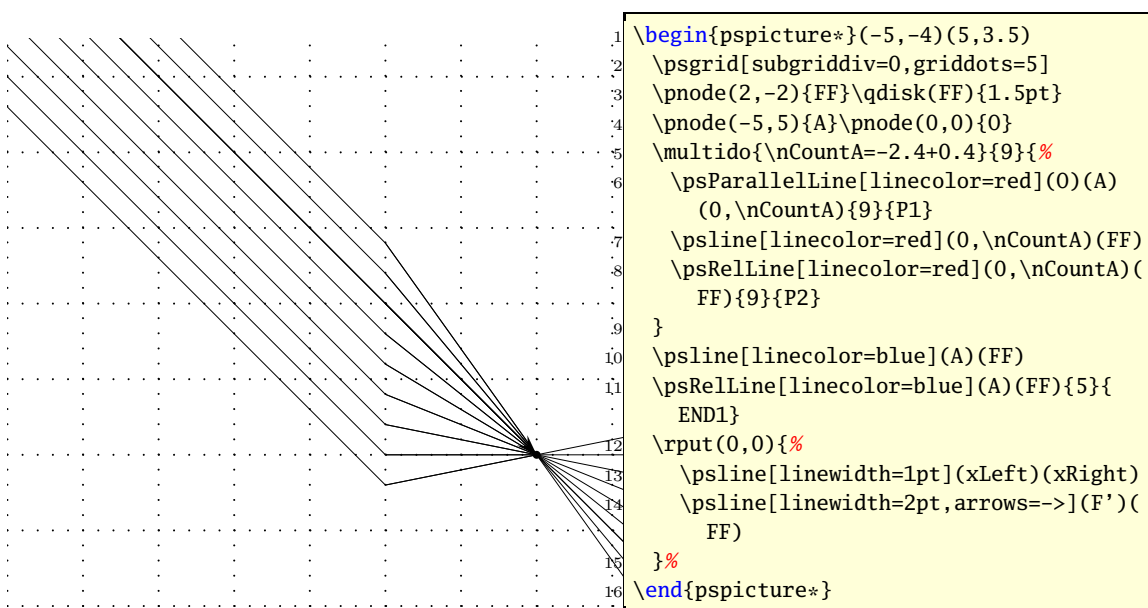
```

5 `\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>}
```

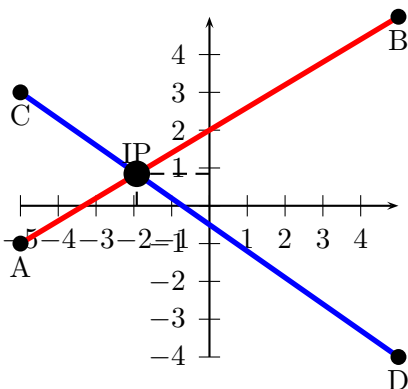
The line starts at P_2 , is parallel to $\overline{P_0P_1}$ and the length of this parallel line depends to the length factor. The end node name must be a valid nodename and shouldn't contain any of the special PostScript characters.



6 `\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>}
```



```

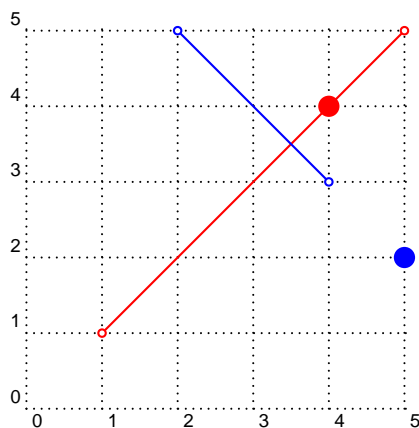
1 \psset{unit=0.5cm}
2 \begin{pspicture}(-5,-4)(5,5)
3   \psaxes{->}(0,0)(-5,-4)(5,5)
4   \psline[linecolor=red,linewidth=2pt](-5,-1)(5,5)
5   \psline[linecolor=blue,linewidth=2pt](-5,3)(5,-4)
6   \qdisk(-5,-1){3pt}\uput[-90](-5,-1){A}
7   \qdisk(5,5){3pt}\uput[-90](5,5){B}
8   \qdisk(-5,3){3pt}\uput[-90](-5,3){C}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

```

7 `\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}`



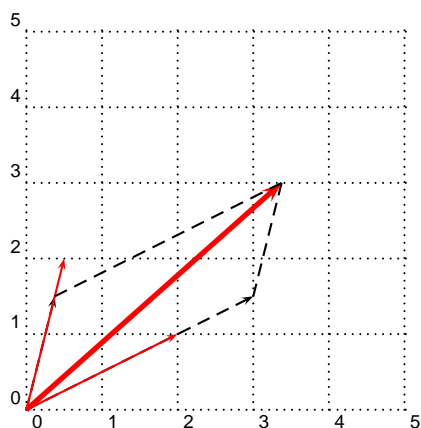
```

1 \begin{pspicture}(5,5)
2   \psgrid[subgriddiv=0,griddots=10]
3   \psset{linecolor=red}
4   \psline{o-o}(1,1)(5,5)
5   \setLNode(1,1)(5,5){0.75}{PI}
6   \qdisk(PI){4pt}
7   \psset{linecolor=blue}
8   \psline{o-o}(4,3)(2,5)
9   \setLNode(4,3)(2,5){-0.5}{PII}
10  \qdisk(PII){4pt}
11 \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}`

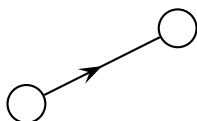


```

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

```

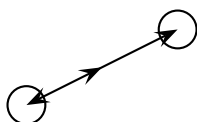
8 \ncline, \pcline and inside errors



```

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

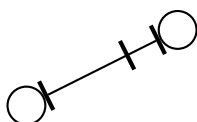
```



```

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

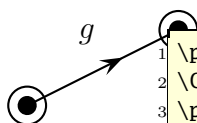
```



```

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

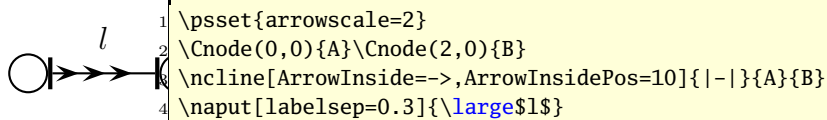
```



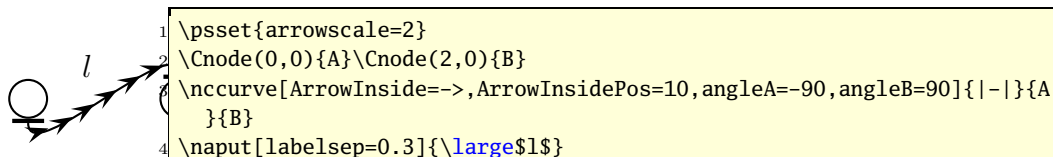
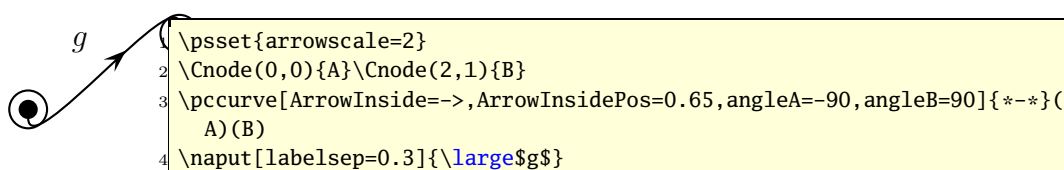
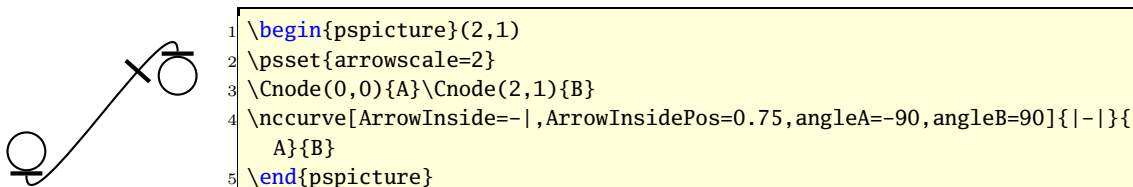
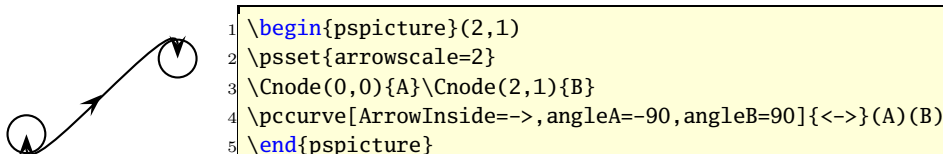
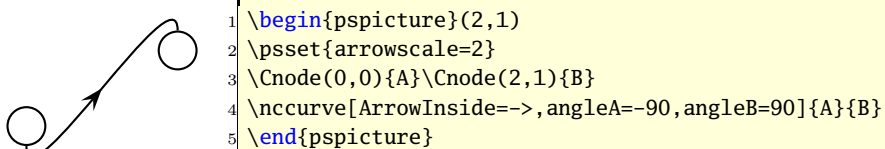
```

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

```



9 \nccurve, \pccurve and inside errors



10 `\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`.

```
1 \def\resetPSTNodeOptions{%
2 \psset[pst-node]{%
3   nodealign=false,%
4   href=0,%
5   vref=.7ex,%
6   framesize=10pt,%
7   nodesep=0pt,%
8   arm=10pt,%
9   offset=0pt,%
10  angle=0,%
11  arcangle=8,%
12  ncurv=.67,%
13  loopsize=1cm,%
14  boxsize=.4cm,%
15  nrot=0,%
16  npos=,%
17  tpos=0.5,%
18  shortput=none,%
19  colsep=1.5cm,%
20  rowsep=1.5cm,%
21  % shortput=tablr,%%
22  mcol=c,%
23  mnode=R,%
24  emnode=none}
25 }
```

11 Credits