pstricks-add
additionals Macros for pstricks*

v.1.2

Herbert Voß

February 29, 2004

Contents

I pstricks 4

1 New options for pspicture environment 4

2 Numeric functions 4

3 Fill style asolid 5

4 \pslineII Colored lines 5
- 4.1 The options . . . 6
- 4.2 Examples . . . 6

5 \pslineIII Variable linewidth 8
- 5.1 The options . . . 8
- 5.2 Examples . . . 8

6 \psbrace 9
- 6.1 Syntax . . . 9
- 6.2 Options . . . 10
- 6.3 Examples . . . 10

7 \psellipse 12
- 7.1 Ellipse based on pst-plot . . . 12
  - 7.1.1 Wedge of a Ellipse . . . 13
- 7.2 New Ellipse Macros . . . 14

*This document was written with Kile: 1.6a (Qt: 3.1.1; KDE: 3.1.1; http://sourceforge.net/projects/kile/) and the PDF output was build with VTeX/Free (http://www.micropress-inc.com/linux)

7.2.1 Arc of an Ellipse . . . 14
7.3 Arc of an Ellipse with anti clockwise direction . . . 14
7.3.1 Wedge of an Ellipse . . . 15

8 Arrows **15**
8.1 Definition . . . 15
8.2 `ArrowInside` Option . . . 16
8.3 `ArrowFill` Option . . . 17
8.4 Examples . . . 18
8.4.1 `\psline` . . . 18
8.4.2 `\pspolygon` . . . 20
8.4.3 `\psbezier` . . . 21
8.4.4 `\pcline` . . . 23
8.4.5 `\pccurve` . . . 23

9 `\psFormatInt` **23**

II `pst-node` **25**

10 `\nclineII` **25**
10.1 The options . . . 25
10.2 Examples . . . 25

11 `\pclineII` **26**

12 `\ncdiag` and `\pcdiag` **26**

13 `\ncdiagg` and `\pcdiagg` **28**

III `pst-plot` **30**

14 New macro `\resetOptions` **30**

15 New options `xyAxes`, `xAxes` and `yAxes` **30**

16 New options `xyLabel`, `xLabel` and `yLabel` **31**

17 New options `xyDecimals`, `xDecimals` and `yDecimals` **31**

18 New option `comma` **32**

19 New options `logBase`, `xlogBase` and `ylogBase` **33**
19.1 y axis (`ylogBase`) . . . 33
19.2 x axis (`xlogBase`) . . . 34
19.3 All axes (`logBase`) . . . 36

19.4 No logstyle (logBase={}) . . . 36
19.5 More examples for the logBase option . . . 37

20 New option logLines **39**

21 New option for \readdata **41**

22 New options for \listplot **42**
22.1 Example for nStep/xStep . . . 43
22.2 Example for nStart/xStart . . . 44
22.3 Example for nEnd/xEnd . . . 45
22.4 Example for all new options . . . 46
22.5 Example for xStart . . . 47
22.6 Example for xStart . . . 48
22.7 Example for plotNo/plotNoMax . . . 48

23 New macro psplotPolynomial **50**

24 Credits **50**

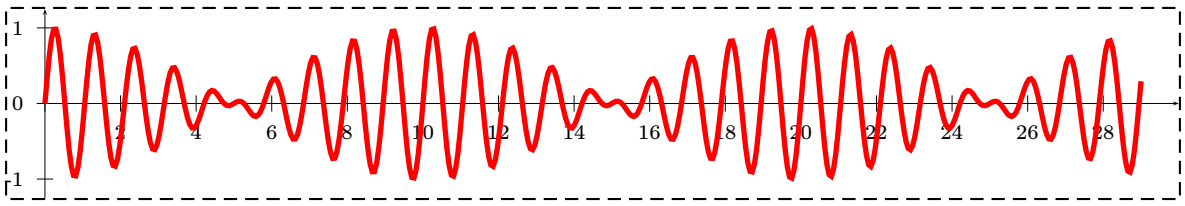
25 The code **52**

Part I
pstricks

1 New options for pspicture environment

Centering a PSTricks objects depends to the right horizontal width, defined with the default environment \begin{pspicture}(...)(...). This is sometimes not easy to find the right values. With the option frame=true it is possible to draw a frame around the defined rectangular. There is no possible fboxsep option here. If you need a "real" frame, then use \psframebox insread.

Name	Default	Meaning
frame	false	draw a frame around the defined pspicture environment
frameStyle	dashed	this is passed to the linestyle option




```
\begin{pspicture}[frame=true](-0.5,-1.25)(15,1.25)% <---!!!!
\psaxes[xunit=.5, Dx=2,linewidth=0.1pt,xyLabel=\scriptsize]{->}(0,0)(0,-1.25)(30,1.25)
\psplot[xunit=.5,yunit=0.5,plotpoints=500,%
 linecolor=red,linewidth=2pt]{0}{29}{x 360 mul sin x 0.9 mul 360 mul sin add}
\end{pspicture}
```

It is also possible to set this frame option globally for all pspicture environments in the usual way with \psset{frame=true}.

2 Numeric functions

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

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

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

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

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

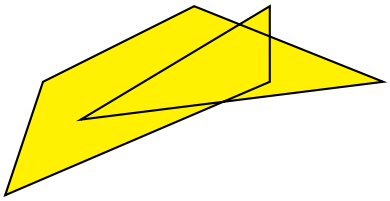
```
\pst@mod{<integer>}{<integer>}{<result as a macro>}

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

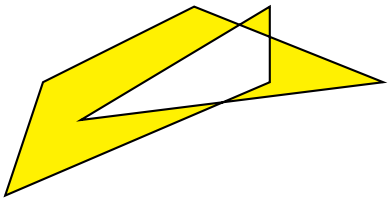
this gives the output 4. Using this internal numeric functions in documents require a setting inside the `makeatletter` and `makeatother` environment. It makes some sense to define a new macroname in the preamble to use it without, e.g. `\let\modulo\pst@mod`.

3 Fill style `asolid`

PostScript has a special fillstyle, called `eofill`, which is available with `pstricks-add` with the option `fillstyle=asolid`. The following two images show the difference, the first one is filled with `fillstyle=solid` and the second one with the new option `fillstyle=asolid`.



```
\pspolygon[unit=0.5cm,fillstyle=solid,%
    fillcolor=yellow]%
    (7,3)(0,0)(1,3)(5,5)(10,3)(2,2)(7,5)(7,3)
```



```
\pspolygon[unit=0.5cm,fillstyle=asolid,%
    fillcolor=yellow]%
    (7,3)(0,0)(1,3)(5,5)(10,3)(2,2)(7,5)(7,3)
```

4 `\pslineII` Colored lines

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

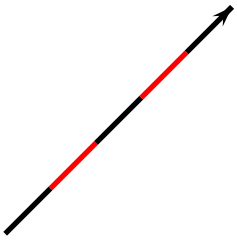
```
1 \pslineII[linewidth=5pt,arrowscale=2]{o-o}(0,0)(12,0)
```

4.1 The options

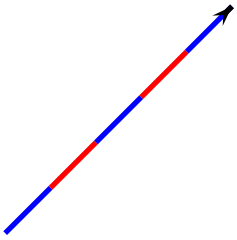
name	meaning
`dashColorI`	first color, default is `black`
`dashColorII`	second color, default is `red`
`dashNo`	the difference in per cent of the colored lines, default ist 0.2
`linecap`	how two lines are connected. 0: no modification 1: rounded edges 2: an additional half square at both ends

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

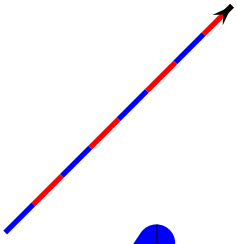
4.2 Examples



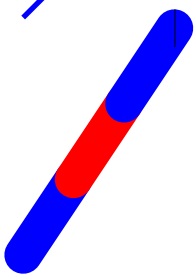
```
\pslineII{->}(0,0)(3,3)
```



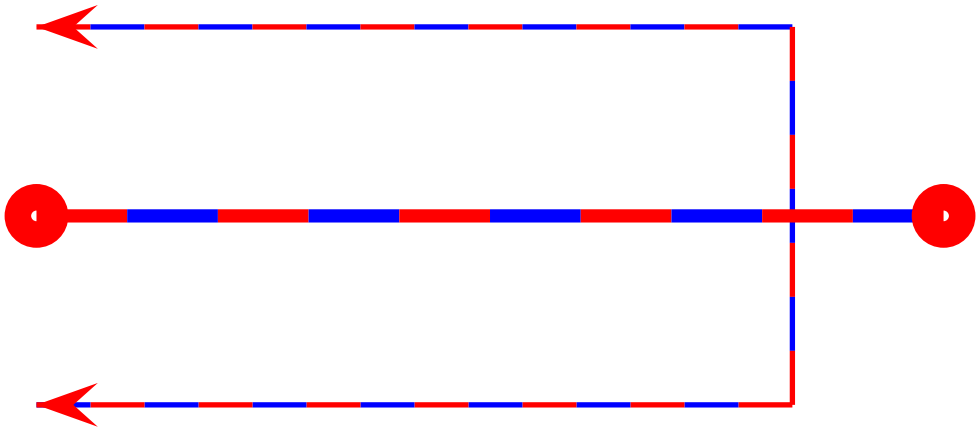
```
\pslineII[dashColorI=blue]{->}(0,0)(3,3)
```



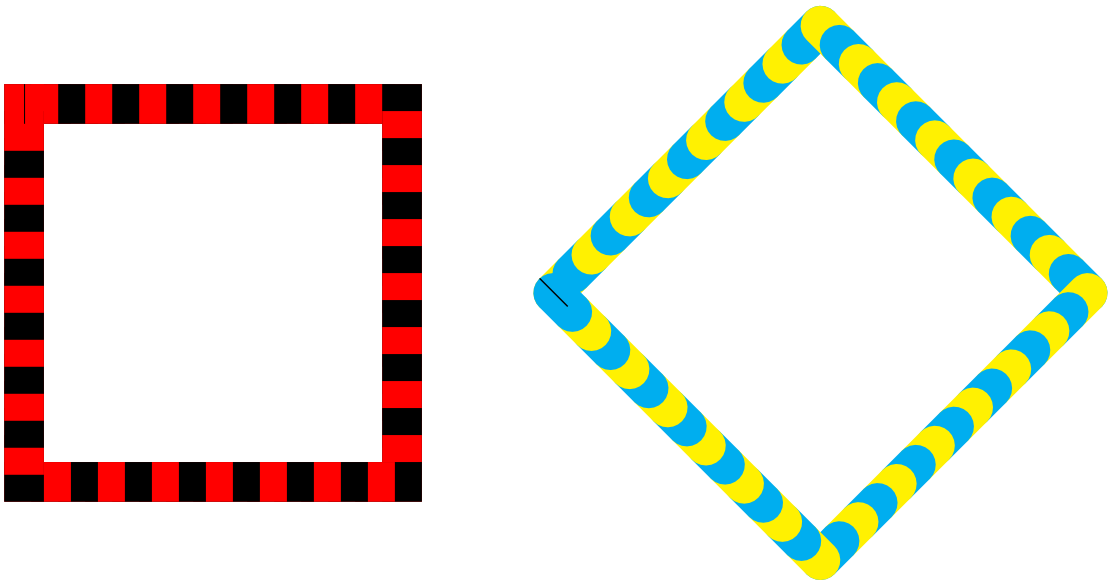
```
\pslineII[dashColorI=blue,dashNo=15]{->}(0,0)(3,3)
```



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



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



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

5 \pslineIII Variable linewidth

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

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

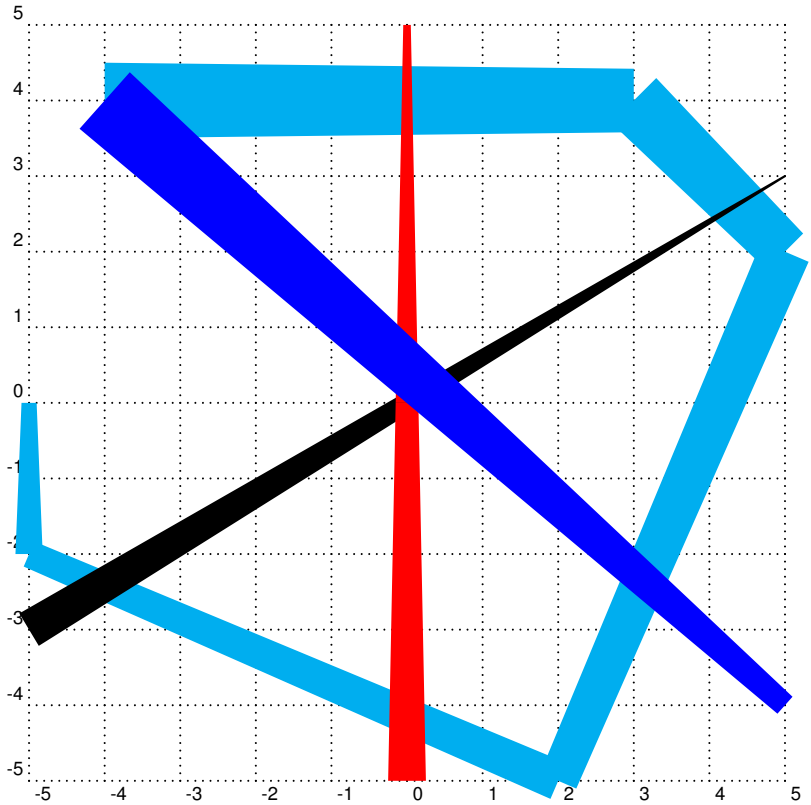
5.1 The options

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

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

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

5.2 Examples

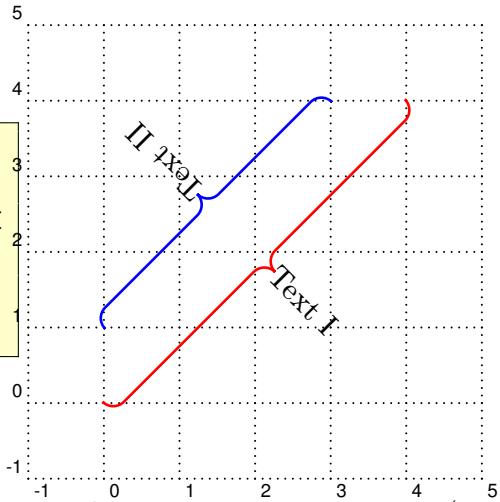


6 \psbrace

6.1 Syntax

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

```
\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}
```



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

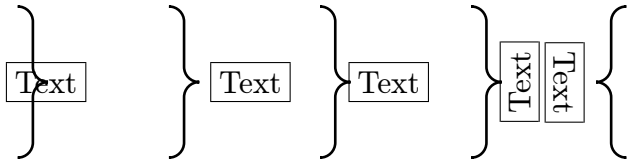
6.2 Options

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

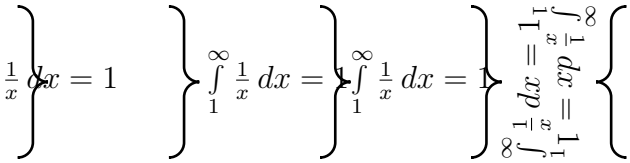
name	meaning
`braceWidth`	default is 0.35
`bracePos`	relative position (default is 0.5)
`nodesepA`	x-separation (default is $0pt$)
`nodesepB`	y-separation (default is $0pt$)
`rot`	additional rotating for the text (default is 0)
`ref`	reference point for the text (default is c)

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

6.3 Examples




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



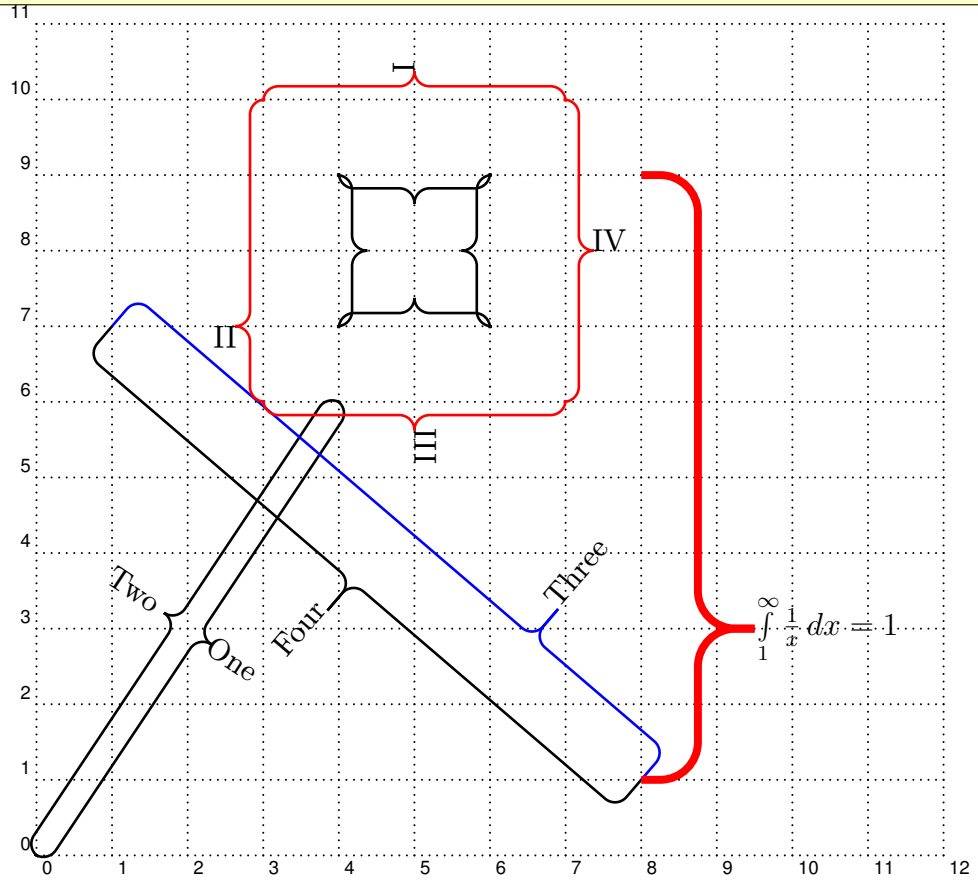

```
\def\someMath{$\int\limits_1^{\infty}\frac{1}{x}\,dx=1$}
\psbrace(0,0)(0,2){\someMath}%
\psbrace[nodesepA=30pt](2,0)(2,2){\someMath}
\psbrace[ref=lC](4,0)(4,2){\someMath}
\psbrace[ref=lt,rot=90,nodesepB=-30pt](6,0)(6,2){\someMath}
\psbrace[ref=lt,rot=90,nodesepB=30pt](8,2)(8,0){\someMath}
```

some very, very long wonderful Text

Text

Text

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




```
\begin{pspicture}(12,11)
\psgrid[subgriddiv=0,griddots=10]
\pnode(0,0){A}
\pnode(4,6){B}
\psbrace[ref=lC](A)(B){One}
```

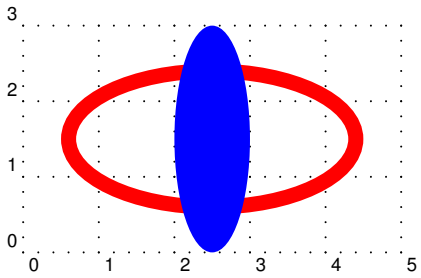
```
\psbrace[rot=180,nodesepA=-5pt,ref=rb](B)(A){Two}
\psbrace[linecolor=blue,bracePos=0.25,%
 braceWidth=1,ref=lB](8,1)(1,7){Three}
\psbrace[braceWidth=-1,rot=180,ref=rB](8,1)(1,7){Four}
\psbrace[linearc=0.5,linecolor=red,linewidth=3pt,%
 braceWidth=1.5,bracePos=0.25,ref=lC](8,1)(8,9){\someMath}
\psbrace(4,9)(6,9){}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

7 \psellipse

pstricks has only the following macro for drawing an ellipse:

```
\psellipse*[<option>](x,y)(a,b)
```

whith (x,y) as the center and (a,b) as the two radians (figure 1).




```
\begin{pspicture}(0,-0.25)(5,3.25)
 \psgrid
 \psellipse[%
  linewidth=0.2,%
  linecolor=red](2.5,1.5)(2,1)
 \psellipse*[%
  linecolor=blue](2.5,1.5)(0.5,1.5)
\end{pspicture}%
```

Figure 1: The pstricks macro \psellipse

7.1 Ellipse based on pst-plot

With the \parametricplot macro from pst-plot [9] we can define a new macro for drawing ellipses:

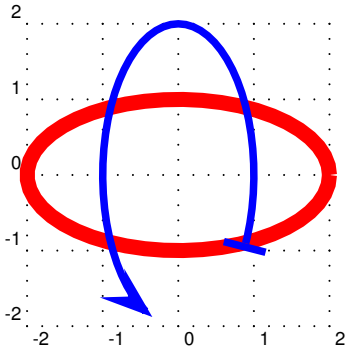
```
% #1 options
% #2 a
% #3 b
% #4 start angle
```

```
% #5 end angle
\newcommand{\pstEllipse}[5][]{%
 \psset{#1}
 \parametricplot{#4}{#5}{#2\space t cos mul #3\space t sin mul}}
```

which has the syntax

```
\pstEllipse[<options>]{a}{b}{start angle}{end angle}
```

This macro is not part of of `pstricks-add`.




```
\begin{pspicture}(-2.25,-1.75)(2.25,1.75)
 \psgrid
 \pstEllipse[%
  linewidth=0.2,%
  linecolor=red]{2}{1}{0}{360}
 \pstEllipse[%
  linewidth=0.1,%
  arrows=|->,%
  arrowsize=0.5,%
  linecolor=blue]{1}{2}{-30}{250}
\end{pspicture}%
```

Figure 2: The macro `\pstEllipse` which uses the `\parametricplot` macro from `pst-plot`

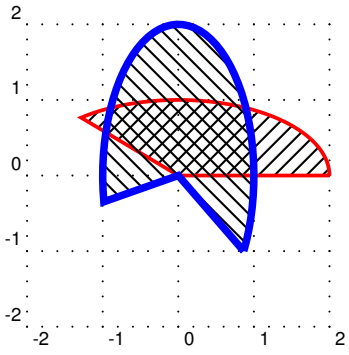
As seen in figure 2 it is no problem to draw arcs of an ellipse. The center of these ellipses are by default $(0, 0)$, with the `\rput` macro it is not a problem to put the ellipse anywhere in the coordinate system.

7.1.1 Wedge of a Ellipse

There exists also a macro for a wedge of an ellipse (figure 3) which uses the following code:

```
% #1 options
% #2 a
% #3 b
% #4 start angle
% #5 end angle
\newcommand{\pstEllipseWedge}[5][]{%
 \psset{#1}
 \pscustom{%
  \parametricplot{#4}{#5}{#2\space t cos mul #3\space t sin mul}%
  \psline(! #2\space #5\space cos mul #3\space #5\space sin mul)%
   (0,0)%
   (! #2\space #4\space cos mul #3\space #4\space sin mul)%
 }%
}
```

This macro is also not part of of `pstricks-add`.




```
\begin{pspicture}(-2.25,-1.75)(2.25,1.75)
 \psgrid
 \pstEllipseWedge[%
  linewidth=0.05,%
  linecolor=red,%
  fillstyle=hlines,%
  fillcolor=red]{2}{1}{0}{130}
 \pstEllipseWedge[%
  linewidth=0.1,%
  linecolor=blue,%
  fillstyle=vlines,%
  fillcolor=blue]{1}{2}{-30}{190}
\end{pspicture}%
```

Figure 3: The macro `\pstEllipseWedge` which uses the `\parametricplot` macro from `pst-plot`

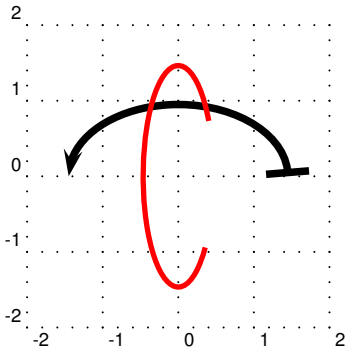
7.2 New Ellipse Macros

All macros defined in this package are original from Timothy Van Zandt and modified by several authors. The available macros are

```
\psEllipticArc[<options>]
    {<arrows>}(<center>)(a,b){start angle}{end angle}
\psEllipticArcN[<options>]
    {<arrows>}(<center>)(a,b){start angle}{end angle}
\psWedgeEllipse[<options>]
    {<arrows>}(<center>)(a,b){start angle}{end angle}
```

7.2.1 Arc of an Ellipse

Figure 4 shows different examples for this macro.

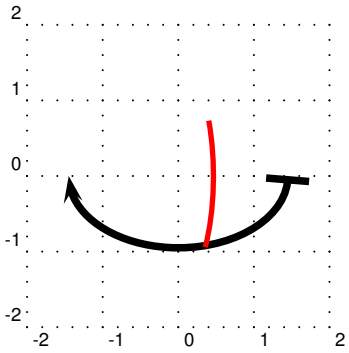



```
\begin{pspicture}(-2.25,-2.25)(2.25,2.25)
 \psEllipticArc[linewidth=0.1]{|->}(0,0)
     (1.5,1){0}{180}
 \psEllipticArc[linecolor=red](0,0)
     (0.5,1.5){30}{320}
\end{pspicture}
```

Figure 4: The macro `\psEllipticArc` from `pst-ellipse`

7.3 Arc of an Ellipse with anti clockwise direction

Figure 5 shows different examples for this macro which is the same than the one figure **??** only drawn anti clockwise.

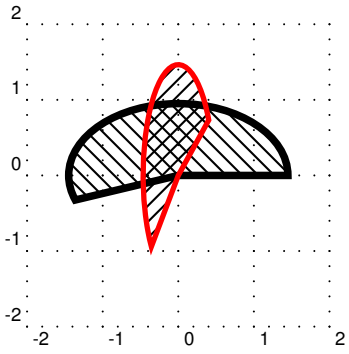



```
\begin{pspicture}(-2.25,-2.25)(2.25,2.25)
 \psEllipticArcN[linewidth=0.1]{|->}(0,0)
    (1.5,1){0}{180}
 \psEllipticArcN[linecolor=red](0,0)
    (0.5,1.5){30}{320}
\end{pspicture}
```

Figure 5: The macro `\psEllipticArcN` from `pst-ellipse`

7.3.1 Wedge of an Ellipse

Figure 6 shows different examples for this macro.




```
\begin{pspicture}(-2.25,-2.25)(2.25,2.25)
\begin{pspicture}(-2.25,-2.25)(2.25,2.25)
 \psgrid
 \psWedgeEllipse[%
  fillstyle=vlines,%
  linewidth=0.1](0,0)(1.5,1){0}{200}
 \psWedgeEllipse[%
  fillstyle=hlines,%
  linecolor=red](0,0)(0.5,1.5){30}{220}
\end{pspicture}
```

Figure 6: The macro `\psWedgeEllipse` from `pst-ellipse`

8 Arrows

8.1 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.

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)
```

8.2 `ArrowInside` Option

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

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

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

8.3 `ArrowFill` Option

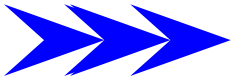
By default all arrows are filled polygons. With the option `ArrowFill=false` there are ”white“ arrows. Only for the begiining/end arrows they are empty, the inside arrows are overpainted with the line.

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

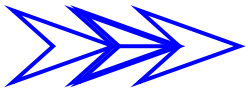
```
\psline[%
   linecolor=red,%
   ArrowFill=false,%
   arrowinset=0]{<->}(0,0)(3,0)
```

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

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

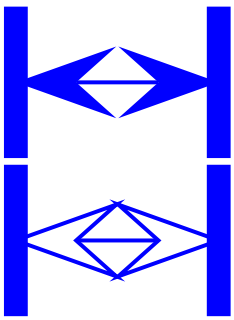


```
\psline[%
    linecolor=blue,%
    arrowscale=6,%
    ArrowFill=false]{<->}(0,0)(3,0)
```



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

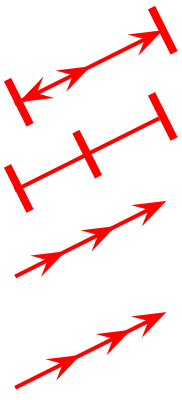
```
\psline[%
    linecolor=blue,%
    arrowscale=6,%
    ArrowFill=false]{>|->|}(0,0)(3,0)
```



8.4 Examples

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

8.4.1 \psline

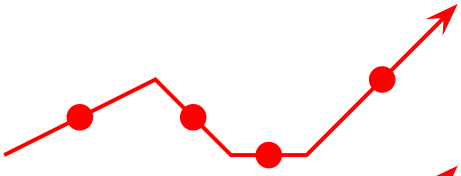


```
\psline[ArrowInside=->]{|<->|}(2,1)
```

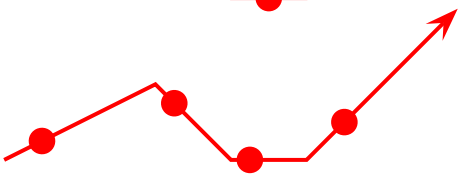
```
\psline[ArrowInside=-|]{|-|}(0,0)(2,1)
```

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

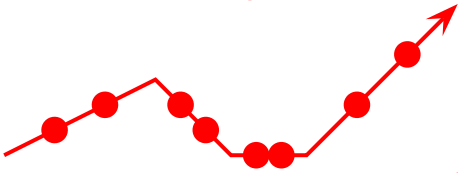
```
\psline[ArrowInside=->,%
   ArrowInsideNo=2,%
   ArrowInsideOffset=0.1]{->}(0,0)(2,1)
```



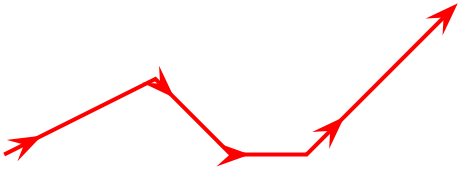
```
\psline[ArrowInside=-*]{->}%
   (0,0)(2,1)(3,0)(4,0)(6,2)
```



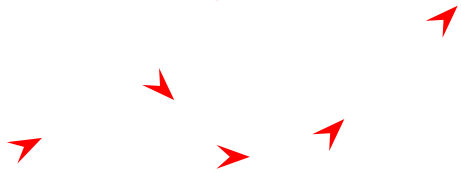
```
\psline[ArrowInside=-*,%
   ArrowInsidePos=0.25]{->}%
      (0,0)(2,1)(3,0)(4,0)(6,2)
```



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

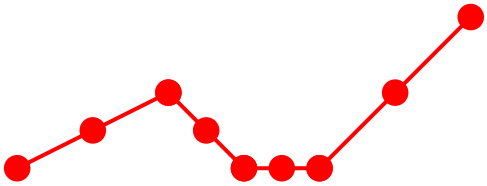


```
\psline[ArrowInside=->,%
   ArrowInsidePos=0.25]{->}%
      (0,0)(2,1)(3,0)(4,0)(6,2)
```

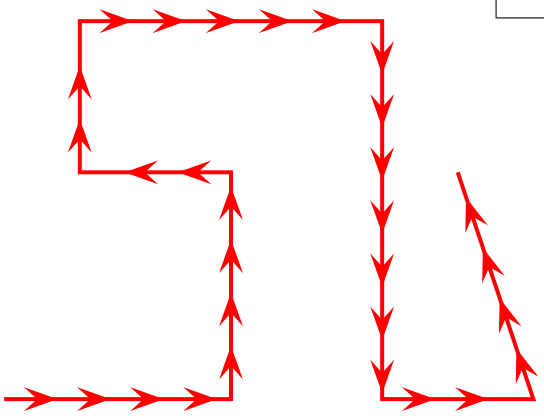


```
\psline[linestyle=none,%
   ArrowInside=->,%
   ArrowInsidePos=0.25]{->}%
      (0,0)(2,1)(3,0)(4,0)(6,2)
```

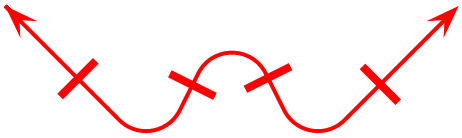
```
\psline[ArrowInside=-<,%
   ArrowInsidePos=0.75]{->}%
      (0,0)(2,1)(3,0)(4,0)(6,2)
```



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

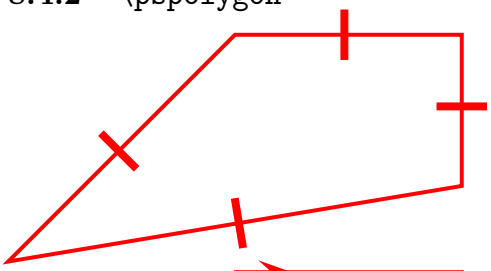


```
\psline[ArrowInside=->,%
   ArrowInsidePos=20](0,0)(3,0)%
      (3,3)(1,3)(1,5)(5,5)%
      (5,0)(7,0)(6,3)
```

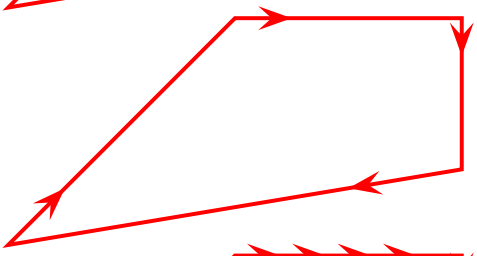


```
\psline[linearc=0.5,%
    ArrowInside=-|]{<->}%
    (0,2)(2,0)(3,2)(4,0)(6,2)
```

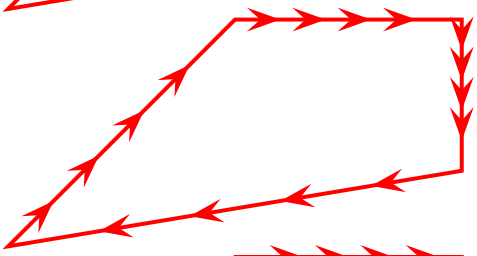
8.4.2 \pspolygon



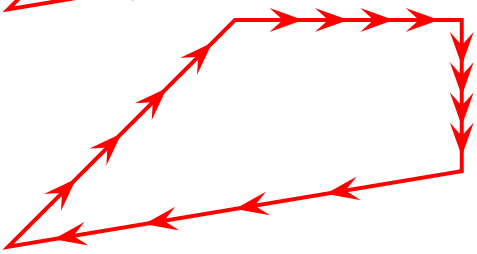
```
\pspolygon[ArrowInside=-|]%
    (0,0)(3,3)(6,3)(6,1)
```



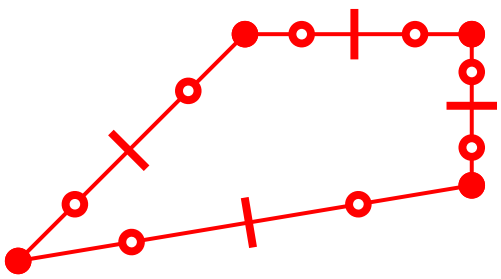
```
\pspolygon[ArrowInside=->,%
   ArrowInsidePos=0.25]%
     (0,0)(3,3)(6,3)(6,1)
```



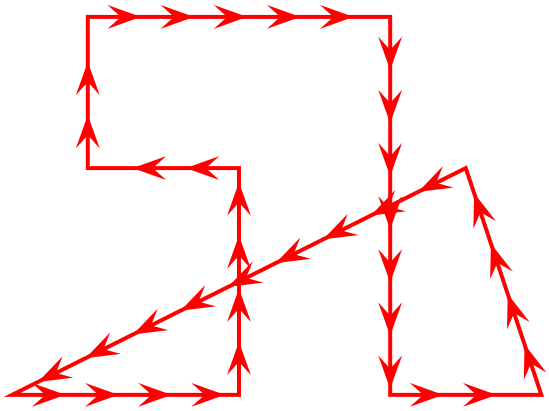
```
\pspolygon[ArrowInside=->,%
   ArrowInsideNo=4]%
     (0,0)(3,3)(6,3)(6,1)
```



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

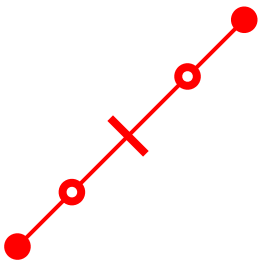


```
\pspolygon[ArrowInside=-|]%
   (0,0)(3,3)(6,3)(6,1)
\psset{linestyle=none,ArrowInside=-*}
\pspolygon[ArrowInsidePos=0]%
   (0,0)(3,3)(6,3)(6,1)
\pspolygon[ArrowInsidePos=1]%
   (0,0)(3,3)(6,3)(6,1)
\psset{ArrowInside=-o}
\pspolygon[ArrowInsidePos=0.25]%
   (0,0)(3,3)(6,3)(6,1)
\pspolygon[ArrowInsidePos=0.75]%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

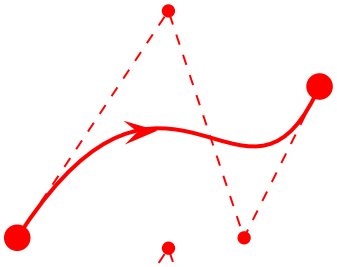


```
\pspolygon[ArrowInside=->,%
    ArrowInsidePos=20](0,0)(3,0)%
      (3,3)(1,3)(1,5)(5,5)%
      (5,0)(7,0)(6,3)
```

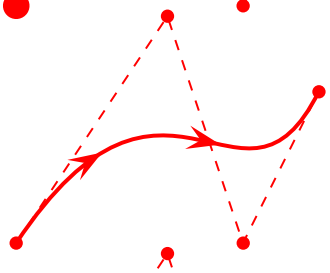
8.4.3 \psbezier



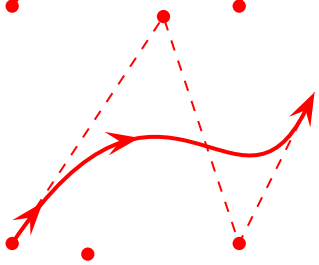
```
\psbezier[ArrowInside=-|](1,1)(2,2)(3,3)
\psset{linestyle=none,ArrowInside=-o}
\psbezier[ArrowInsidePos=0.25](1,1)(2,2)(3,3)
\psbezier[ArrowInsidePos=0.75](1,1)(2,2)(3,3)
\psset{linestyle=none,ArrowInside=-*}
\psbezier[ArrowInsidePos=0](1,1)(2,2)(3,3)
\psbezier[ArrowInsidePos=1](1,1)(2,2)(3,3)
```



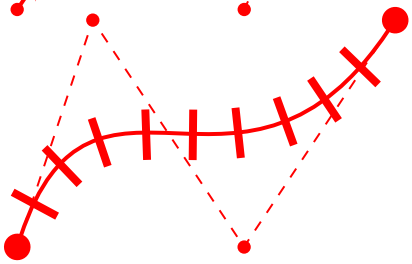
```
\psbezier[ArrowInside=->,%
    showpoints=true]%
       {*-*}(2,3)(3,0)(4,2)
```



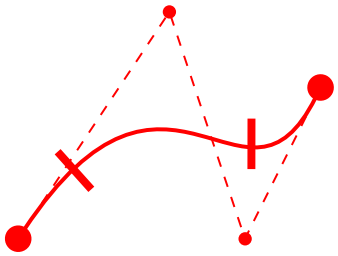
```
\psbezier[ArrowInside=->,%
    showpoints=true,%
    ArrowInsideNo=2](2,3)(3,0)(4,2)
```



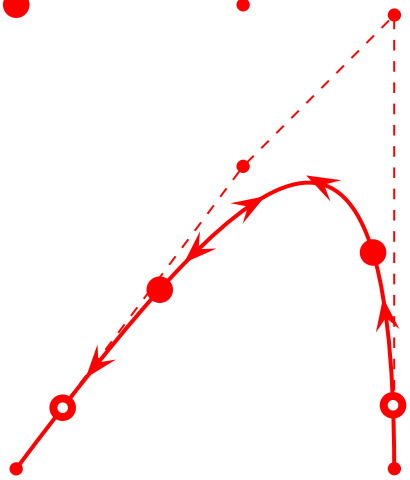
```
\psbezier[ArrowInside=->,%
    showpoints=true,%
    ArrowInsideNo=2,%
    ArrowInsideOffset=-0.2]{->}(2,3)(3,0)(4,2)
```



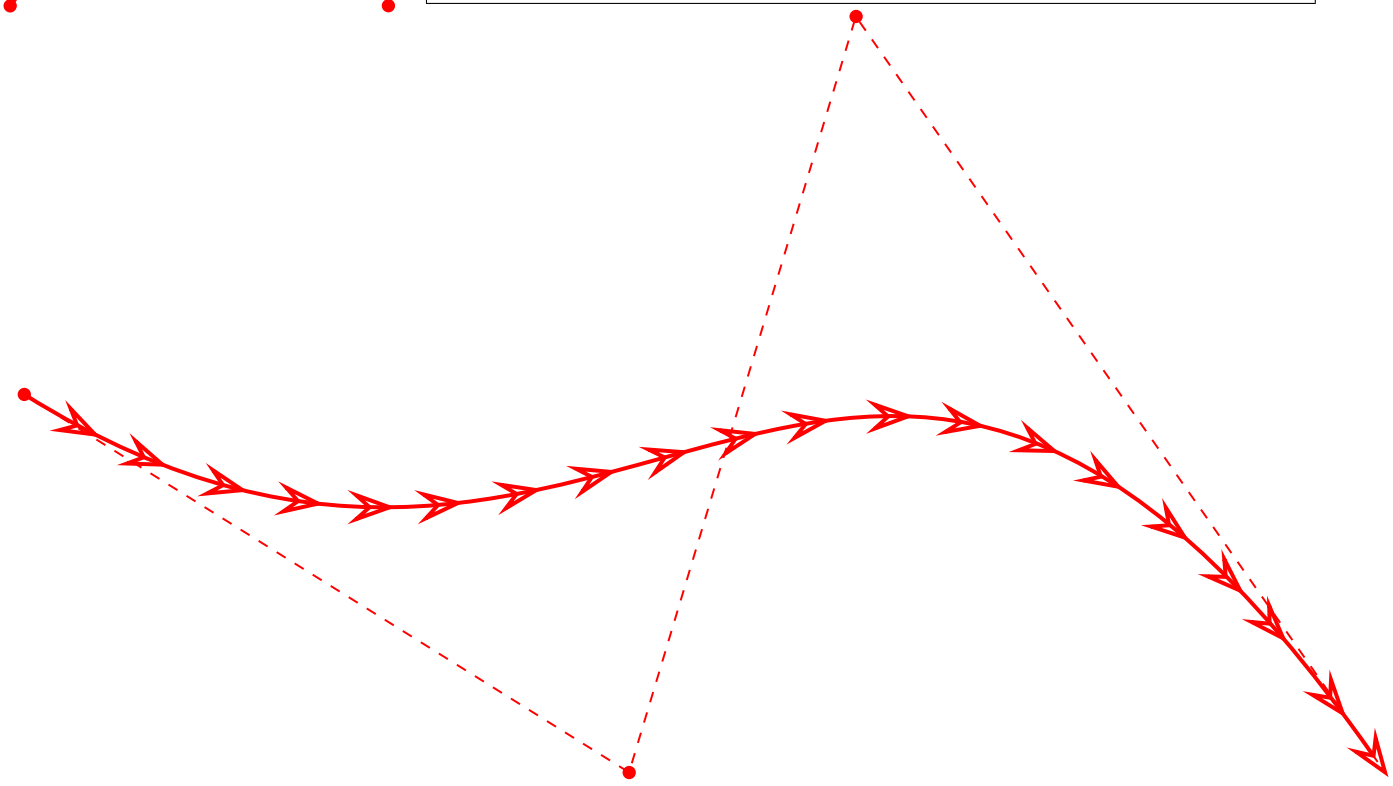
```
\psbezier[ArrowInsideNo=9,%
    ArrowInside=-|, showpoints=true]%
       {*-*}(1,3)(3,0)(5,3)
```



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



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



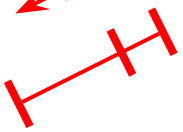
```
\psbezier[ArrowInsideNo=19,%
    ArrowInside=->,%
    showpoints=true]{->}(-3,0)(5,-5)(8,5)(15,-5)
```

8.4.4 \pcline

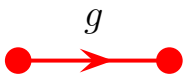
These examples need the package pst-node.

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

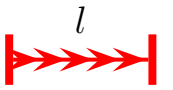
```
\pcline[ArrowInside=->]{<->}(0,0)(2,1)
```



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



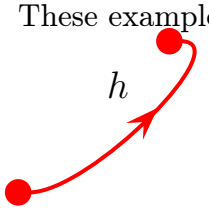

```
\pcline[ArrowInside=->,%
   ArrowInsidePos=0.65]{*-*}(0,0)(2,0)
\naput[labelsep=0.3]{\large$g$}
```



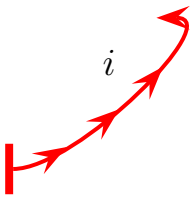

```
\pcline[ArrowInside=->,ArrowInsidePos=10]{|-|}(0,0)(2,0)
\naput[labelsep=0.3]{\large$l$}
```

8.4.5 \pccurve

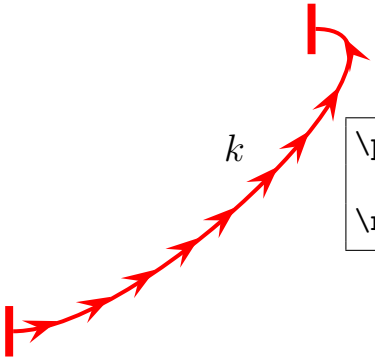
These examples also need the package pst-node.




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




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




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

9 \psFormatInt

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

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

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

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

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

Part II
pst-node

10 \nclineII

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

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


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

10.1 The options

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

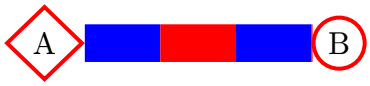
name	meaning
`dashColorI`	first color, default is `black`
`dashColorII`	second color, default is `red`
`dashNo`	the difference in per cent of the colored lines, default ist 0.2

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

10.2 Examples


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


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

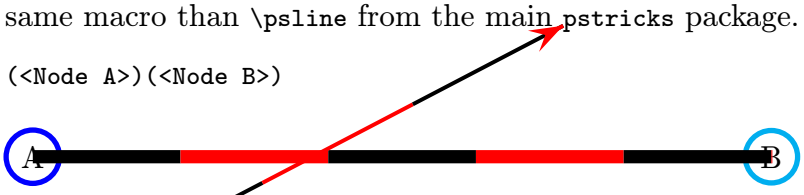



```
\dianode{A}{A}%
\hspace{3cm}%
\circlenode{B}{B}%
\nclineII[dashColorI=blue,%
    linecap=1,%
    dashNo=0.3,linewidth=0.5]{A}{B}
```

11 \pclineII

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

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




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

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

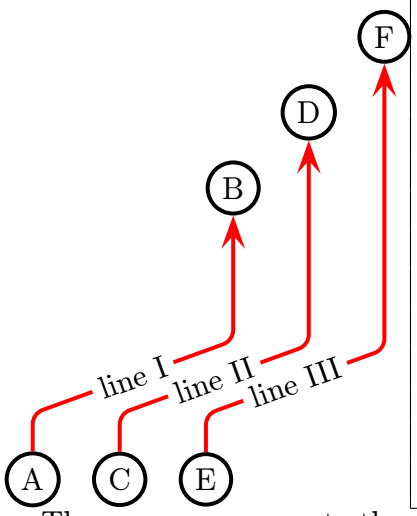
```
This macro makes only sense when connecting two
''invisible`` nodes, like this connection from
here\pnode{D}\pclineII{->}(D)(C){} to the above word \verb|pstricks|.
```

12 \ncdiag and \pcdiag

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

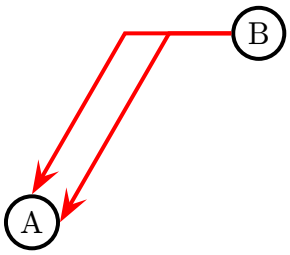
```
\ncdiag[<options>]{<Node A>}{<Node B>}
\pcdiag[<options>](<Node A>)(<Node B>)
```

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

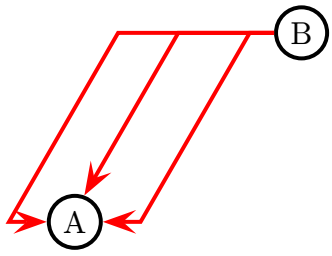



```
\psset{linecolor=black}
\circlenode{A}{A}%
\quad\circlenode{C}{C}%
\quad\circlenode{E}{E}
\rput(0,4){\circlenode{B}{B}}
\rput(1,5){\circlenode{D}{D}}
\rput(2,6){\circlenode{F}{F}}
{\psset{arrowscale=2,linearc=0.2,%
  linecolor=red,armA=0.5, angleA=90,angleB=-90}
\ncdiag[lineAngle=20]{->}{A}{B}
\ncput*[nrot=:U]{line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

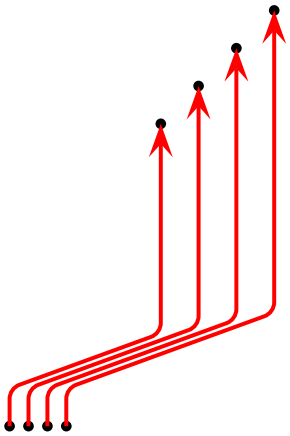
The `ncdiag` macro sets the `armB` dynamical to the calculated value. Any user setting of this `armB` is overwritten by the macro. The `armA` could be set to a zero length:




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




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

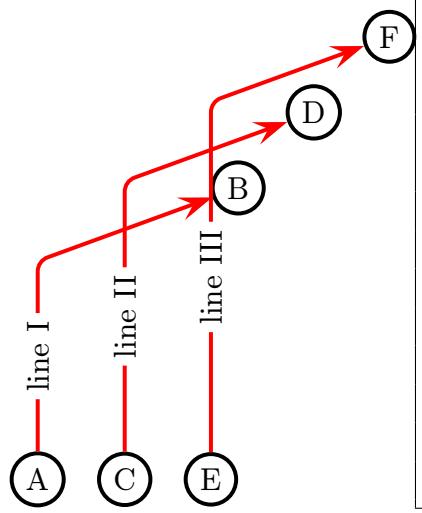


```
\psset{linecolor=black}
\cnode*(0,0){2pt}{A}%
\cnode*(0.25,0){2pt}{C}%
\cnode*(0.5,0){2pt}{E}%
\cnode*(0.75,0){2pt}{G}%
\cnode*(2,4){2pt}{B}%
\cnode*(2.5,4.5){2pt}{D}%
\cnode*(3,5){2pt}{F}%
\cnode*(3.5,5.5){2pt}{H}%
{\psset{arrowscale=2,linearc=0.2,%
  linecolor=red,armA=0.5, angleA=90,angleB=-90}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

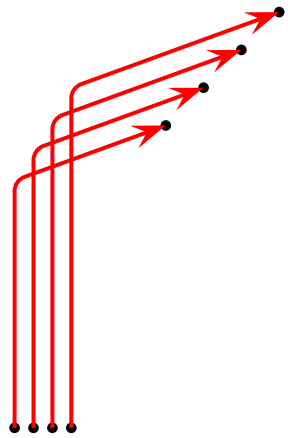
13 \ncdiagg and \pcdiagg

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

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

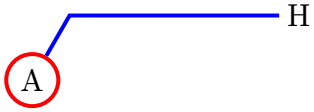



```
\psset{linecolor=black}
\circlenode{A}{A}%
\quad\circlenode{C}{C}%
\quad\circlenode{E}{E}
\rput(0,4){\circlenode{B}{B}}
\rput(1,5){\circlenode{D}{D}}
\rput(2,6){\circlenode{F}{F}}
{\psset{arrowscale=2,linearc=0.2,%
  linecolor=red,armA=0.5, angleA=90}
\ncdiagg[lineAngle=-160]{->}{A}{B}
\ncput*[nrot=:U]{line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



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

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




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


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


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

Part III
pst-plot

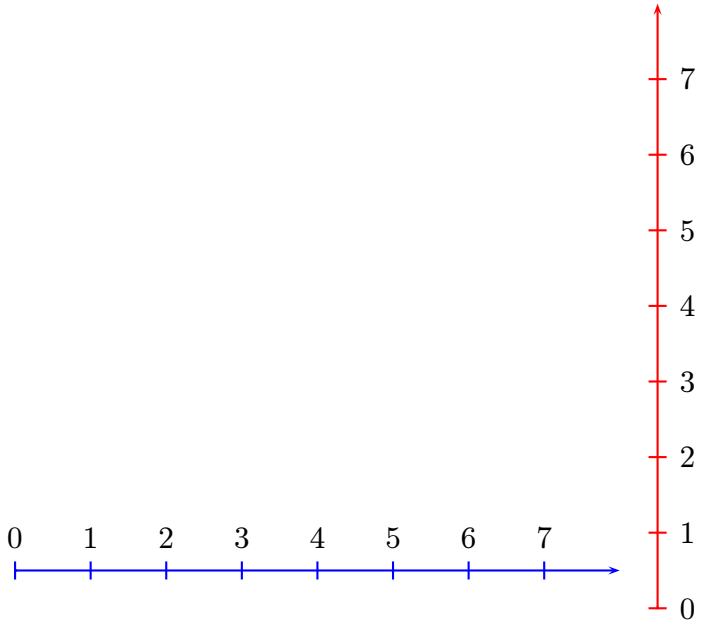
14 New macro \resetOptions

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

15 New options xyAxes, xAxes and yAxes

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

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




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

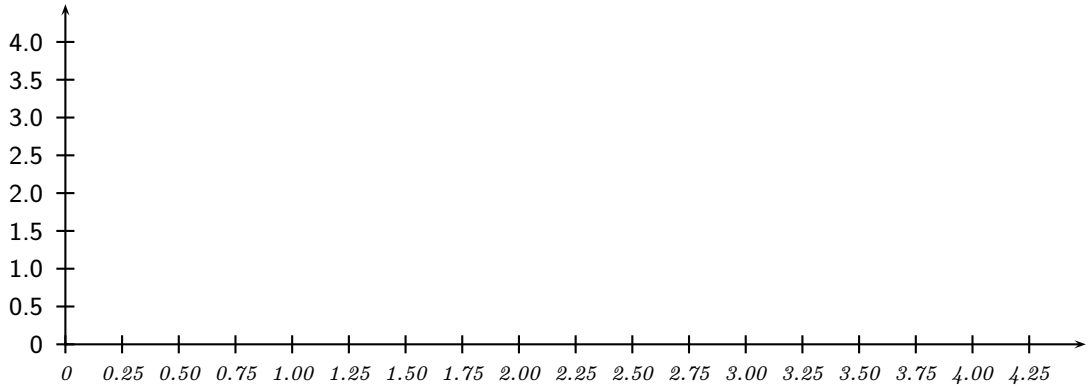
```
\end{pspicture}
```

16 New options xyLabel, xLabel and yLabel

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

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

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



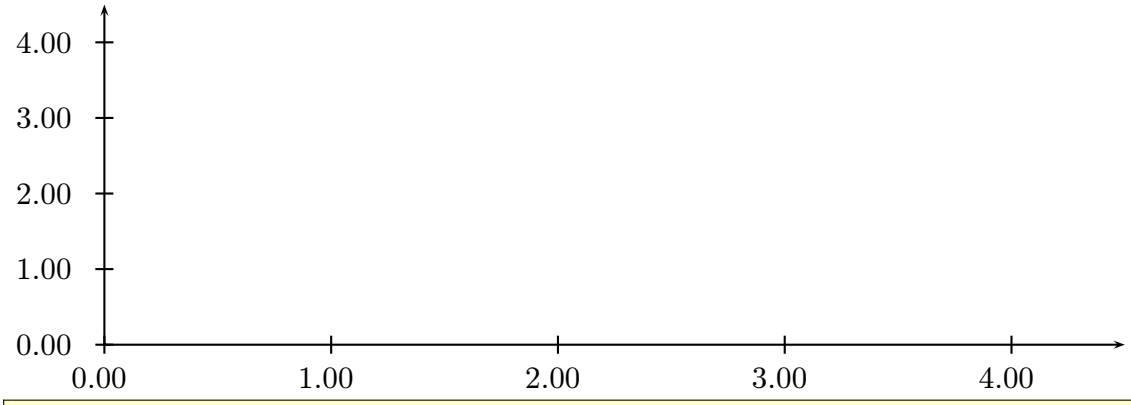

```
{\psset{yunit=1cm,xunit=3cm}
\begin{pspicture}(-0.2,-0.5)(5,4.75)
\psaxes[xLabel={\scriptsize\itshape},%
   yLabel={\sffamily\footnotesize},%
   Dy=0.5, Dx=0.25]{->}(0,0)(4.5,4.5)
\end{pspicture}}
```

17 New options xyDecimals, xDecimals and yDecimals

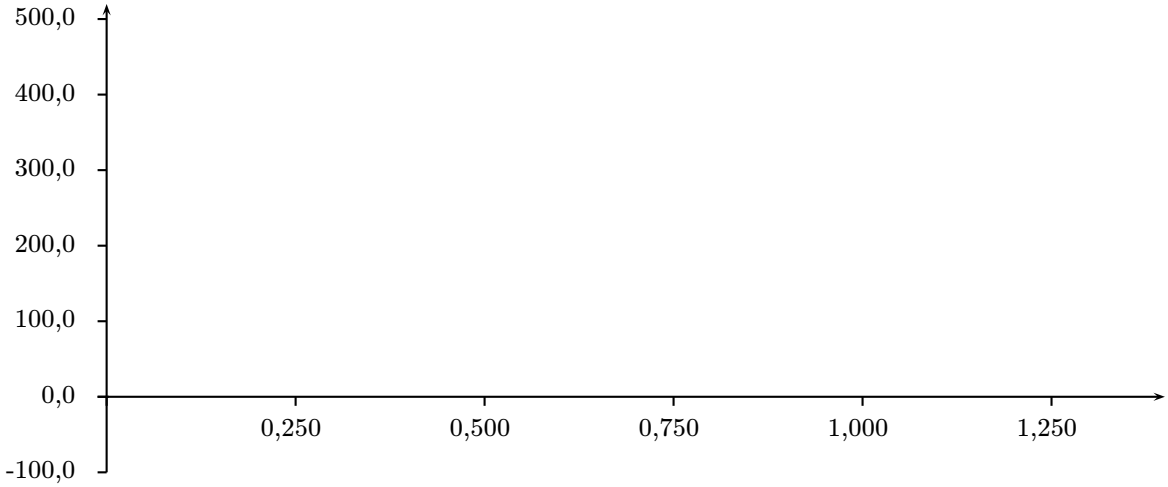
By default the labels of the axes get numbers with or without decimals, just depending to the numbers. With these options `??Decimals` it is possible to determine the decimals, where the option `xyDecimals` sets this identical for both axes.

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

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




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



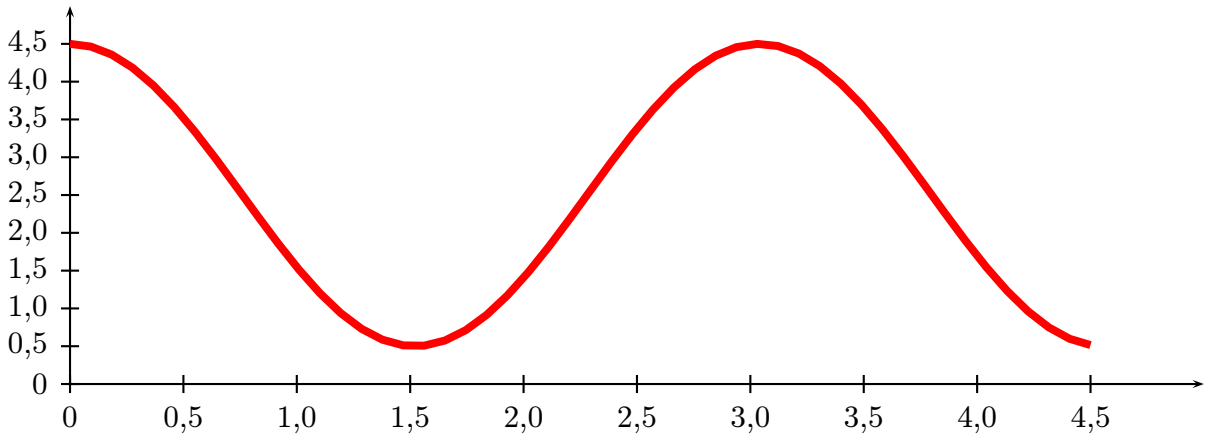

```
\begin{pspicture}(-0.1,-150)(1.5,550.0)
 \psaxes[Dx=0.25,Dy=100, tickstyle=bottom,%
  xyLabel=\footnotesize,comma,xDecimals=3,yDecimals=1]%
  {->}(0,0)(0,-100)(1.4,520)
\end{pspicture}
```

18 New option `comma`

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

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

```
\end{pspicture}
```



19 New options logBase, xlogBase and ylogBase

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

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

For an intervall of $[10^{-3}...10^{2}]$ choose a `pstricks` intervall of [-3,2]. `pstricks` takes 0 as the origin of this axes, which is wrong if we want to have a logarithm axes. With the options `Oy` and `Ox` we can set the origin to -3, so that the first label gets 10^{-3}. If this is not done by the user than `pst-plot-add` does it by default.

19.1 y axis (ylogBase)

Figure 7 shows the graph of the function $y = \log x$ with a logarithmic y-axis. The code is:

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

The values for the `psaxes` y-coordinate are now the exponents to the base 10 and for the right function to the base e: $10^{-3} \ldots 10^{1}$ which corresponds to the given y-intervall $-3 \ldots 1.5$, where

only integers as exponents are possible. These logarithm labels have no effect to the internal used units. To draw the logarithm function we have to use the math function

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

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

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

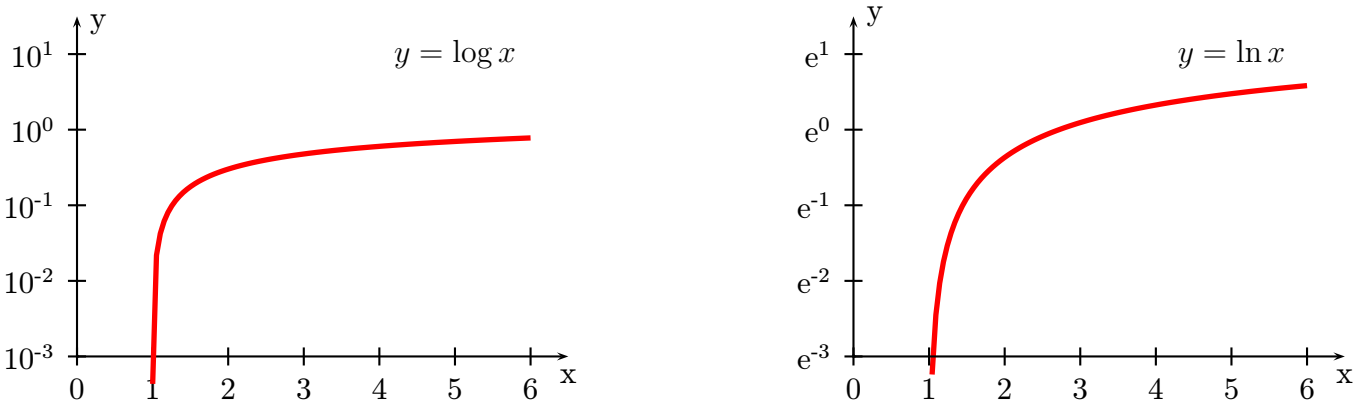


Figure 7: log axes usage: (`ylogBase=10` and ylogBase=e)

19.2 x axis (`xlogBase`)

Figure 8 shows the same for the x axis. Now we have to use the easy math function

$$y = x$$

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

The code for figure 8:

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

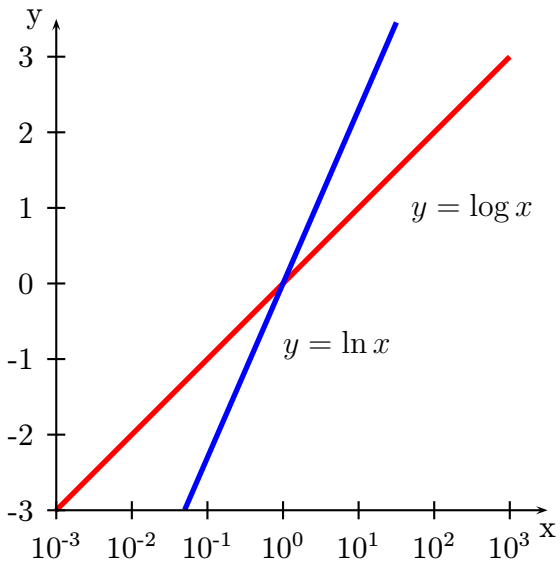
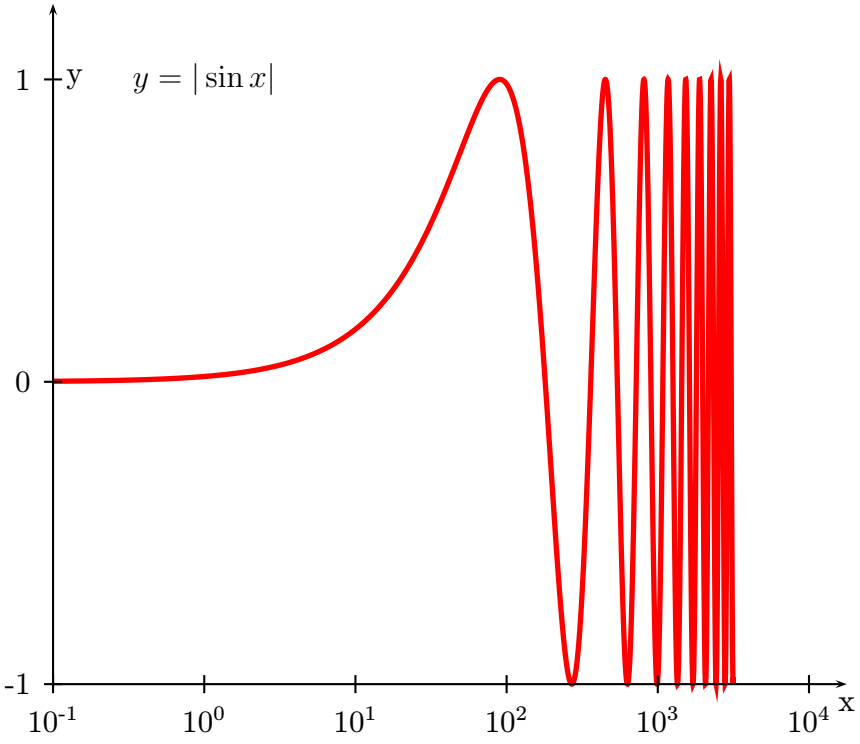


Figure 8: log axes usage: (`xlogBase=10`)




```
{\psset{yunit=4cm,xunit=2cm}
\begin{pspicture}(-1.25,-1.25)(4.25,1.5)
 \uput[-90](4.25,-1){x}
 \uput[0](-1,1){y}
 \rput(0,1){$y=|\sin x|$}
```

```
 \psplot[linewidth=2pt,%
  plotpoints=5000,linecolor=red]{-1}{3.5}{10 x exp sin } % y=|sin(x)|
 \psaxes[xlogBase=10,logLines=x,Oy=-1]{->}(-1,-1)(4.25,1.25)
\end{pspicture}}
```

19.3 All axes (logBase)

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

$$y = \log x$$

and is shown in figure 9.

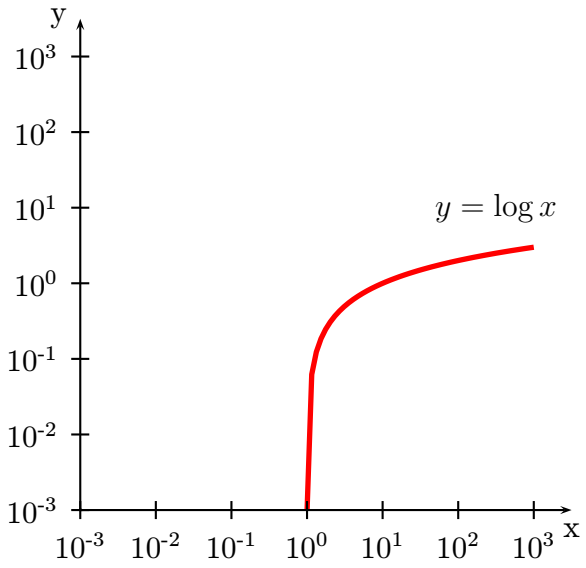


Figure 9: log axes usage: (xlogBase=10)

The code for figure 9:

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

19.4 No logstyle (logBase={})

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

The code for figure 10:

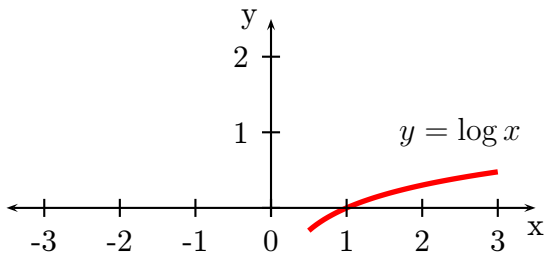


Figure 10: log axes usage: (`logBase={}`)

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

19.5 More examples for the `logBase` option

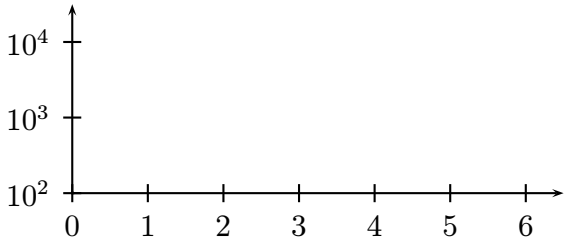


Figure 11: log axes usage: (`ylogBase=10`)

The code for figure 11:

```
\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}
```

The code for figure 12:

```
\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}
```

The code for figure 13:

```
\begin{pspicture}(-0.5,-1.25)(6.5,4.5)
  \psplot{0}{6}{x x cos add log}                  % x   + cox(x)
```

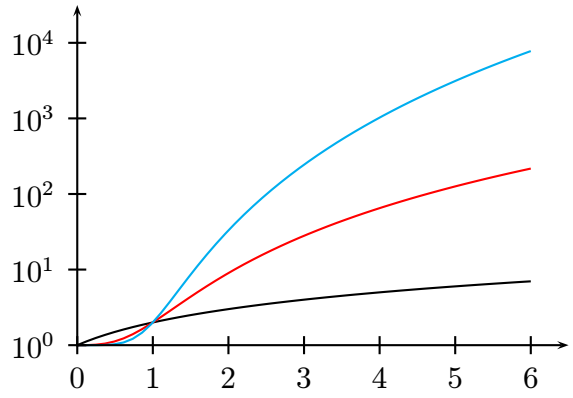


Figure 12: log axes usage: (`ylogBase=10`)

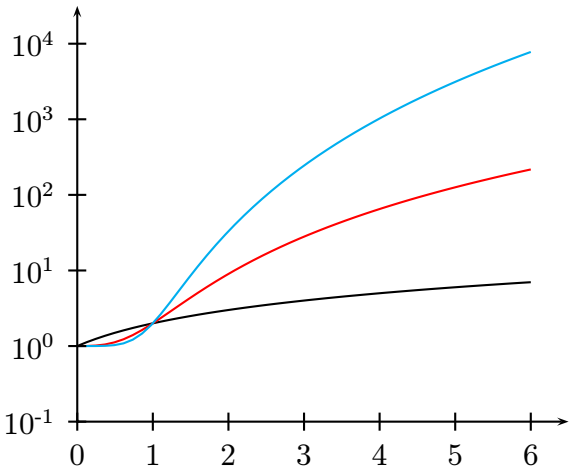


Figure 13: log axes usage: (`ylogBase=10`)

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

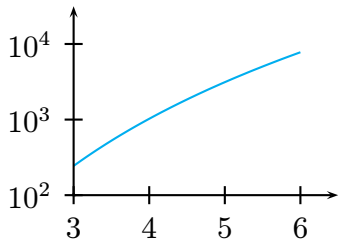


Figure 14: log axes usage: (`ylogBase=10`)

The code for figure 14:

```
\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}
```

20 New option `logLines`

The syntax is

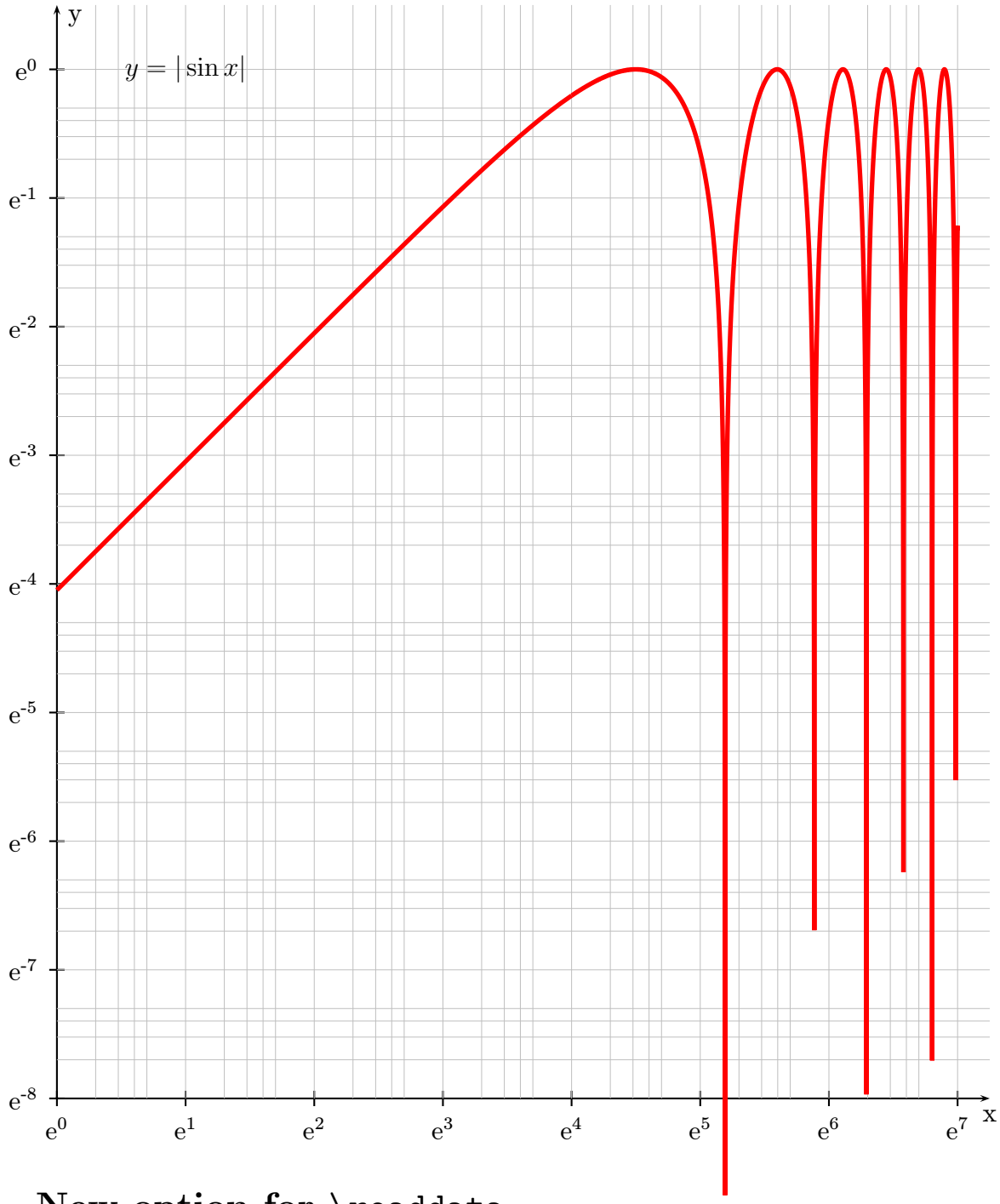
`logLines=all|x|y|none`

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


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

The default settings for these logarithm lines is

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



21 New option for \readdata

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

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

The default value for `nStep` is 1.

22 New options for \listplot

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

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

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

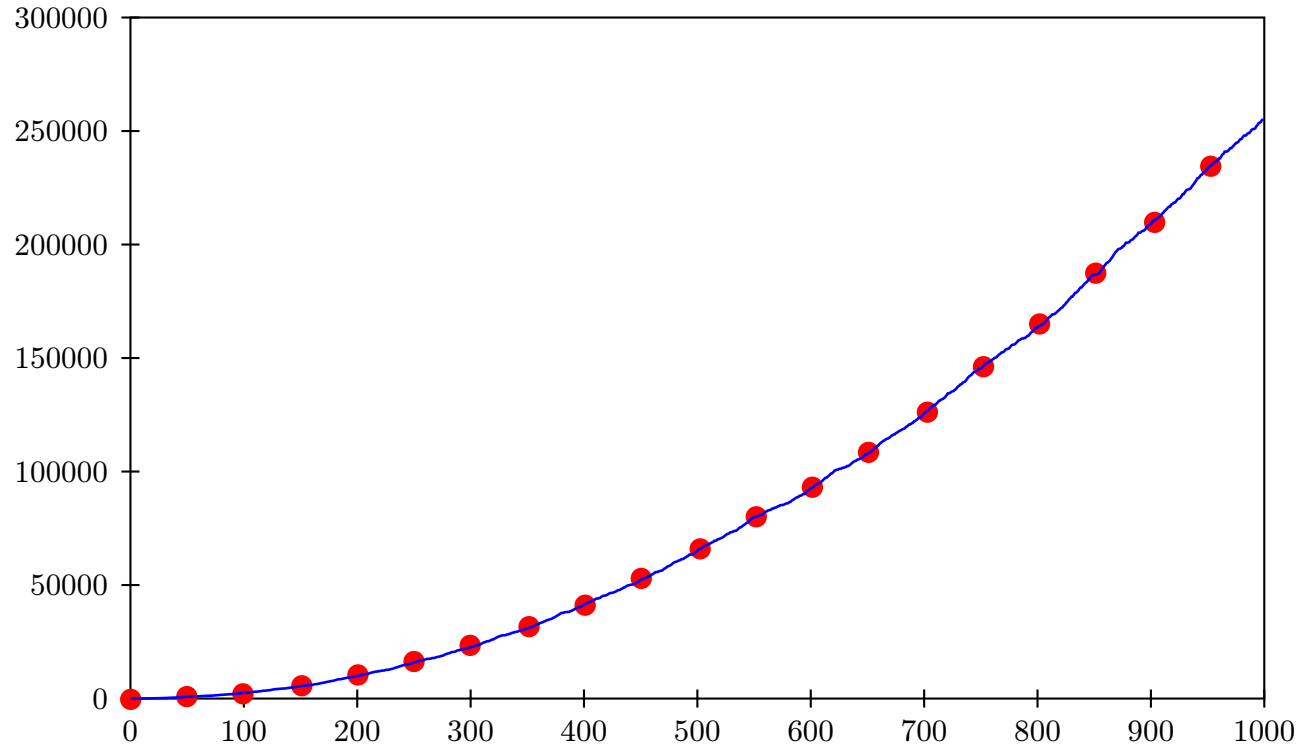
```
\readdata[nStep=10]{\data}{/home/voss/data/data1.dat}
```

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

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

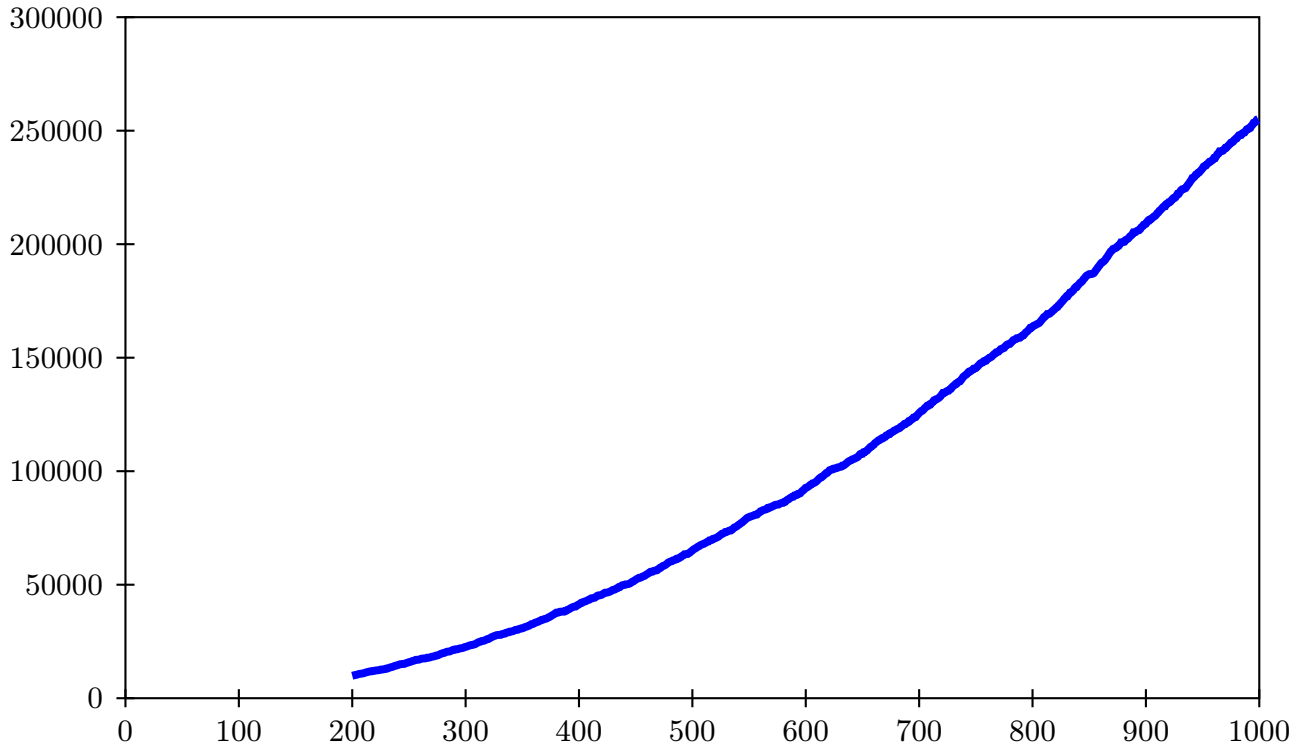
22.1 Example for nStep/xStep

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



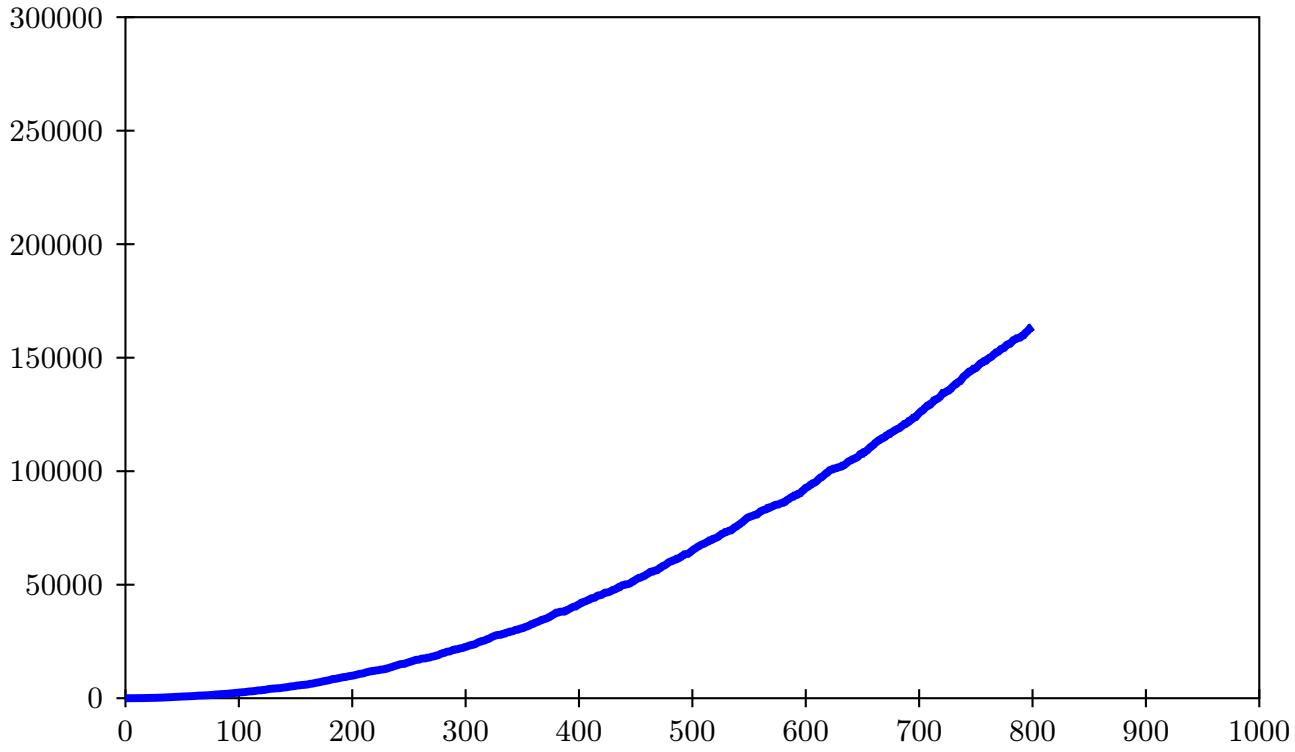

```
\readdata{\data}{data.dat}
\psset{xunit=0.15mm,yunit=0.0003mm}
\begin{pspicture}(-80,-30000)(1000,300000)
\psaxes[axesstyle=frame,Dx=100,dx=100,%
    Dy=50000,dy=50000](1000,300000)
\listplot[nStep=50,%
    linewidth=3pt,%
    linecolor=red,%
    plotstyle=dots]{\data}
\listplot[linewidth=1pt,%
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

22.2 Example for nStart/xStart



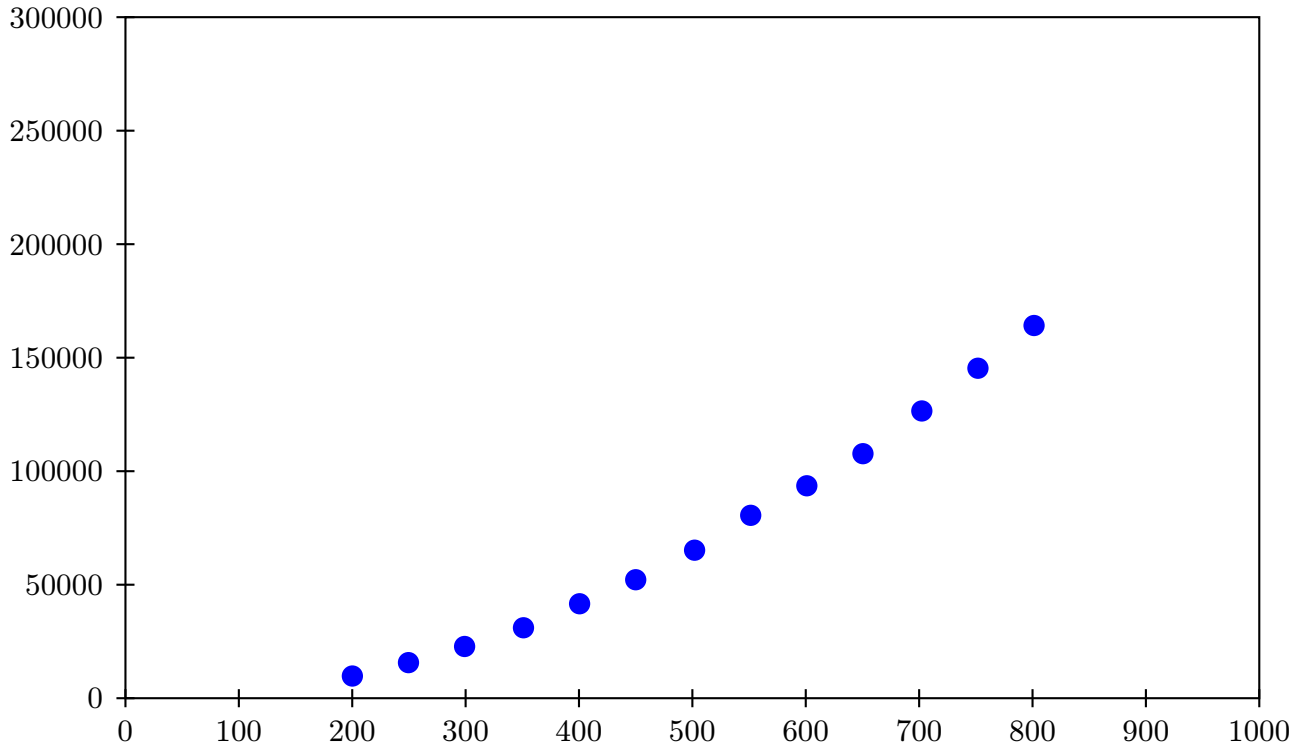

```
\readdata{\data}{data.dat}
\psset{xunit=0.15mm,yunit=0.0003mm}
\begin{pspicture}(-80,-30000)(1000,310000)
\psaxes[axesstyle=frame,Dx=100,dx=100,%
   Dy=50000,dy=50000](1000,300000)
\listplot[nStart=200,%
 linewidth=3pt,%
   linecolor=blue]{\data}
\end{pspicture}
```

22.3 Example for nEnd/xEnd




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

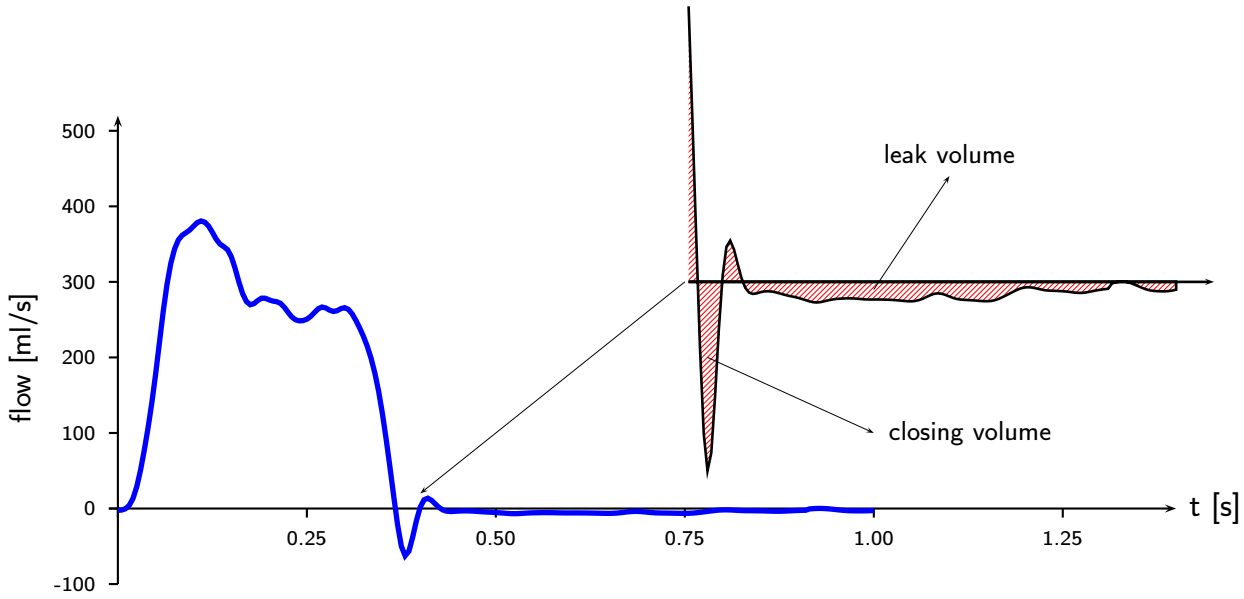
22.4 Example for all new options




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

22.5 Example for `xStart`

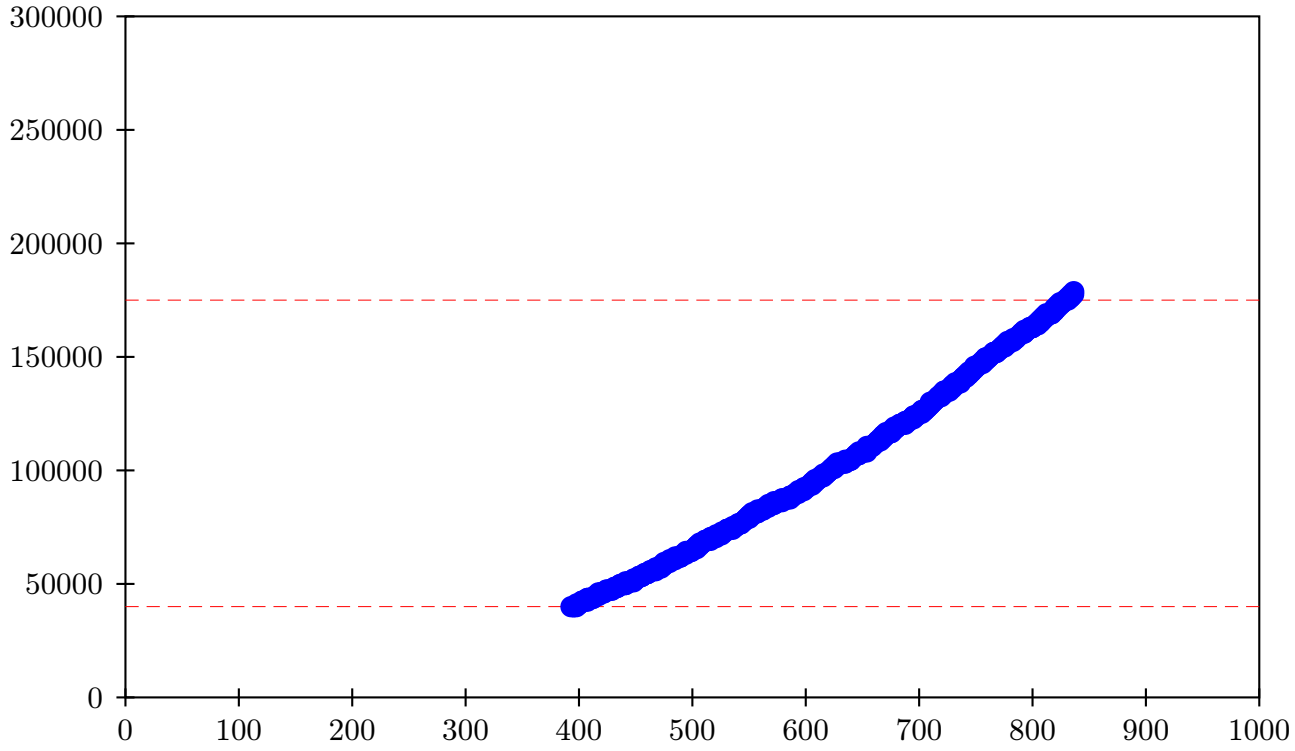
This example shows the use of the same plot with different units and different `xStart` value. The blue curve is the original plot of the data records. To show the important part of the curve there is another one plotted with a greater `yunit` and a start value of `xStart=0.35`. This makes it possible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




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

```
\end{pspicture}
```

22.6 Example for xStart




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

22.7 Example for plotNo/plotNoMax

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

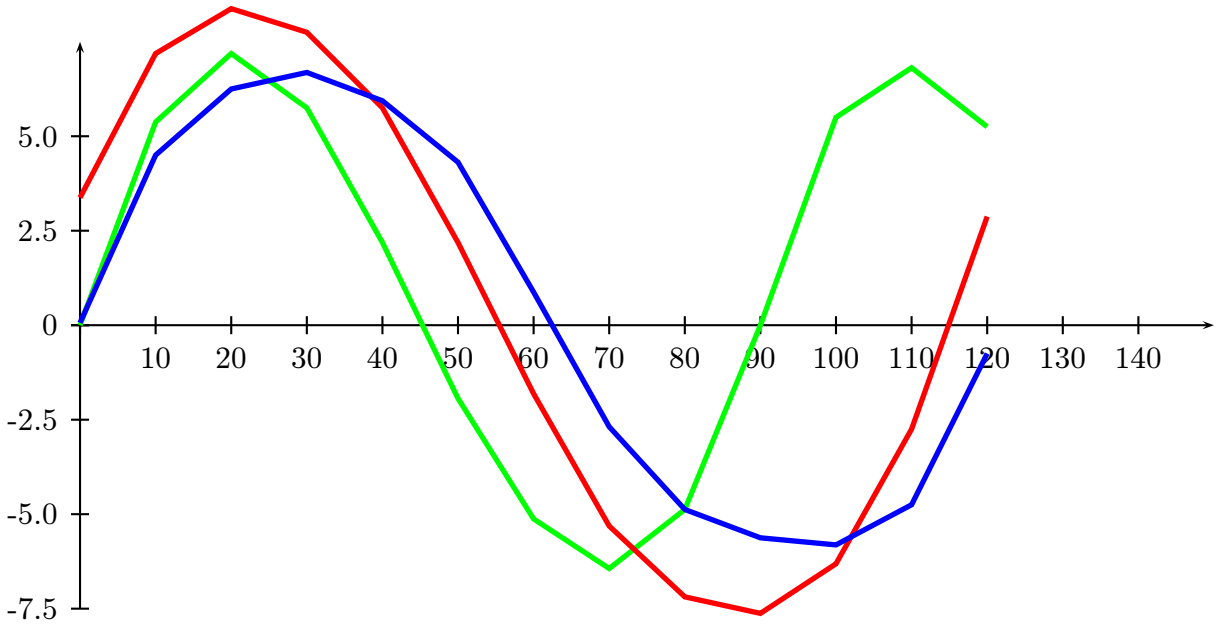
```
x y1 y2 y3 y4 ... yMax
x y1 y2 y3 y4 ... yMax
...
```

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

We have the following data file:

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

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




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

\psset{xunit=0.1cm, yunit=0.5cm}
\begin{pspicture}(0,-7.5)(150,10)
```

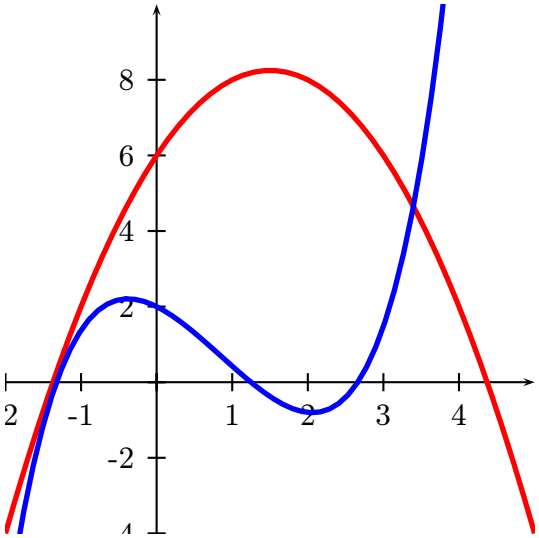
```
\psaxes[Dx=10,Dy=2.5]{->}(0,0)(0,-7.5)(150,7.5)
\psset{linewidth=2pt,plotstyle=line}
\listplot[linecolor=green,plotNo=1,plotNoMax=3]{\Data}
\listplot[linecolor=red,plotNo=2,plotNoMax=3]{\Data}
\listplot[linecolor=blue,plotNo=3,plotNoMax=3]{\Data}
\end{pspicture}
```

23 New macro psplotPolynomial

Plotting a polynom is one of the standard cases in mathematics. `pstricks-add` provides the new macro `\psplotPolynomial` which makes it very easy to define the function:

`\psplotPolynomial[coeff=<a0 a1 a2 ...>,<other options>]{<xStart>}{<xEnd>}`

Only the order of the coefficients is important not the number.




```
\psset{yunit=0.5cm,xunit=1cm}
\begin{pspicture*}(-2,-4)(5,10)
 \psaxes[Dy=2]{->}(0,0)(-2,-4)(5,10)
 \psplotPolynomial[coeff=6 3 -1,linecolor=red,linewidth=2pt]{-2}{5}
 \psplotPolynomial[coeff=2 -1 -1 .5 -.1 .025,linecolor=blue,linewidth=2pt]{-2}{4}
\end{pspicture*}
```

24 Credits

Denis Girou | Jens-Uwe Morawski | Timothy Van Zandt

References

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

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

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

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

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

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

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

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

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

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

25 The code

```
%%
%% This is file ‘pstricks-add.tex’,
%%
%% IMPORTANT NOTICE:
%%
%% Package ‘pstricks-add.tex’
%%
%% Herbert Voss <voss@perce.de>
%%
%% This program can be redistributed and/or modified under the terms
%% of the LaTeX Project Public License Distributed from CTAN archives
%% in directory macros/latex/base/lppl.txt.
%%
%% DESCRIPTION:
%%  ‘pstricks-add’ is a PSTricks package for additionals to the standard
%%       pstricks package
%%
\def\fileversion{1.2}
\let\pstricksAddFV\fileversion
\def\filedate{2004/02/10}
\message{‘pstricks-add’ v\fileversion, \filedate\space (Herbert Voss)}
\csname PSTricksAddLoaded\endcsname
\let\PSTricksAddLoaded\endinput
% Requires PSTricks, pst-node
\usepackage{pstcol}
\ifx\PSTricksLoaded\endinput\else\input pstricks \fi
\ifx\PSTnodesLoaded\endinput\else\input pst-node \fi
\ifx\PSTplotLoaded\endinput\else\input pst-plot \fi
\usepackage{multido}
%
\input pst-key
%
\edef\PstAtCode{\the\catcode‘\@} \catcode‘\@=11\relax
\SpecialCoor
%
%%%%%% \begin{pspicture} %%%%%%%%%%%%%%%%%%
\define@key{psset}{mode}{\edef\psk@mode{#1}}
\setkeys{psset}{mode=0}
\newif\ifps@Frame
\def\psset@frame#1{\@nameuse{ps@Frame#1}}
\psset@frame{false}
\define@key{psset}{frameStyle}{\edef\psk@frameStyle{#1}}
\setkeys{psset}{frameStyle=dashed}
%
\def\pst@picture{\@ifnextchar[{\pst@@picture}{\pst@@picture[mode=0]}}
```

```
\def\pst@@picture[#1]#2(#3,#4){%
 \psset{#1}%
 \@ifnextchar(%
  {\pst@@@picture[\psk@mode](#3,#4)}%
  {\pst@@@picture[\psk@mode](0,0)(#3,#4)}%
}
\def\endpspicture{%
 \pst@killglue
 \endgroup
 \egroup
 \ifdim\wd\pst@hbox=\z@\else\fi
 \ht\pst@hbox=\pst@dimd
 \dp\pst@hbox=-\pst@dimb
 \setbox\pst@hbox=\hbox{%
 \kern-\pst@dima
 \ifx\pst@tempa\@empty\else
  \advance\pst@dimd-\pst@dimb
  \pst@dimd=\pst@tempa\pst@dimd
  \advance\pst@dimd\pst@dimb
  \lower\pst@dimd
 \fi
 \box\pst@hbox%
 \kern\pst@dimc}%
 \if@star\setbox\pst@hbox=\hbox{\clipbox@@\z@}\fi%
 \leavevmode%
 \ifps@Frame%
%  \setbox\@tempboxa\hbox{\box
  \psframebox[framesep=0pt,linestyle=\psk@frameStyle]{\box\pst@hbox}%
%  {\fboxsep=0pt\@frameb@x\relax}%
 \else\box\pst@hbox\fi%
 \endgroup%
}%
%
% A modulo macro for integer values
% \pst@mod{34}{6}\value ==> \value is 4
%
\def\pst@mod#1#2#3{%
 \pst@cnta=#1\pst@cntb=#2\relax
 \pst@cntc=\pst@cnta
 \divide\pst@cnta by \pst@cntb
 \multiply\pst@cntb by \pst@cnta
 \advance\pst@cntc by -\pst@cntb
 \def\pst@tempa{\the\pst@cntc}
 \let#3\pst@tempa
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%
\define@key{psset}{intSeparator}{\edef\psk@intSeparator{#1}}
\setkeys{psset}{intSeparator={,}}
%
\def\psFormatInt{\@ifnextchar[{\psFormatInt@i}{\psFormatInt@i[]}}
\def\psFormatInt@i[#1]#2{{%
    \def\ps@tempa{#1}
    \ifx\ps@tempa\@empty\else\psset{#1}\fi
    \count1=#2\count2=\count1
    \ifnum\count1=0 0\else
 \ifnum\count1>999999
     \count3=\count1
     \divide\count3 by 1000000
     \the\count3\psk@intSeparator
     \multiply\count3 by 1000000
     \advance\count1 by -\count3 % modulo 1000000
 \fi
 \ifnum\count2>999
     \count3=\count1
     \divide\count3 by 1000
     \ifnum\count2>99999
  \ifnum\count3<100 0\fi
  \ifnum\count3<10 0\fi
     \fi
     \the\count3\psk@intSeparator
     \multiply\count3 by 1000
     \advance\count1 by -\count3 %modulo 1000
 \fi%
 \ifnum\count2>999
     \ifnum\count1<100 0\fi
     \ifnum\count1<10 0\fi
 \fi
 \the\count1
    \fi
}}
%
% a new fillstyle
\def\psfs@asolid{\pst@fill{\pst@usecolor\psfillcolor eofill}}
%
\define@key{psset}{braceWidth}{\edef\psk@braceWidth{#1}}
\define@key{psset}{bracePos}{\edef\psk@bracePos{#1}}
\setkeys{psset}{braceWidth=0.35,bracePos=0.5}
%
\def\@@rput@iv(#1){\pst@killglue\pst@makebox{\@@rput@v{#1}}}
\def\@@rput@v#1{%
 \begingroup
  \use@par
```

```
  \pst@makesmall\pst@hbox
  \pst@Verb{%
   Alpha 90 sub \psk@braceWidth\space 0 lt {180 add} if
   \ifx\psk@rot\@empty\else\psk@rot add \fi
   /rotAngle exch def
  }%
  \setbox\pst@hbox=\hbox{%
   \pst@Verb{rotAngle \tx@RotBegin}%
   \box\pst@hbox\pst@Verb{\tx@RotEnd}%
  }%
  \psput@{#1}\pst@hbox
 \endgroup
 \ignorespaces%
}
%
\def\psbrace{\@ifnextchar[{\@psbrace}{\@psbrace[]}}
\def\@psbrace[#1](#2)(#3)#4{{
 \setkeys{psset}{linearc=0.2,linewidth=1pt,%
  nodesepA=0pt,nodesepB=0pt,bracePos=0.5}% the default
 \setkeys{psset}{#1}%
 \pst@getcoor{#2}\pst@tempa%
 \pst@getcoor{#3}\pst@tempb%
 \pnode(!%
  /bW2 \psk@braceWidth\space 2.0 div def
  \pst@tempa /YA exch \pst@number\psyunit div def
  /XA exch \pst@number\psxunit div def
  \pst@tempb /YB exch \pst@number\psyunit div def
  /XB exch \pst@number\psxunit div def
  /Alpha YB YA sub XB XA sub atan def
  /xMid XB XA sub \psk@bracePos\space mul XA add def
  /yMid YB YA sub \psk@bracePos\space mul YA add def
  /@deltaX Alpha sin bW2 mul def
  /@deltaY Alpha cos bW2 mul def
  /@xTemp xMid @deltaX 2 mul add def
  /@yTemp yMid @deltaY 2 mul sub def
  @xTemp @yTemp){@tempNode}
 \pst@getcoor{@tempNode}\pst@tempc%
 \@@rput@iv(! %
  \pst@tempc /Yc exch def
  /Xc exch def
  Xc \psk@nodesepA\space add \pst@number\psxunit div
  Yc \psk@nodesepB\space add \pst@number\psxunit div ){#4}
 \psline(#2)%
  (! XA @deltaX add YA @deltaY sub)%
  (! @xTemp @deltaX sub @yTemp @deltaY add)%
  (@tempNode)
 \psline(@tempNode)%
```

```
  (! @xTemp @deltaX sub @yTemp @deltaY add)%
  (! XB @deltaX add YB @deltaY sub)%
  (#3)%
}}
%
% from Dennis Giroux: http://www.tug.org/pipermail/pstricks/2001/000507.html
%
% I - Definition of \PstWedgeEllipse, a generalization of \pswedge for wedges
%     of ellipses (from the code of \pswedge and \psellipse)
%
\def\psWedgeEllipse{\def\pst@par{}\pst@object{psWedgeEllipse}}
\def\psWedgeEllipse@i(#1){%
 \@ifnextchar({\psWedgeEllipse@ii(#1)}{\pstWedgeEllipse@ii(0,0)(#1)}}
\def\psWedgeEllipse@ii(#1)(#2)#3#4{%
 \begin@ClosedObj
  \pst@getangle{#3}\pst@tempa
  \pst@getangle{#4}\pst@tempb
  \pst@getcoor{#1}\pst@tempc
  \pst@@getcoor{#2}%
  \def\pst@linetype{1}%
  \addto@pscode{%
   \pst@tempa \pst@tempb
   \pst@coor
   \pst@tempc moveto
   \ifdim\psk@dimen\p@=\z@\else
    \psk@dimen CLW mul dup 3 1 roll
    sub 3 1 roll sub exch
   \fi
   \pst@tempc
   \tx@Ellipse
   closepath%
  }%
  \showpointsfalse
 \end@ClosedObj%
}
%
% arcs
%
\def\psEllipticArcN{\def\pst@par{}\pst@object{psellipticarcn}}
\def\psellipticarcn@i{\let\if@psarcn\iftrue\psellipticarc@ii}
%
\def\psEllipticArc{\def\pst@par{}\pst@object{psellipticarc}}
\def\psellipticarc@i{\let\if@psarcn\iffalse\psellipticarc@ii}
%
\let\if@psarcn\iffalse
\def\psellipticarc@ii{\pst@getarrows\psellipticarc@iii}
\def\psellipticarc@iii(#1){%
```

```
 \@ifnextchar({\psellipticarc@iv(#1)}{\psellipticarc@iv(0,0)(#1)}}
\def\psellipticarc@iv(#1)(#2)#3#4{%
 \begin@OpenObj
  \pst@getcoor{#1}\pst@tempa
  \pst@getcoor{#2}\pst@tempb
  \pst@getangle{#3}\pst@tempc
  \pst@getangle{#4}\pst@tempd
  \addto@pscode{\psellipticarc@definearg \psellipticarc@draw}%
  \showpointsfalse
 \end@OpenObj%
}
\def\psellipticarc@definearg{%
 \pst@tempa /y ED /x ED % Origin
 \pst@tempb            % radii. Now adjust:
 \ifdim\psk@dimen\p@=\z@\else
  \psk@dimen CLW mul dup 3 1 roll
  sub 3 1 roll sub exch
 \fi
 /ry ED /rx ED
 /angleA
  /d  {  \if@psarcn sub \else add \fi } def
  \pst@tempc \psk@arcsepA 2 div
  \tx@ArcAdjust
 def
 /angleB
  /d  {  \if@psarcn add \else sub \fi } def
  \pst@tempd \psk@arcsepB 2 div
  \tx@ArcAdjust
 def
 \ifshowpoints\psellipticarc@showpoints\fi
 \ifx\psk@arrowA\@empty
  \ifnum\psk@liftpen=2
   angleA cos rx mul x add
   angleA sin ry mul y add
   moveto
  \fi
 \fi%
}
\def\psellipticarc@draw{%
 0 0 1
 angleA
 \ifx\psk@arrowA\@empty\else
  { ArrowA CP }
  { \if@psarcn sub \else add \fi }
  \tx@EllipticArcArrow
 \fi
 angleB
```

```
 \ifx\psk@arrowB\@empty\else
  { ArrowB }
  { \if@psarcn add \else sub \fi }
  \tx@EllipticArcArrow
 \fi
 /mtrx CM def
 x y T
 rx ry scale
 \if@psarcn arcn \else arc \fi
 mtrx setmatrix%
}
\def\psellipticarc@showpoints{%
 gsave
 /mtrx CM def
 x y T
 rx ry scale
 0 0 moveto
 0 0 1 \pst@tempc \pst@tempd
 \ifcase\psarc@type arc \or arcn \fi
 closepath
 mtrx setmatrix
 CLW 2 div SLW
 [ \psk@dash\space ] 0 setdash stroke
 grestore %
}
\pst@def{ArcAdjust}<%
% given a target length (targetLength) and an initial angle (angle0) [in the stack],
% let M(angle0)=(rx*cos(angle0),ry*sin(angle0))=(x0,y0).
% This computes an angle t such that (x0,y0) is at distance targetLength from the
    point M(t)=(rx*cos(t),ry*sin(t)).
% NOTE: this an absolute angle, it does not have to be added or substracted to angle0
% contrary to TvZ's code.
% To achieve, this, one iterates the following process: start with some angle t,
% compute the point M' at distance targetLength of (x0,y0) on the semi-line [(x0,y0) M
    (t)].
% Now take t' (= new angle) so that (0,0) M(t') and M' are aligned.
%
% Another difference with TvZ's code is that we need d (=add/sub) to be defined.
% the value of d = add/sub is used to know on which side we have to move.
% It is only used in the initialisation of the angle before the iteration.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Input stack: 1: target length 2: initial angle
% variables used : rx, ry, d (=add/sub)
%
 /targetLength ED /angle0 ED
 /x0 rx angle0 cos mul def
```

```
 /y0 ry angle0 sin mul def
% we are looking for an angle t such that (x0,y0) is at distance targetLength from the point M(t)=(rx*cos(t),ry*sin(t)))
%initialisation of angle (using 1st order approx = TvZ's code)
 targetLength 57.2958 mul
 angle0 sin rx mul dup mul
 angle0 cos ry mul dup mul
 add sqrt div
% if initialisation angle is two large (more than 90 degrees) set it to 90 degrees
% (if the ellipse is very curved at the point where we draw the arrow, the value can be much more than 360 degrees !)
% this should avoid going on the wrong side (more than 180 degrees) or go near
% a bad attractive point (at 180 degrees)
 dup 90 ge { pop 90 } if
 angle0 exch d
% maximum number of times to iterate the iterative procedure:
 30
% iterative procedure: takes an angle t on top of stack, computes a better angle (an put it on top of stack)
 { dup
% compute distance D between (x0,y0) and M(t)
 dup cos rx mul x0 sub dup mul exch sin ry mul y0 sub dup mul add sqrt
% if D almost equals targetLength, we stop
 dup targetLength sub abs 1e-5 le { pop exit } if
% stack now contains D t
% compute the point M(t') at distance targetLength of (x0,y0) on the semi-line [(x0,y0) M(t)]:
% M(t')= ( (x(t)-x0)*targetLength/d+x0 , (y(t)-y0)*targetLength/d+y0 )
 exch dup cos rx mul x0 sub exch sin ry mul y0 sub
% stack contains: y(t)-y0, x(t)-x0, d
 2 index \tx@Div targetLength mul y0 add ry \tx@Div exch
 2 index \tx@Div targetLength mul x0 add rx \tx@Div
% stack contains x(t')/rx , y(t')/ry , d
% now compute t', and remove D from stack
 atan exch pop
 } repeat
% we don't look at what happened... in particular, if targetLength is greater than the diameter of the ellipse...
% the final angle will be around /angle0 + 180. maybe we should treat this pathological case...
%after iteration, stack contains an angle t such that M(t) is the tail of the arrow
% to give back the result as a an angle relative to angle0 we could add the following line:
% angle0 sub 0 exch d
>
\pst@def{EllipticArcArrow}<%
 /d ED     % add/sub
```

```
 /b ED     % arrow procedure
 /a1 ED    % angle
 gsave
 newpath
 0 -1000 moveto
 clip               % Set clippath far from arrow.
 newpath
 0 1 0 0 b          % Draw arrow to determine length.
 grestore
% Length of arrow is on top of stack. Next 3 numbers are junk.
%
 a1 exch \tx@ArcAdjust % Angular position of base of arrow.
 /a2 ED
 pop pop pop
 a2 cos rx mul x add
 a2 sin ry mul y add
 a1 cos rx mul x add
 a1 sin ry mul y add
 % Now arrow tip coor and base coor are on stack.
 b pop pop pop pop    % Draw arrow, and discard coordinates.
 a2 CLW 8 div
% change value of d (test it by looking if `` 1 1 d '' gives 2 or not )
 1 1 d 2 eq { /d { sub } def } { /d { add } def } ifelse
 \tx@ArcAdjust
% resets original value of d
 1 1 d 2 eq { /d { sub } def } { /d { add } def } ifelse> % Adjust angle to give
    overlap.
%

%
% -------------- the arrow part -------------
%  the original table
% \def\pst@arrowtable{,<->,<<->>,>-<,>>-<<,(-),[-]}
\edef\pst@arrowtable{\pst@arrowtable,)-(,]-[,|>-<|}
%
\@namedef{psas@<|}{%
    \psk@tbarsize \tx@Tbar
    0 CLW 2 div T
    newpath
    true \psk@arrowinset \psk@arrowlength \psk@arrowsize \tx@Arrow%
}
% ]-[ arrow
\def\tx@BracketOut{BracketOut }
\@namedef{psas@[}{%
 /BracketOut {%
 CLW mul add dup CLW sub 2 div
%/x ED mul CLW add
```

```
 /x ED mul neg
 /y ED
 /z CLW 2 div def
 x neg y moveto
 x neg CLW 2 div L x CLW 2 div L x y L stroke 0 CLW moveto } def
 \psk@bracketlength \psk@tbarsize \tx@BracketOut
}
% )-( arrow
\def\tx@RoundBracketOut{RoundBracketOut }
\@namedef{psas@(}{%
 /RoundBracketOut {%
 CLW mul add dup 2 div
%/x ED mul
 /x ED mul neg
 /y ED
 /mtrx CM def
 0 CLW
 2 div T x y mul 0 ne { x y scale } if
 1 1 moveto
 .85 .5 .35 0 0 0 curveto
 -.35 0 -.85 .5 -1 1 curveto
 mtrx setmatrix stroke 0 CLW moveto } def
 \psk@rbracketlength \psk@tbarsize \tx@RoundBracketOut
}
%
% Redefininition of \psset@arrowscale to store value of X scale factor
\def\psset@arrowscale#1{\pst@getscale{#1}\psk@arrowscale}
\psset@arrowscale{1}

\def\psset@arrowscale#1{%
 \psset@arrowscale@i#1 \@nil
 \pst@getscale{#1}\psk@arrowscale%
}
\def\psset@arrowscale@i#1 #2\@nil{\edef\pst@arrowscale{#1}}
\def\pst@arrowscale{1}

% New parameter "arrowfill", with default as "true"
\newif\ifpsArrowFill
\def\psset@ArrowFill#1{\@nameuse{psArrowFill#1}}
\psset@ArrowFill{true}

% Modification of the PostScript macro Arrow to choose to fill or not the arrow
% (it require to restore the current linewidth, despite of the scaling)
\pst@def{Arrow}<{%
    CLW mul add dup 2 div
    /w ED mul dup
    /h ED mul
```

```
    /a ED { 0 h T 1 -1 scale } if
    gsave
    \ifpsArrowFill\else
 \pst@number\pslinewidth \pst@arrowscale\space div SLW
    \fi
    w neg h moveto
    0 0 L w h L w neg a neg rlineto
    \ifpsArrowFill gsave fill grestore \else gsave closepath stroke grestore \fi
 grestore
 0 h a sub moveto
}>
\@namedef{psas@>>}{%
    false \psk@arrowinset \psk@arrowlength \psk@arrowsize \tx@Arrow
    0 h a sub T
    gsave
    newpath
    false \psk@arrowinset \psk@arrowlength \psk@arrowsize \tx@Arrow
    CP
    grestore
    moveto
}
%
\@namedef{psas@<<}{%
    true \psk@arrowinset \psk@arrowlength \psk@arrowsize \tx@Arrow
    0 h neg a add T
    false \psk@arrowinset \psk@arrowlength \psk@arrowsize \tx@Arrow
    0 h a 5 mul 2 div sub moveto
}
%
% DG addition begin - Dec. 18/19, 1997 and Oct. 11, 2002
% Adapted from \psset@arrows
\def\psset@ArrowInside#1{%
    \begingroup
  \pst@activearrows
  \xdef\pst@tempg{<#1}%
    \endgroup
    \expandafter\psset@@ArrowInside\pst@tempg\@empty-\@empty\@nil
    \if@pst\else
  \@pstrickserr{Bad intermediate arrow specification: #1}\@ehpa
    \fi%
}
% Adapted from \psset@@arrows
\def\psset@@ArrowInside#1-#2\@empty#3\@nil{%
    \@psttrue
    \def\next##1,#1-##2,##3\@nil{\def\pst@tempg{##2}}%
    \expandafter\next\pst@arrowtable,#1-#1,\@nil
    \@ifundefined{psas@#2}%
```

```
  {\@pstfalse\def\psk@ArrowInside{}}%
  {\def\psk@ArrowInside{#2}}%
}
% Default value empty
\def\psk@ArrowInside{}
% Modified version of \pst@addarrowdef
\def\pst@addarrowdef{%
    \addto@pscode{%
  /ArrowA {
   \ifx\psk@arrowA\@empty
    \pst@oplineto
   \else
    \pst@arrowdef{A}
    moveto
   \fi
  } def
  /ArrowB {
   \ifx\psk@arrowB\@empty \else \pst@arrowdef{B} \fi
  } def
% DG addition
  /ArrowInside {
   \ifx\psk@ArrowInside\@empty \else \pst@arrowdefA{Inside} \fi
  } def
 }%
}
% Adapted from \pst@arrowdef
\def\pst@arrowdefA#1{%
 \ifnum\pst@repeatarrowsflag>\z@
  /Arrow#1c [ 6 2 roll ] cvx def Arrow#1c
 \fi
 \tx@BeginArrow
 \psk@arrowscale
 \@nameuse{psas@\@nameuse{psk@Arrow#1}}
 \tx@EndArrow%
}
% ArrowInsidePos parameter (default value 0.5)
\def\psset@ArrowInsidePos#1{\pst@checknum{#1}\psk@ArrowInsidePos}%
\psset@ArrowInsidePos{0.5}
%
% Modified version of \begin@ClosedObj
\def\begin@ClosedObj{%
    \leavevmode
    \pst@killglue
    \begingroup
 \use@par
 \solid@star
 \ifpsdoubleline \pst@setdoublesep \fi
```

```
 \pst@addarrowdef% DG addition
 \init@pscode
}
%
% Redefinition of the PostScript /Line macro to print the intermediate
% arrow on each segment of the line
%
\def\psset@ArrowInsideNo#1{\pst@checknum{#1}\psk@ArrowInsideNo}% hv 20031001
\def\psset@ArrowInsideOffset#1{\pst@checknum{#1}\psk@ArrowInsideOffset}% hv 20031001
\psset{ArrowInsideNo=1,ArrowInsideOffset=0}
%
\pst@def{Line}<{%
 NArray n 0 eq not { n 1 eq { 0 0 /n 2 def } if
    (\psk@ArrowInside) length 0 gt {
  2 copy /y1 ED /x1 ED ArrowA x1 y1
     /n n 1 sub def
     n {
   4 copy
   /y1 ED /x1 ED /y2 ED /x2 ED
   x1 y1
   \psk@ArrowInsidePos\space 1 gt{
    /Alpha y2 y1 sub x2 x1 sub atan def
    /ArrowPos \psk@ArrowInsideOffset\space def
    /Length x2 x1 sub y2 y1 sub Pyth def
    /dArrowPos \psk@ArrowInsidePos\space abs def
    {
     /ArrowPos ArrowPos dArrowPos add def
     ArrowPos Length gt { exit } if
     x1 Alpha cos ArrowPos mul add
     y1 Alpha sin ArrowPos mul add
     ArrowInside
     pop pop
    } loop
   }{
    /ArrowPos \psk@ArrowInsideOffset\space def
    /dArrowPos \psk@ArrowInsideNo 1 gt {%
     1.0 \psk@ArrowInsideNo 1.0 add div
    }{ \psk@ArrowInsidePos } ifelse def
    \psk@ArrowInsideNo\space cvi {
     /ArrowPos ArrowPos dArrowPos add def
     x2 x1 sub ArrowPos mul x1 add
     y2 y1 sub ArrowPos mul y1 add
     ArrowInside
     pop pop
    } repeat
   } ifelse
   pop pop Lineto
```

```
  } repeat
 }{ ArrowA /n n 2 sub def n { Lineto } repeat } ifelse
 CP 4 2 roll ArrowB L pop pop } if%
}>
%
% Redefinition of the PostScript /Polygon macro to print the intermediate
% arrow on each segment of the line
\pst@def{Polygon}<{%
 NArray n 2 eq { 0 0 /n 3 def } if
 n 3 lt {
  n { pop pop } repeat
 }{
  n 3 gt { CheckClosed } if
  n 2 mul
  -2 roll
  /y0 ED
     /x0 ED
     /y1 ED
     /x1 ED
     /xx1 x1 def
     /yy1 y1 def
     x1 y1
     /x1 x0 x1 add 2 div def
     /y1 y0 y1 add 2 div def
     x1 y1 moveto
     /n n 2 sub def
  /drawArrows {
   x11 y11
   \psk@ArrowInsidePos\space 1 gt {
    /Alpha y12 y11 sub x12 x11 sub atan def
    /ArrowPos \psk@ArrowInsideOffset\space def
    /Length x12 x11 sub y12 y11 sub Pyth def
    /dArrowPos \psk@ArrowInsidePos\space abs def
    {
     /ArrowPos ArrowPos dArrowPos add def
     ArrowPos Length gt { exit } if
     x11 Alpha cos ArrowPos mul add
     y11 Alpha sin ArrowPos mul add
     ArrowInside
     pop pop
    } loop
   }{
    /ArrowPos \psk@ArrowInsideOffset\space def
    /dArrowPos \psk@ArrowInsideNo 1 gt {%
     1.0 \psk@ArrowInsideNo 1.0 add div
    }{ \psk@ArrowInsidePos } ifelse def
     \psk@ArrowInsideNo cvi {
```

```
    /ArrowPos ArrowPos dArrowPos add def
    x12 x11 sub ArrowPos mul x11 add
    y12 y11 sub ArrowPos mul y11 add
         ArrowInside
    pop pop
   } repeat
  } ifelse
  pop pop Lineto
 } def
 n {
  4 copy
  /y11 ED /x11 ED /y12 ED /x12 ED
  drawArrows
 } repeat
 x1 y1 x0 y0
 6 4 roll
 2 copy
 /y11 ED /x11 ED /y12 y0 def /x12 x0 def
 drawArrows
 /y11 y0 def /x11 x0 def /y12 yy1 def /x12 xx1 def
 drawArrows
 pop pop
    closepath
} ifelse %
}>
%
%
% Redefinition of the PostScript /OpenBezier macro to print the intermediate
% arrow
\pst@def{OpenBezier}<{%
 /dArrowPos \psk@ArrowInsideNo 1 gt {%
  1.0 \psk@ArrowInsideNo 1.0 add div
 }{ \psk@ArrowInsidePos } ifelse def
 BezierNArray
 n 1 eq { pop pop
 }{ 2 copy
  /y0 ED /x0 ED
  ArrowA
  n 4 sub 3 idiv { 6 2 roll 4 2 roll curveto } repeat
  6 2 roll
  4 2 roll
  ArrowB
  /y3 ED /x3 ED /y2 ED /x2 ED /y1 ED /x1 ED
    /cx x1 x0 sub 3 mul def
  /cy y1 y0 sub 3 mul def
  /bx x2 x1 sub 3 mul cx sub def
  /by y2 y1 sub 3 mul cy sub def
```

```
/ax x3 x0 sub cx sub bx sub def
/ay y3 y0 sub cy sub by sub def
/getValues {
 ax t0 3 exp mul bx t0 t0 mul mul add cx t0 mul add x0 add
 ay t0 3 exp mul by t0 t0 mul mul add cy t0 mul add y0 add
 ax t 3 exp mul bx t t mul mul add cx t mul add x0 add
 ay t 3 exp mul by t t mul mul add cy t mul add y0 add
} def
/getdL {
 getValues
 3 -1 roll sub 3 1 roll sub Pyth
} def
/CurveLength {
 /u 0 def
 /du 0.01 def
 0 100 {
  /t0 u def
  /u u du add def
  /t u def
  getdL add
 } repeat } def
/GetArrowPos {
 /ende \psk@ArrowInsidePos\space 1 gt
  {ArrowPos}
  {ArrowPos CurveLength mul} ifelse def
 /u 0 def
 /du 0.01 def
 /sum 0 def
 {
  /t0 u def
  /u u du add def
     /t u def
  /sum getdL sum add def
  sum ende gt {exit} if
 } loop u
} def
/ArrowPos \psk@ArrowInsideOffset\space def
/loopNo \psk@ArrowInsidePos\space 1 gt {%
 CurveLength \psk@ArrowInsidePos\space div cvi
 }{ \psk@ArrowInsideNo } ifelse def
loopNo cvi {
 /ArrowPos ArrowPos dArrowPos add def
 /t GetArrowPos def
 /t0 t 0.95 mul def
 getValues
 ArrowInside pop pop pop pop
} repeat
```

```
  x1 y1 x2 y2 x3 y3 curveto
 } ifelse
}>
%
% Redefinition of the PostScript /NCLine macro to print the intermediate
% arrow of the line
\pst@def{NCLine}<{%
 NCCoor
 tx@Dict begin
 ArrowA CP 4 2 roll ArrowB
 4 copy
 /y2 ED /x2 ED /y1 ED /x1 ED
 x1 y1
 \psk@ArrowInsidePos\space 1 gt {
  /Alpha y2 y1 sub x2 x1 sub atan def
  /ArrowPos \psk@ArrowInsideOffset\space def
  /Length x2 x1 sub y2 y1 sub Pyth def
  /dArrowPos \psk@ArrowInsidePos\space abs def
  {%
   /ArrowPos ArrowPos dArrowPos add def
   ArrowPos Length gt { exit } if
   x1 Alpha cos ArrowPos mul add
   y1 Alpha sin ArrowPos mul add
   ArrowInside
   pop pop
  } loop
 }{%
  /ArrowPos \psk@ArrowInsideOffset\space def
  /dArrowPos \psk@ArrowInsideNo 1 gt {%
   1.0 \psk@ArrowInsideNo 1.0 add div
  }{ \psk@ArrowInsidePos } ifelse def
  \psk@ArrowInsideNo\space cvi {
   /ArrowPos ArrowPos dArrowPos add def
   x2 x1 sub ArrowPos mul x1 add
   y2 y1 sub ArrowPos mul y1 add
   ArrowInside
   pop pop
  } repeat
 } ifelse
 pop pop lineto pop pop
 end%
}>
%
\pst@def{NCCurve}<{%
 GetEdgeA GetEdgeB
 xA1 xB1 sub yA1 yB1 sub
 Pyth 2 div dup 3 -1 roll mul
```

```
/ArmA ED
mul
/ArmB ED
/ArmTypeA 0 def
/ArmTypeB 0 def
GetArmA GetArmB
xA2 yA2 xA1 yA1
2 copy
/y0 ED /x0 ED
tx@Dict begin
 ArrowA
end
xB2 yB2 xB1 yB1
tx@Dict begin
 ArrowB
end
/y3 ED /x3 ED /y2 ED /x2 ED /y1 ED /x1 ED
/cx x1 x0 sub 3 mul def
/cy y1 y0 sub 3 mul def
/bx x2 x1 sub 3 mul cx sub def
/by y2 y1 sub 3 mul cy sub def
/ax x3 x0 sub cx sub bx sub def
/ay y3 y0 sub cy sub by sub def
/getValues {
 ax t0 3 exp mul bx t0 t0 mul mul add cx t0 mul add x0 add
 ay t0 3 exp mul by t0 t0 mul mul add cy t0 mul add y0 add
 ax t 3 exp mul bx t t mul mul add cx t mul add x0 add
ay t 3 exp mul by t t mul mul add cy t mul add y0 add
} def
/getdL {
 getValues
 3 -1 roll sub 3 1 roll sub Pyth
} def
/CurveLength {
 /u 0 def
 /du 0.01 def
 0 100 {
  /t0 u def
  /u u du add def
  /t u def
  getdL add
 } repeat } def
/GetArrowPos {
 /ende \psk@ArrowInsidePos\space 1 gt {ArrowPos}{ArrowPos CurveLength mul} ifelse def
 /u 0 def
 /du 0.01 def
 /sum 0 def
```

```
  {
   /t0 u def
   /u u du add def
   /t u def
   /sum getdL sum add def
   sum ende gt {exit} if
  } loop u
 } def
 /dArrowPos \psk@ArrowInsideNo 1 gt {%
  1.0 \psk@ArrowInsideNo 1.0 add div
 }{ \psk@ArrowInsidePos } ifelse def
 /ArrowPos \psk@ArrowInsideOffset\space def
 /loopNo \psk@ArrowInsidePos\space 1 gt {%
  CurveLength \psk@ArrowInsidePos\space div cvi
  }{ \psk@ArrowInsideNo } ifelse def
 loopNo cvi {
  /ArrowPos ArrowPos dArrowPos add def
  /t GetArrowPos def
  /t0 t 0.95 mul def
  getValues
  ArrowInside pop pop pop pop
 } repeat
 x1 y1 x2 y2 x3 y3 curveto
 /LPutVar [ xA1 yA1 xA2 yA2 xB2 yB2 xB1 yB1 ] cvx def
 /LPutPos { t LPutVar BezierMidpoint } def
 /HPutPos { { HPutLines } HPutCurve } def
 /VPutPos { { VPutLines } HPutCurve } def
}>
%
\def\psset@dashColorI#1{\pst@getcolor{#1}\psDashColorI}
\def\psset@dashColorII#1{\pst@getcolor{#1}\psDashColorII}
\define@key{psset}{dashNo}{\edef\psk@dashNo{#1}}
%
\define@key{psset}{linecap}{\edef\psk@linecap{#1}}
\psset{dashColorI=black,dashColorII=red,dashNo=0.2,linecap=0}
%
\pst@def{LineII}<{%
 NArray
 /n n 1 sub def
 /y1 ED /x1 ED x1 y1 ArrowA x1 y1 moveto
 \psk@linecap\space 3 gt
  {\psk@linecap\space 0 lt {0 setlinecap} if }
  {\psk@linecap\space setlinecap} ifelse
 n {
  /y2 ED /x2 ED
  /y0 y1 def /x0 x1 def
  /length x2 x1 sub y2 y1 sub Pyth def
```

```
  \psk@dashNo\space 1.0 lt
   {/cntMax 1.0 \psk@dashNo\space div .49 add cvi def}
   {/cntMax length \psk@dashNo\space div .49 add cvi def} ifelse
  x2 x1 sub cntMax div /dx ED
  y2 y1 sub cntMax div /dy ED
  /cnt 0 def
  cntMax {
   gsave
   /x1 x1 dx add def
   /y1 y1 dy add def
   x1 y1
   cnt 2 mod 0 eq
    { \pst@usecolor\psDashColorI }
    { \pst@usecolor\psDashColorII } ifelse
   lineto stroke
   /cnt cnt 1 add def
   grestore
   x1 y1 moveto
  } repeat
  /y1 y2 def /x1 x2 def
 } repeat
 x0 y0 x2 y2 ArrowB L pop pop
}>
%
\def\pslineII{\pst@object{pslineII}}
\def\pslineII@i{%
    \pst@getarrows{%
        \begin@OpenObj
        \pst@getcoors[\pslineII@ii%
    }%
}
\def\pslineII@ii{%
 \addto@pscode{%
  \pst@cp
  \ifshowpoints true \else false \fi
  \tx@LineII
 }%
 \end@OpenObj%
}
%
\newdimen\pswBegin\newdimen\pswEnd
\def\psset@wBegin#1{\pssetlength\pswBegin{#1}}
\def\psset@wEnd#1{\pssetlength\pswEnd{#1}}
\setkeys{psset}{wBegin=\pslinewidth, wEnd=\pslinewidth}
%
\pst@def{LineIII}<{%
 NArray
```

```
 /n n 1 sub def
 /YA ED /XA ED
 /bA2 \pst@number\pswBegin\space 2.0 div def
 /bB2 \pst@number\pswEnd\space 2.0 div def
 /xStep bB2 bA2 sub n div def
 0 setlinecap
 n {
  XA YA moveto
  /YB ED /XB ED
  /Alpha YB YA sub XB XA sub atan def
  /Alphasin Alpha sin def
  /Alphacos Alpha cos def
        \pst@usecolor\pslinecolor
  0.01 SLW
  /bB2 bA2 xStep add def
  XA Alphasin bA2 mul sub
  YA Alphacos bA2 mul add lineto
  XB Alphasin bB2 mul sub
  YB Alphacos bB2 mul add lineto
  XB Alphasin bB2 mul add
  YB Alphacos bB2 mul sub lineto
  XA Alphasin bA2 mul add
  YA Alphacos bA2 mul sub lineto
  XA YA lineto fill
  /XA XB def /YA YB def
  /bA2 bB2 def
 } repeat
}>
%
\def\pslineIII{\pst@object{pslineIII}}
\def\pslineIII@i{%
    \pst@getarrows{%
        \begin@OpenObj
        \pst@getcoors[\pslineIII@ii%
    }%
}
\def\pslineIII@ii{%
 \addto@pscode{%
  \pst@cp
  \ifshowpoints true \else false \fi
  \tx@LineIII
 }%
 \end@OpenObj%
}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%                                                          %%%%%%%%%%
%%%%%%%%%%              pst-node                                    %%%%%%%%%%
```

```
%%%%%%%%%%                                                          %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
\pst@def{NCLineII}<{
    NCCoor
 /y1 ED /x1 ED x1 y1 ArrowA x1 y1 moveto
    /y2 ED /x2 ED
    /y0 y1 def /x0 x1 def
    /length x2 x1 sub y2 y1 sub Pyth def
    \psk@dashNo\space 1.0 lt
     {/cntMax 1.0 \psk@dashNo\space div .49 add cvi def}
 {/cntMax length \psk@dashNo\space div .49 add cvi def} ifelse
    x2 x1 sub cntMax div /dx ED
    y2 y1 sub cntMax div /dy ED
    /cnt 0 def
    cntMax {
  gsave
  /x1 x1 dx add def
  /y1 y1 dy add def
  x1 y1
  cnt 2 mod 0 eq
      { \pst@usecolor\psDashColorI }
      { \pst@usecolor\psDashColorII } ifelse
  lineto stroke
  /cnt cnt 1 add def
  grestore
  x1 y1 moveto
    } repeat
    x0 y0 x2 y2 ArrowB L pop pop%
}>
%
\def\nclineII{\pst@object{nclineII}}%
\def\nclineII@i{\check@arrow{\nclineII@ii}}%
\def\nclineII@ii#1#2{\nc@object{Open}{#1}{#2}{.5}%
    {\tx@NCLineII /LPutPos { xB yB xA yA \tx@LPutLine } def}%
}%
\def\pclineII{\pst@object{pclineII}}%
\def\pclineII@i{\pc@object\nclineII@ii}%
%
\def\psset@lineAngle#1{\psset@armB{0.5}\edef\psk@lineAngle{#1}}%
\psset@lineAngle{0}%
%
\pst@def{NCDiag}<{
 GetEdgeA GetEdgeB GetArmA GetArmB mark
 \psk@lineAngle\space abs 0 gt {
  /xTemp xA2 10 add def
  /yTemp yA2 \psk@lineAngle\space dup sin exch cos div 10 mul add def
```

```
  /dY1 yTemp yA2 sub def
  /dX1 xTemp xA2 sub def
  /dY2 yB2 yB1 sub def
  /dX2 xB2 xB1 sub def
  dX1 abs 0.01 lt {
   /m2 dY2 dX2 div def
   /xB2 xA2 def
   /yB2 xA2 xB1 sub m2 mul yB1 add def
     }{
   dX2 abs 0.01 lt {
    /m1 dY1 dX1 div def
    /xB2 xB1 def
    /yB2 xB1 xA2 sub m1 mul yA2 add def
        }{%
    /m1 dY1 dX1 div def
    /m2 dY2 dX2 div def
    /xB2 m1 xA2 mul m2 xB1 mul sub yA2 sub yB1 add m1 m2 sub div def
    /yB2 xB2 xA2 sub m1 mul yA2 add def
   } ifelse
  } ifelse
 } if
 ArmB 0 ne { xB1 yB1 } if
 xB2 yB2
 xA2 yA2
 ArmA 0 ne { xA1 yA1 } if
 tx@Dict begin false Line end
 /LPutVar [ xB1 yB1 xB2 yB2 xA2 yA2 xA1 yA1 ] cvx def
 /LPutPos { LPutLines } def
 /HPutPos { HPutLines } def
 /VPutPos { VPutLines } def
}>
% hv 2003-12-22
\pst@def{NCDiagg}<{
 GetEdgeA GetArmA \psk@lineAngle\space abs 0 gt { \psk@lineAngle\space }
  { yB yA2 sub xB xA2 sub Atan 180 add } ifelse /AngleB ED
 GetEdgeB mark
 \psk@lineAngle\space abs 0 gt {
  /dY2 yA2 yA1 sub def
  /dX2 xA2 xA1 sub def
  \psk@lineAngle\space abs 90 eq {
   /m2 dY2 dX2 div def
   /yA2 xB xA2 sub m2 mul yA2 add def
   /xA2 xB def
  }{
   /m1 \psk@lineAngle\space dup sin exch cos div def % tan alpha
   dX2 abs 0.01 lt {
    /yA2 xA1 xB sub m1 mul yB add def
```

```
    /xA2 xA1 def
   }{%
    /m2 dY2 dX2 div def
    /xA2 m1 xB mul m2 xA2 mul sub yA2 add yB sub m1 m2 sub div def
    /yA2 xA2 xB sub m1 mul yB add def
   } ifelse
  } ifelse
 } if
 xB1 yB1 xA2 yA2
 ArmA 0 ne { xA1 yA1 } if
 tx@Dict begin false Line end
 /LPutVar [ xB1 yB1 xA2 yA2 xA1 yA1 ] cvx def
 /LPutPos { LPutLines } def
 /HPutPos { HPutLines } def
 /VPutPos { VPutLines } def
}>
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%                                                          %%%%%%%%%%
%%%%%%%%%%              pst-plot                                    %%%%%%%%%%
%%%%%%%%%%                                                          %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
\newif\ifPst@plot@comma
\define@key{psset}{comma}[true]{\@nameuse{Pst@plot@comma#1}}
%
\newif\ifPst@plot@xAxes
\newif\ifPst@plot@yAxes
\newif\ifPst@plot@xyAxes
\define@key{psset}{xAxes}[true]{\@nameuse{Pst@plot@xAxes#1}}
\define@key{psset}{yAxes}[true]{\@nameuse{Pst@plot@yAxes#1}}
\define@key{psset}{xyAxes}[true]{\psset{xAxes=true,yAxes=true}}%
\psset{xyAxes=true}
%
\def\psset@xyDecimals#1{\psset@xDecimals{#1}\psset@yDecimals{#1}}
\define@key{psset}{xDecimals}{\def\psk@xDecimals{#1}}
\define@key{psset}{yDecimals}{\def\psk@yDecimals{#1}}
\psset{xDecimals={}, yDecimals={}}%
%
\def\psset@xyLabel#1{\psset{xLabel=#1, yLabel=#1}}
\def\psset@xLabel#1{\def\psk@xLabel{#1}}
\def\psset@yLabel#1{\def\psk@yLabel{#1}}
\psset@xLabel{}\psset@yLabel{}
%
\def\psset@xlogBase#1{\edef\psk@xlogBase{#1}}
\def\psset@ylogBase#1{\edef\psk@ylogBase{#1}}
\def\psset@logBase#1{\psset@xlogBase{#1}\psset@ylogBase{#1}}%
```

```
\psset@xlogBase{}\psset@ylogBase{}
%
%logLines=all|x|y|none (0,1,2,3)
\def\psset@logLines#1{\pst@expandafter\psset@@logLines{#1}\@nil\psk@logLines}
\def\psset@@logLines#1#2\@nil#3{%
  \ifx#1a\let#3\z@\else
    \ifx#1x\let#3\@ne\else
      \ifx#1y\let#3\tw@\else
        \ifx#1n\let#3\thr@@\else
          \@pstrickserr{Bad argument: `#1#2'}\@ehpa
  \fi\fi\fi\fi}
\psset@logLines{none}
%
%
\newcount\@zero\@zero=0
%
\def\pshlabel#1{%
 \edef\@xyDecimals{\psk@xDecimals}%
 \psk@xLabel%
 \ifnum\psk@ticks<\tw@ % ticks=all|x
  \ifx\psk@xlogBase\@empty
   \expandafter\@LabelComma#1..\@nil%
  \else%
   {\psk@xLabel\psk@xlogBase\textsuperscript{#1}}%
  \fi%
    \fi%
}
\def\psvlabel#1{%
 \edef\@xyDecimals{\psk@yDecimals}%
 \psk@yLabel%
 \ifodd\psk@ticks % ticks=all||y (0,2)
 \else
  \ifx\psk@ylogBase\@empty
   \expandafter\@LabelComma#1..\@nil%
  \else%
   {\psk@yLabel\psk@ylogBase\textsuperscript{#1}}%
  \fi%
    \fi%
}
\newcount\@digitcounter\@digitcounter=0\relax
\def\@inc@digitcounter{\global\advance\@digitcounter by 1\relax}
\def\@get@digitcounter{\the\@digitcounter\relax}
\def\@Reset@digitcounter{\global\@digitcounter=0\relax}
\def\@zeroFill{%
 \ifnum \@xyDecimals>\@get@digitcounter
  \bgroup
   0\@inc@digitcounter\@zeroFill
```

```
  \egroup%
 \fi%
}
% #1 the value, maybe empty
%
\def\@process@digits#1#2;{%
 \ifx *#1\@zeroFill\else#1\@inc@digitcounter\@process@digits#2;\fi%
}
\def\@writeDecimals#1{%
 \ifx\@xyDecimals\@empty% take value as is
  \def\@tempa{#1}% write only if not empty
  \ifx\@tempa\@empty% write nothing
  \else
   \ifPst@plot@comma,\else.\fi%
   #1%
  \fi%
 \else% write only \xy@decimals
  \ifnum\@xyDecimals>\@zero
   \ifPst@plot@comma,\else.\fi
   \@Reset@digitcounter
   \@process@digits#1*;
  \fi%
 \fi%
}
%% #1 integer
%% #2 decimals
%% #3 dot
\def\@LabelComma#1.#2.#3\@nil{%
 \def\dummy{#1}%
 \ifx\dummy\@empty\the\@zero\else#1\fi% the integer part
 \def\dummy{#2}%
 \ifx\dummy\@empty\@writeDecimals{}\else\@writeDecimals{#2}\fi%
}
%
% \pst@ticks{angle}{dx}{n}{int}
% int=1 if ticks appear on top of axes, 0 otherwise.
\def\pst@ticks#1#2#3#4{%
  \begin@SpecialObj%
    \addto@pscode{%
      #1 rotate
      /n #3 def
      /dx #2 def
      n 0 lt { /dx dx neg def /n n neg def } if
      /y2 \psk@ticksize CLW 2 div add def
      /y1 y2 neg def
      \ifnum\psk@tickstyle=1
        \ifdim#4<\z@ /y2 \else /y1 \fi 0 def
```

```
      \else%
        \ifnum\psk@tickstyle=-1%
          \ifdim#4<\z@ /y1 \else /y2 \fi 0 def
        \fi
      \fi
      /x 0 def % original was /x dx def, but we need the first tick
      n 1 add { x y1 moveto x y2 lineto stroke /x x dx add def } repeat}%
  \end@SpecialObj%
}
%
\def\psaxes{\def\pst@par{}\pst@object{psaxes}}
\def\psaxes@i{\pst@getarrows\psaxes@ii}
\def\psaxes@ii(#1){\@ifnextchar({\psaxes@iii(#1)}{\psaxes@iv(0,0)(0,0)(#1)}}
\def\psaxes@iii(#1)(#2){%
  \@ifnextchar(%
    {\psaxes@iv(#1)(#2)}%
    {\psaxes@iv(#1)(#1)(#2)}}
\def\psaxes@iv(#1,#2)(#3,#4)(#5,#6){%
  \setbox\pst@hbox=\hbox\bgroup
    \use@par
    \pssetxlength\pst@dimg{#1}% o-x
    \pssetylength\pst@dimh{#2}% o-y
    \pssetxlength\pst@dima{#3}% bl-x
    \pssetylength\pst@dimb{#4}% bl-y
    \pssetxlength\pst@dimc{#5}% ur-x
    \pssetylength\pst@dimd{#6}% ur-y
% Whole thing will be translated to origin:
    \advance\pst@dima-\pst@dimg % Dist. from bl-x to o-x
    \advance\pst@dimb-\pst@dimh % Dist. from bl-y to o-y
    \advance\pst@dimc-\pst@dimg % Dist. from ur-x to o-x
    \advance\pst@dimd-\pst@dimh % Dist. from ur-y to o-y
% Make lines/arrows or frame:
    \@nameuse{psxs@\psk@axesstyle}%
% "\pslabelsep" should be from the edge of the axis.
    \advance\pslabelsep.5\pslinewidth
% Now the ticks and labels. Start by checking for "\multido".
% !!Need to fix this so that does nothing when there are 0 ticks.!!
 \ifPst@plot@xAxes
    \begingroup
      \ifdim\pst@dimb=\z@\else\showoriginfalse\fi
      \ifnum\psk@dx=\z@
        \pst@dimg=\psk@Dx\psxunit
        \edef\psk@dx{\number\pst@dimg}%
      \fi
      \ifnum\psk@ticks<\tw@
        \ifnum\psk@tickstyle>\z@\else
          \advance\pslabelsep\psk@ticksize\p@
```

```
        \fi
      \fi
      \pst@hlabels\pst@dimc\psk@arrowB
      \pst@hlabels\pst@dima\psk@arrowA
    \endgroup%
 \fi%
 \ifPst@plot@yAxes
 \begingroup
      \ifdim\pst@dima=\z@\else\showoriginfalse\fi
      \ifnum\psk@dy=\z@
         \pst@dimg=\psk@Dy\psyunit
         \edef\psk@dy{\number\pst@dimg}%
      \fi
      \ifodd\psk@ticks\else
        \ifnum\psk@tickstyle>\z@\else
          \advance\pslabelsep\psk@ticksize\p@
        \fi
      \fi
      \pst@vlabels\pst@dimd\psk@arrowB
      \pst@vlabels\pst@dimb\psk@arrowA
    \endgroup%
 \fi%
% Now close "\pst@hbox" (which is 0-dimensional), and put it at the origin.
  \egroup
  \pssetxlength\pst@dimg{#1}%
  \pssetylength\pst@dimh{#2}%
  \leavevmode\psput@cartesian\pst@hbox
  \ignorespaces%
}

\def\psxs@axes{%
  \ifPst@plot@xAxes\psxs@@axes\pst@dima\pst@dimc{}\fi% the x-Axes
  \ifPst@plot@yAxes\psxs@@axes\pst@dimb\pst@dimd{exch}\fi%
}
%
\def\psaxes@iv(#1,#2)(#3,#4)(#5,#6){%
 \setbox\pst@hbox=\hbox\bgroup%
 \use@par%
 \pssetxlength\pst@dimg{#1}% o-x
 \pssetylength\pst@dimh{#2}% o-y
 \pssetxlength\pst@dima{#3}% bl-x
 \pssetylength\pst@dimb{#4}% bl-y
 \pssetxlength\pst@dimc{#5}% ur-x
 \pssetylength\pst@dimd{#6}% ur-y
% D.G. modification begin - Apr. 6, 1998
    % If minimum values are negative in log mode, we modify Ox
    % (respectively Oy) if this was not done by the user
```

```
    % X axis labels (\psk@log = 0 or 1)
 \ifx\psk@xlogBase\@empty\else%
  \ifdim\psk@Ox pt=\z@
   \ifdim#3pt<\z@
    \pssetxlength\pst@dimg{#3}% o-x
    \psset@Ox{#3}%
   \fi%
  \fi%
 \fi%
% Y axis labels (\psk@log = 0 or 2)
 \ifx\psk@ylogBase\@empty\else%
  \ifdim#4pt<\z@
   \ifdim\psk@Oy pt=\z@
    \pssetylength\pst@dimh{#4}% o-y
    \psset@Oy{#4}%
   \fi
  \fi
 \fi
% D.G. modification end
% Whole thing will be translated to origin:
 \advance\pst@dima-\pst@dimg % Dist. from bl-x to o-x
 \advance\pst@dimb-\pst@dimh % Dist. from bl-y to o-y
 \advance\pst@dimc-\pst@dimg % Dist. from ur-x to o-x
 \advance\pst@dimd-\pst@dimh % Dist. from ur-y to o-y
% Make lines/arrows or frame:
 \@nameuse{psxs@\psk@axesstyle}%
% "\pslabelsep" should be from the edge of the axis.
 \advance\pslabelsep.5\pslinewidth%
 \begingroup
  \ifdim\pst@dima=\z@\else\showoriginfalse\fi
  \ifnum\psk@dy=\z@
   \pst@dimg=\psk@Dy\psyunit
   \edef\psk@dy{\number\pst@dimg}%
  \fi%
  \ifodd\psk@ticks\else%
   \ifnum\psk@tickstyle>\z@\else%
    \advance\pslabelsep\psk@ticksize\p@%
   \fi%
  \fi%
%  \pst@vlabels\pst@dimd\psk@arrowB
%  \pst@vlabels\pst@dimb\psk@arrowA
  \pst@vlabels{\pst@dimd}{\psk@arrowB}{#3}{#5}%
  \pst@vlabels{\pst@dimb}{\psk@arrowA}{#3}{#5}%
 \endgroup%
 \begingroup%
  \ifdim\pst@dimb=\z@\else\showoriginfalse\fi%
  \ifnum\psk@dx=\z@%
```

```
    \pst@dimg=\psk@Dx\psxunit%
    \edef\psk@dx{\number\pst@dimg}%
   \fi%
   \ifnum\psk@ticks<\tw@
    \ifnum\psk@tickstyle>\z@\else%
     \advance\pslabelsep\psk@ticksize\p@%
    \fi%
   \fi%
%  \pst@hlabels\pst@dimc\psk@arrowB
%  \pst@hlabels\pst@dima\psk@arrowA
   \pst@hlabels{\pst@dimc}{\psk@arrowB}{#4}{#6}%
   \pst@hlabels{\pst@dima}{\psk@arrowA}{#4}{#6}%
  \endgroup%
% Now close "\pst@hbox" (which is 0-dimensional), and put it at the origin.
  \egroup%
  \pssetxlength\pst@dimg{#1}%
  \pssetylength\pst@dimh{#2}%
  \leavevmode\psput@cartesian\pst@hbox%
  \ignorespaces%
}
%
\def\pst@hlabels#1#2#3#4{%
 \ifdim#1=\z@\else%
  \ifx#2\empty\else%
   \advance#1\ifdim#1>\z@-\fi7\pslinewidth%
  \fi%
  \pst@cnta=#1\relax%             % Distance (in sp) to end.
  \divide\pst@cnta\psk@dx\relax%  % Number of ticks/labels
  \ifnum\pst@cnta=\z@\else%
   \pst@dimb=\psk@dx sp%          % Space between ticks.
   \ifnum\psk@ticks<\tw@%
    \ifPst@plot@xAxes\pst@ticks{0}{\pst@number\pst@dimb}%
     {\the\pst@cnta}{\pst@dimd}\fi%
   \fi%
   \ifPst@plot@yAxes\else\showoriginfalse\fi%
   \ifnum\psk@labels<\tw@ \ifPst@plot@xAxes\pst@@hlabels\fi\fi%
   \showoriginfalse%
   \ifnum\psk@logLines<\tw@
    \pst@cntb=\pst@cnta%
    \ifnum\pst@cnta<\z@
     \multiply\pst@cntb\m@ne%
    \fi%
    \psset{arrows=-,linewidth=0.1pt,linecolor=lightgray}%
    \addto@pscode{%
     /logn { log 1 sub } def%
     /yT3 { #3\space \psk@Oy\space sub } def
     /yT4 { #4\space \psk@Oy\space sub } def
```

```
    }%
    \multips(\ifnum\pst@cnta<\z@-\fi\pst@dimb,0)%
     (\ifnum\pst@cnta<\z@-\fi\pst@dimb,0){\pst@cntb}{%\@nil
      \psline(!0 yT3)(! 0 yT4)%
      \psline(! 2 logn yT3)(! 2 logn yT4)%
      \psline(! 3 logn yT3)(! 3 logn yT4)%
      \psline(! 4 logn yT3)(! 4 logn yT4)%
      \psline(! 5 logn yT3)(! 5 logn yT4)%
     }%
   \fi%
  \fi%
 \fi%
}
%
\def\pst@vlabels#1#2#3#4{%
 \ifdim#1=\z@\else%
  \ifx#2\empty\else%
   \advance#1\ifdim#1>\z@-\fi7\pslinewidth%
  \fi%
  \pst@cnta=#1\relax       %     % Distance (in sp) to end.
  \divide\pst@cnta\psk@dy\relax % % Number of ticks/labels
  \ifnum\pst@cnta=\z@\else%
   \pst@dima=\psk@dy sp%         % Space between ticks.
   \ifodd\psk@ticks\else%
    \ifPst@plot@yAxes\pst@ticks{90}{\pst@number\pst@dima}%
     {\the\pst@cnta}{-\pst@dimc}\fi%
   \fi%
   \ifPst@plot@xAxes\else\showorigintrue\fi%
   \ifodd\psk@labels\else\ifPst@plot@yAxes\pst@@vlabels\fi\fi%
   \showoriginfalse%
   \ifodd\psk@logLines\else%
    \pst@cntb=\pst@cnta%
    \ifnum\pst@cnta<\z@%
     \multiply\pst@cntb\m@ne%
    \fi%
    \psset{arrows=-,linewidth=0.1pt,linecolor=lightgray}%
    \addto@pscode{%
     /logn { log 1 sub } def%
     /xT3 { #3\space \psk@Ox\space sub } def
     /xT4 { #4\space \psk@Ox\space sub } def
    }%
    \multips(0,\ifnum\pst@cnta<\z@-\fi\pst@dima)%
     (0,\ifnum\pst@cnta<\z@-\fi\pst@dima){\pst@cntb}{%\@nil
      \psline(! xT3 0)(! xT4 0)%
      \psline(! xT3 2 logn)(! xT4 2 logn)%
      \psline(! xT3 3 logn)(! xT4 3 logn)%
      \psline(! xT3 4 logn)(! xT4 4 logn)%
```

```
      \psline(! xT3 5 logn)(! xT4 5 logn)%
     }%
    \fi%
   \fi%
  \fi%
}
%
%
\def\psset@nStep#1{\edef\psk@nStep{#1}}
\def\psset@nStart#1{\edef\psk@nStart{#1}}
\def\psset@nEnd#1{\edef\psk@nEnd{#1}}
\def\psset@xStep#1{\edef\psk@xStep{#1}}
\def\psset@yStep#1{\edef\psk@yStep{#1}}
%
\def\psset@xStart#1{\edef\psk@xStart{#1}}
\def\psset@xEnd#1{\edef\psk@xEnd{#1}}
\def\psset@yStart#1{\edef\psk@yStart{#1}}
\def\psset@yEnd#1{\edef\psk@yEnd{#1}}
%
\def\psset@plotNo#1{\edef\psk@plotNo{#1}}
\def\psset@plotNoMax#1{\edef\psk@plotNoMax{#1}}
%
\psset{nStep=1, nStart=0, nEnd={},%
 xStep=0, yStep=0, xStart={}, xEnd={}, yStart={}, yEnd={}, comma=false,%
 plotNo=1,plotNoMax=1}
%
\def\listplot@ii#1{%
 \@nameuse{beginplot@\psplotstyle}%
 \addto@pscode{/D {} def mark}%
 #1%
 \addto@pscode{%
  \tx@PreparePoints
  \pst@number\psxunit
  \pst@number\psyunit
  \tx@ScalePoints%
 }%
 \@nameuse{endplot@\psplotstyle}%
}
%
\pst@def{PreparePoints}<{%
 counttomark /m exch def
 /n m \psk@plotNoMax\space 1 add div cvi def
 \psk@plotNoMax\space 1 gt {% multiple data files?
  n {
   \psk@plotNoMax\space \psk@plotNo\space 1 sub neg roll % x yNo y y y ...
   \psk@plotNoMax\space 1 sub { pop } repeat % x yNo
   /m m \psk@plotNoMax\space 1 sub sub def
```

```
   m 2 roll
  } repeat
 } if % no multiple data files
% counttomark /m exch def
% /n m 2 div cvi def
 /xMax -99999 def /yMax -99999 def
 /xP 0 def /yP 0 def
 m copy
 n {
  /y exch def /x exch def
  xMax x lt { /xMax x def } if
  yMax y lt {/yMax y def } if
  xP x gt { /xP x def } if
  yP y gt { /yP y def } if
 } repeat
% m 2 roll
 \psk@xStep\space 0 gt \psk@yStep\space 0 gt or (\psk@xStart) length 0 gt or
  (\psk@yStart) length 0 gt or (\psk@xEnd) length 0 gt or (\psk@yEnd) length 0 gt or {
%
  (\psk@xStart) length 0 gt {\psk@xStart\space }{ xP } ifelse /xStart exch def
  (\psk@yStart) length 0 gt {\psk@yStart\space }{ yP } ifelse /yStart exch def
  (\psk@xEnd) length 0 gt { \psk@xEnd\space }{ xMax } ifelse /xEnd exch def
  (\psk@yEnd) length 0 gt { \psk@yEnd\space }{ yMax } ifelse /yEnd exch def
  n {
   m -2 roll
   2 copy /yVal exch def /xVal exch def
   xVal xP ge
   yVal yP ge and
   xVal xEnd le and
   yVal yEnd le and
   xVal xStart ge and
   yVal yStart ge and {
    /xP xP \psk@xStep\space add def
    /yP yP \psk@yStep\space add def
   }{%
    pop pop
    /m m 2 sub def
   } ifelse
  } repeat
 }{%
  /ncount 0 def
  (\psk@nEnd) length 0 gt { \psk@nEnd\space }{ m } ifelse /nEnd exch def
  n {
   m -2 roll
   \psk@nStep\space 1 gt {
    ncount \psk@nStep\space mod 0 eq }{ true } ifelse
   ncount nEnd le and
```

```
   ncount \psk@nStart\space ge and not {
    pop pop
    /m m 2 sub def
   } if
   /ncount ncount 1 add def
  } repeat
 } ifelse
}>
%
\def\psIVaxes{\@ifnextchar[{\psIVaxes@i}{\psIVaxes@i[]}}
\def\psIVaxes@i[#1](#2,#3)(#4,#5){%
 \edef\@psxMax{#3}
 \edef\@psyMax{#5}
 \psaxes[#1](#2,#3)(#4,#5)
 \@ifnextchar[{\psIVaxes@ii}{\psIVaxes@ii[]}%
}
\def\psIVaxes@ii[#1](#2,#3){%
 \psset{#1}
 \pst@ticks{90}{#2}{5}{1}
% \pst@ticks{angle}{dx}{n}{int}
% int=1 if ticks appear on top of axes, 0 otherwise.
  \ignorespaces
 \@ifnextchar[{\psIVaxes@iii}{\psIVaxes@iii[]}%
}
\def\psIVaxes@iii[#1](#2,#3){{%
}}
%
\newcount\linecnt
\begingroup
\catcode`\,=13
\catcode`\_=13
\gdef\savedata@#1[#2]{%
  \xdef\pst@tempg{#2_}%
  \endgroup
  \let#1\pst@tempg
  \global\let\pst@tempg\relax
  \ignorespaces}
\gdef\readdata@{%
%  \typeout{No: \the\datacnt}
  \read1 to \pst@tempa
  \ifnum\linecnt=\psk@nStep
%    \typeout{reached nStep --> \psk@nStep}
%    \typeout{tempa --> \pst@tempa{}}
    \global\linecnt=0
    \expandafter\readdata@@\pst@tempa_\@nil
  \fi
  \global\advance\linecnt by 1
```

```
  \ifeof1\else\expandafter\readdata@\fi}
\gdef\pst@@readfile#1#2\@nil{\addto@pscode{,#1#2}}%
\gdef\readdata@@#1#2\@nil{\xdef\pst@tempg{\pst@tempg,#1#2}}%
\endgroup

\def\readdata{\@ifnextchar[{\readdata@i}{\readdata@i[]}}
\def\readdata@i[#1]#2#3{{%
  \def\pst@tempa{#1}
  \ifx\pst@tempa\@empty\else\psset{#1}\fi
  \openin1=#3
  \begingroup
  \edef\pst@tempg{}%
  \ifeof1
    \@pstrickserr{Data file `#3' not found.}\@ehpa
  \else
    \pst@datadelimiters
    \catcode`\[=1
    \catcode`\]=2
    \global\linecnt=\psk@nStep
    \readdata@%
  \fi
  \endgroup
  \global\let#2\pst@tempg
  \global\let\pst@tempg\relax
  \ignorespaces%
}}
%
%
\def\psset@coeff#1{\edef\psk@coeff{#1}}
\psset{coeff=1} % coeff=a0 a1 a2 a3 ...
%
\def\psplotPolynomial{\@ifnextchar[{\psplotPolynomial@i}{\psplotPolynomial@i[]}}
\def\psplotPolynomial@i[#1]#2#3{{%
 \psset{#1}
 \psplot{#2}{#3}{%
  /Horner {
   aload length
   dup 2 add -1 roll
   exch 1 sub {
    dup 4 1 roll
    mul add exch
   } repeat
   pop
  } def
  x [\psk@coeff] Horner
 }
}}
```

```
%
\def\resetOptions{%
    \@zero=0%
    \psset{%
%%%% pstricks %%%%%%%%
 unit=1cm,%
 swapaxes=false,%
 showpoints=false,%
 border=0pt, bordercolor=white,%
 doubleline=false, doublesep=1.25\pslinewidth,%
 doublecolor=white,%
 shadow=false, shadowsize=3pt, shadowangle=-45, shadowcolor=darkgray,%
 linewidth=.8pt,%
 linecolor=black,%
 dash=5pt 3pt, dashadjust=true,%
 dotsep=3pt,%
 linestyle=solid,%
 fillcolor=white,%
 hatchwidth=.8pt, hatchsep=4pt, hatchcolor=black, hatchangle=45,%
 fillstyle=none,%
 arrows=-, arrowscale=1, arrowsize=1.5pt 2, arrowlength=1.4, arrowinset=.4,%
 tbarsize=2pt 5,%
 bracketlength=.15, rbracketlength=.15,%
 liftpen=0, linetype=0,%
 gangle=0,%
 curvature=1 .1 0,%
 dotsize=2pt 2,%
 dotangle=0, dotscale=1, dotstyle=*,%
 linearc=0pt,%
 framearc=0,%
 cornersize=relative,%
 dimen=outer,%
 gridwidth=.8pt, griddots=0, gridcolor=black,%
 subgridwidth=.4pt, subgridcolor=gray, subgriddots=0, subgriddiv=5,%
 gridlabels=10pt, gridlabelcolor=black,%
 framesep=3pt, boxsep=true,%
 trimode=U,%
 arcsep=0,radius=.25cm,%
 ref=c,rot=0,labelsep=5pt,refangle=0,%
%%%%%%%% pst-plot %%%%%%%%%%%%
 plotstyle=line,plotpoints=50,%
 ticksize=3pt,tickstyle=full,ticks=all,%
 labels=all,Ox=0,Dx=1,dx=0,Oy=0,Dy=1,dy=0,%
 showorigin=true,%
 axesstyle=axes,%
 intSeparator={,}%
 braceWidth=0.35,bracePos=0.5,%
```

```
 arrowscale=1,%
 ArrowFill=true,%
 ArrowInsidePos=0.5,%
 ArrowInsideNo=1,ArrowInsideOffset=0,%
 dashColorI=black,dashColorII=red,dashNo=0.2,linecap=0,%
 wBegin=\pslinewidth, wEnd=\pslinewidth,%
 lineAngle=0,%
 xyAxes=true,%
 xDecimals={},yDecimals={},%
 xLabel={},yLabel={},%
 xlogBase={},ylogBase={},%
 logLines=none,%
 nStep=1,nStart=0,nEnd={},%
 xStep=0,yStep=0,xStart={},xEnd={},yStart={},yEnd={},comma=false,%
%%%%%%%% 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}
}
%
\catcode`\@=\PstAtCode\relax
%
%% END: pstricks-add.tex
\endinput

%% CHANGE-LOG

% v 1.2 added pstplotPolynomial
```

```
% v 1.13 make nStep working for readdata, too
% v 1.12 fixed bug with spurious blank in \endpspicture
% v 1.11 added stuff for data files like x,y1,y2,y3,y4,...
% v 1.10 new numerical macro \pst@mod
% v 1.0h fix a bug in frame setting
% v 1.0g added option frame
% v 1.0f fixed a new introduced bug with xyAxes
% v 1.0e fixed some bugs with xyAxes options
% v 1.0d added more options for \resetOptions
% v 1.0c added \resetPlotOptions
% v 1.0b added options xAxes and yAxes
% v 1.0a added \ncdiagg
% v 1.0 initial version, which collects all the other new macros for
%          pst-plot and pst-node
% v 0.9c add \psFormatInt
% v 0.9b add \pslineIII
% v 0.9a add relative dashNo
% v 0.9 add \pslineII
% v 0.8e add bracePos
% v 0.8d fix bug in arrows
% v 0.8c small tweaks to \@@rput@iv
% v 0.8b now every object can be passed to psbrace
% v 0.8a fix bug with arrow
% v 0.8 ArrowFill added
% v 0.7a adding option asolid as fillstyle
% v 0.7 ArrowInsidePos>1 for all macros
% v 0.6 ArrowInsidePos>1 sets nowthe arrows every n-th pt
% v 0.5a small improvements to the code (use of Pyth)
% v 0.5 new psbezier and pcline to get the arrows in the right place!
% v 0.4 fix bug in psbezier and nccurve, to get the right arrow position
```