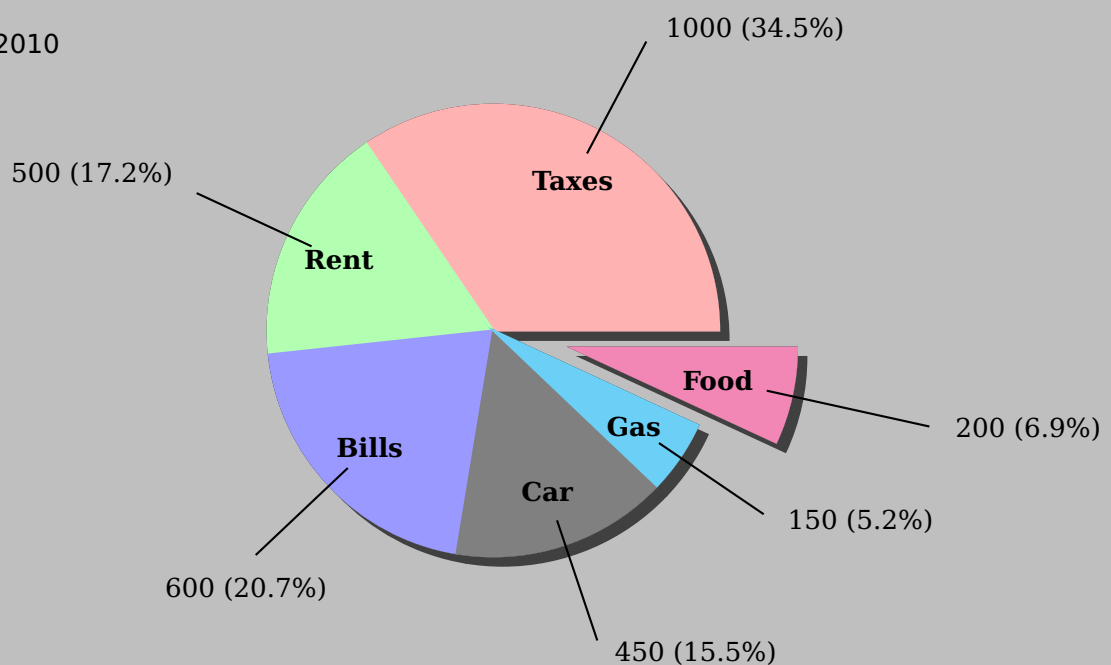


PSTricks

pstricks-add additional Macros for pstricks v.3.39

April 5, 2010



Documentation by
Herbert Voß

Package author(s):
Dominique Rodriguez
Herbert Voß

This version of `pstricks-add` needs `pstricks.tex` version >1.04 from June 2004, otherwise the additional macros may not work as expected. The ellipsis material and the option `asolid` (renamed to `eofill`) are now part of the new `pstricks.tex` package, available at CTAN or at <http://perce.de/LaTeX/>. `pstricks-add` will for ever be an experimental and dynamical package, try it at your own risk.

- It is important to load `pstricks-add` as the **last** PSTricks related package, otherwise a lot of the macros won't work in the expected way.
- `pstricks-add` uses the extended version of the `keyval` package. So be sure that you have installed `pst-xkey` which is part of the `xkeyval`-package, and that all packages that use the old `keyval` interface are loaded **before** the `xkeyval`.^[1]
- the option `tickstyle` from `pst-plot` is no longer supported; use `ticksize` instead.
- the option `xyLabel` is no longer supported; use the option `labelFontSize` instead.
- if `pstricks-add` is loaded together with the package `pst-func` then `InsideArrow` of the `\psbezier` macro doesn't work!

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

Contents

1. <code>\psGetSlope</code> and <code>\psGetDistance</code>	5
2. "Handmade" lines :-)	6
3. <code>\rmultiput</code> : a multiple <code>\rput</code>	7
4. <code>\psrotate</code> : Rotating objects	8
5. <code>\psComment</code> : comments to a graphic	10
6. <code>\psChart</code> : a pie chart	11
7. <code>\psHomothetie</code> : central dilatation	14
8. <code>\psbrace</code>	15
9. Random dots	20
10. <code>\psDice</code>	22
11. <code>\psFormatInt</code>	23
12. <code>\psRelNode</code> and <code>\psDefPSPNodes</code>	24
13. <code>\psRelLine</code>	25
14. <code>\psParallelLine</code>	28
15. <code>\psIntersectionPoint</code>	29
16. <code>psCancel</code> environment	31
17. <code>\psStep</code>	33
18. Tangent lines	36
18.1. <code>\psTangentLine</code> and option <code>Tnormal</code>	36
18.2. <code>\psplotTangent</code> and option <code>Tnormal</code>	37
19. Successive derivatives of a function	42
20. Variable step for plotting a curve	44
20.1. Theory	44
20.2. The cosine	44
20.3. The Napierian Logarithm	45
20.4. Sine of the inverse of x	46
20.5. A really complecated function	47
20.6. A hyperbola	48
20.7. Using <code>\parametricplot</code>	49
21. New math functions and their derivatives	50
21.1. The inverse sine and its derivative	50
21.2. The inverse cosine and its derivative	51
21.3. The inverse tangent and its derivative	52
21.4. Hyperbolic functions	53
22. <code>\psplotDiffEqn</code> – solving differential equations	57
22.1. Variable step for differential equations	58
22.2. Equation of second order	62
22.3. Save final state of a equation	76
23. <code>\psMatrixPlot</code>	77
24. Dashed Lines	83
25. Arrows	85
25.1. Definition	85
25.2. Multiple arrows	85

25.3. hookarrow	86
25.4. hookrightarrow and hookleftarrow	87
25.5. ArrowInside Option	87
25.6. ArrowFill Option	88
25.7. Examples	89
25.8. Special arrows v-V,t-T, and f-F	96
25.9. Special arrow option arrowLW	97
26. Transparent colors	97
27. „Manipulating transparent colors”	97
28. Calculated colors	98
29. Gouraud shading	100
A. \resetOptions	103
B. PostScript	103
C. List of all optional arguments for pstricks-add	104
References	105

1. \psGetSlope and \psGetDistance

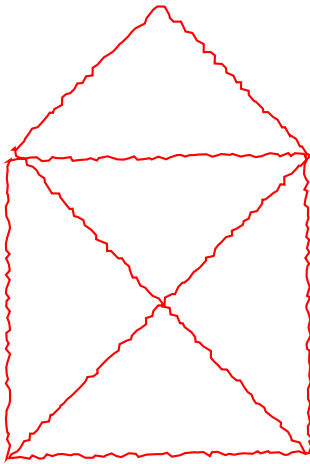
```
\psGetSlope( $x_1, y_1$ )( $x_2, y_2$ )\langle macro \rangle
\psGetDistance( $x_1, y_1$ )( $x_2, y_2$ )\langle macro \rangle
```

```
0.0 5.00146
2.0 2.23656
-0.2 5.09982
0.00615
```

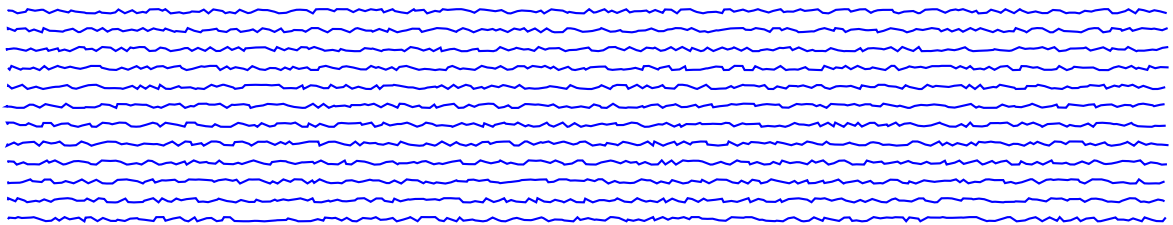
```
1 \psGetSlope(-2,1)(3,1)\SlopeVal \SlopeVal \quad
2 \psGetDistance(-2,1)(3,1)\DVal \DVal\
3 \psGetSlope(-2,1)(-3,-1)\SlopeVal \SlopeVal\quad
4 \psGetDistance(-2,1)(-3,-1)\DVal \DVal\
5 \psGetSlope(-2,0)(3,-1)\SlopeVal \SlopeVal\quad
6 \psGetDistance(-2,0)(3,-1)\DVal \DVal\
7 \psGetSlope(-2111,-12)(3,1)\SlopeVal \SlopeVal\quad
8 %\psGetDistance(-2111,-12)(3,1)\DVal ==> Overflow!
```

2. "Handmade" lines :-)

```
\pslineByHand [Options] (x1,y1)(x2,y2)(x3,y3) ...
```

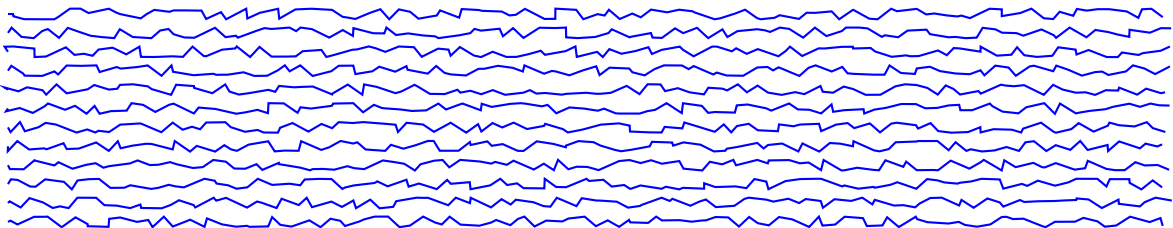


```
1 \begin{pspicture}(4,6)
2 \psset{unit=2cm}
3 \pslineByHand[linecolor=red](0,0)(0,2)(2,2)
   (2,0)(0,0)(2,2)(1,3)(0,2)(2,0)
4 \end{pspicture}
```



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

The amplitude and the width can be changed by the optional arguments `varsteptol` and `VarStepEpsilon`. Both are preset to `VarStepEpsilon=2, varsteptol=0.8`.

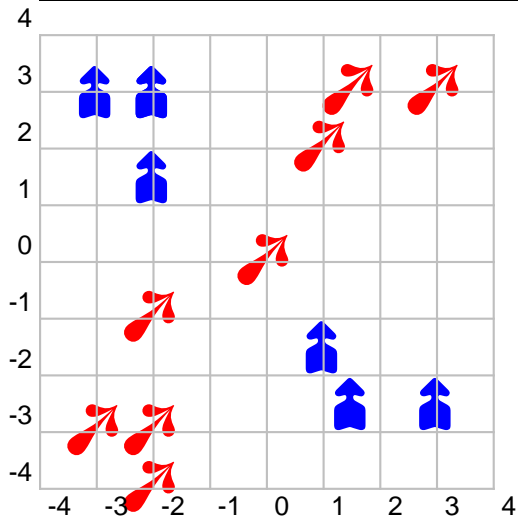


```
1 \begin{pspicture}(\linewidth,3)
2 \multido{\rA=0.00+0.25}{12}{%
3 \pslineByHand[linecolor=blue,VarStepEpsilon=4,varsteptol=2](0,\rA)(\linewidth
   ,\rA)}
4 \end{pspicture}
```

3. \rmultiput: a multiple \rput

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

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

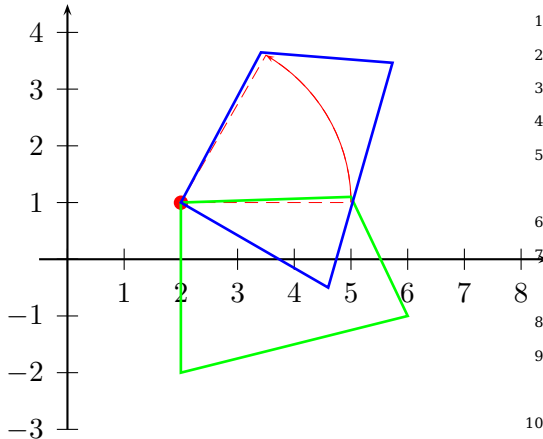


```
1 \psset{unit=0.75}
2 \begin{pspicture}(-4,-4)(4,4)
3 \rmultiput[rot=45]{\red\psscalebox{3}{\ding
4 {250}}}%
5 (-2,-4)(-2,-3)(-3,-3)(-2,-1)(0,0)(1,2)
6 (1.5,3)(3,3)
7 \rmultiput[rot=90,ref=LC]{\blue\psscalebox
8 {2}{\ding{253}}}%
9 (-2,2.5)(-2,2.5)(-3,2.5)(-2,1)(1,-2)
10 (1.5,-3)(3,-3)
11 \psgrid[subgriddiv=0,gridcolor=lightgray]
12 \end{pspicture}
```

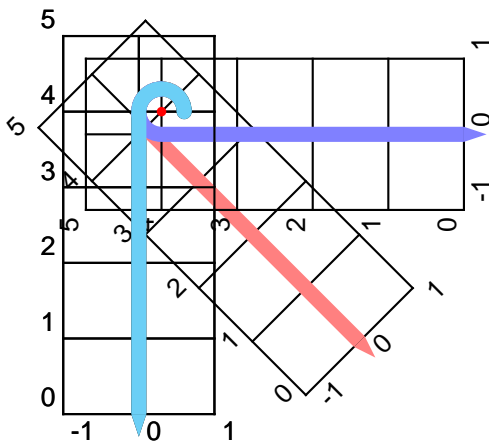
4. \psrotate: Rotating objects

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

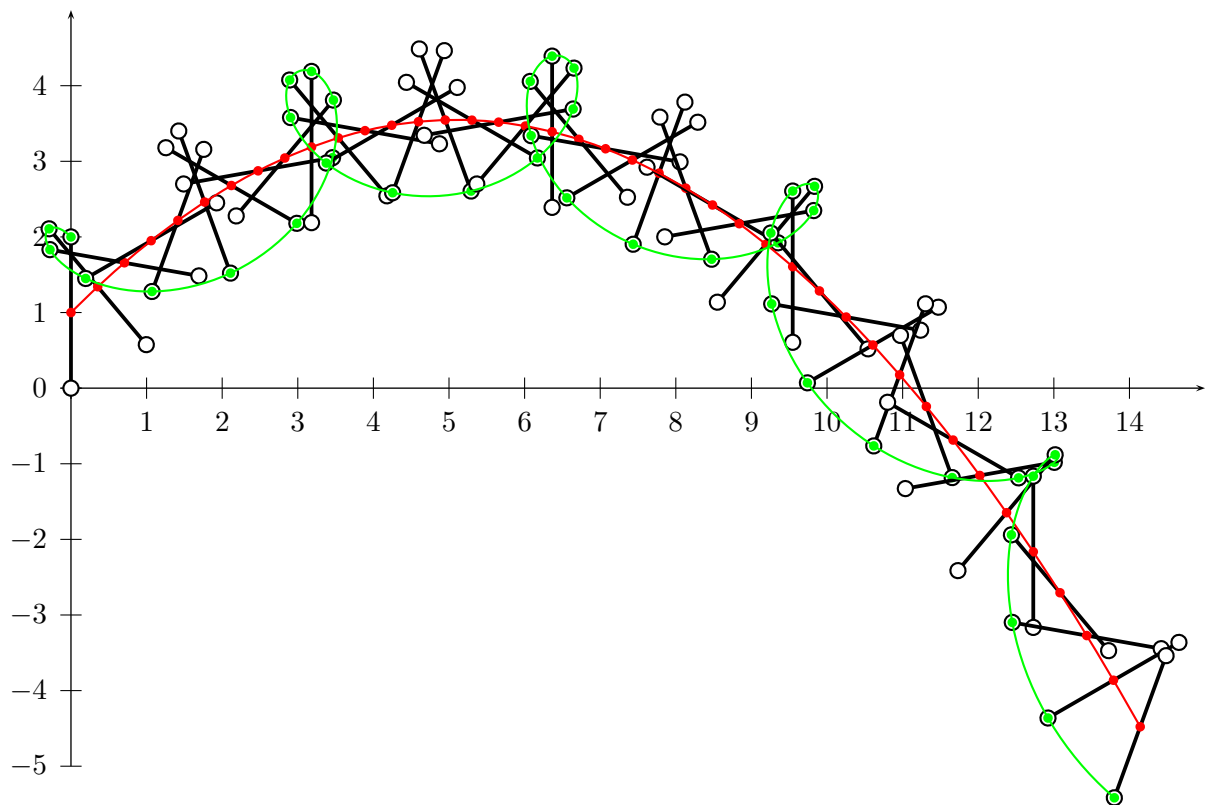
```
\psrotate [Options] (x,y){rot angle}{object}
```



```
1 \psset{unit=0.75}
2 \begin{pspicture}(-0.5,-3.5)(8.5,4.5)
3 \psaxes{->}(0,0)(-0.5,-3)(8.5,4.5)
4 \psdots[linecolor=red,dotscale=1.5](2,1)
5 \psarc[linecolor=red,linewidth=0.4pt,
6 showpoints=true]
7 {->}(2,1){3}{0}{60}
8 \pspolygon[linecolor=green,linewidth=1pt]
9 (2,1)(5,1.1)(6,-1)(2,-2)
10 \psrotate(2,1){60}{%
11 \pspolygon[linecolor=blue,linewidth=1pt]
12 (2,1)(5,1.1)(6,-1)(2,-2)}
13 \end{pspicture}
```



```
1 \begin{pspicture}(-1,-1)(3,6)
2 \def\cane{% Idea by Manuel Luque
3 \psgrid[subgriddiv=0](-1,0)(1,5)
4 \pscustom[linewidth=2mm]{\psline(0,4)\psarcn
5 (0.3,4){0.3}{180}{360}}%
6 \pscircle*(0.6,4){0.1}\pstriangle*(0,0)
7 (0.2,-0.3)}
8 \def\Object{}
9 \cane
10 \psrotate(0.3,4){45}{\psset{linecolor=red
11 !50}\cane}
12 \psrotate(0.3,4){90}{\psset{linecolor=blue
13 !50}\cane}
14 \psrotate(0.3,4){360}{\psset{linecolor=cyan
15 !50}\cane}
16 \psdot[linecolor=red](0.3,4)
17 \end{pspicture}
```

```

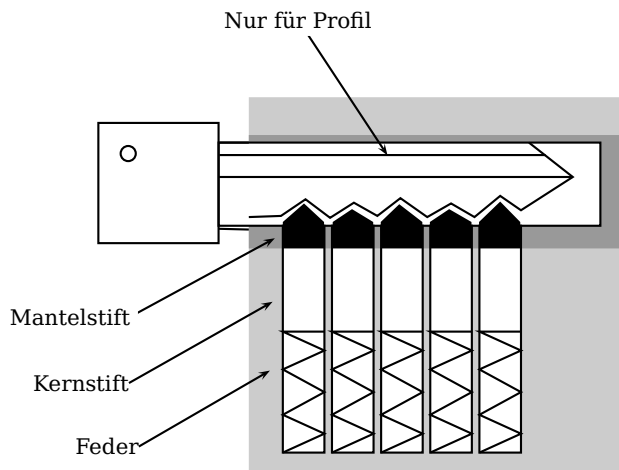
1 \begin{pspicture}(0,-6)(15,5)
2 \def\majorette{\psline[linewidth=0.5mm](0,2)% Idea by Manuel Luque
3     \pscircle[fillstyle=solid]{0.1}
4     \pscircle[fillstyle=solid](0,2){0.1}}
5 \psaxes[linewidth=0.5pt]{->}(0,0)(0,-5)(15,5)
6 \pstVerb{/V0 10 def /Alpha 45 def}% vitesse initiale, angle de lancement
7 \multido{\nT=0.0+0.05,\iA=0+40}{41}{%
8   \pstVerb{/nT \nT\space def}%
9   \rput(!V0 Alpha cos mul nT mul -9.81 2 div nT dup mul mul V0 Alpha sin mul
10    nT mul add){%
11     \psrotate(0,1){\iA}{\majorette\psdot[linecolor=red](0,1)\psdot[linecolor=
12     green](0,2)}}}
13 \parametricplot[linecolor=red]{0}{2}{% trajectoire du milieu
14   V0 Alpha cos mul t mul -9.81 2 div t dup mul mul V0 Alpha sin mul t mul add
15   1 add}
16 \parametricplot[linecolor=green,plotpoints=360]{0}{2}{% d'une extremite
17   V0 Alpha cos mul t mul 800 t mul sin sub % x(t)
18   -9.81 2 div t dup mul mul V0 Alpha sin mul t mul add 1 add 800 t mul cos
19   add }%y(t)
20 \end{pspicture}

```

5. \psComment: comments to a graphic

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

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



```

1 \SpecialCoor\newsstyle{weiss}{fillstyle=solid,fillcolor=white}
2 \footnotesize\psset{unit=0.5cm,dimen=middle}
3 \begin{pspicture}(-12,-4)(6,10)
4 \psframe*[linecolor=black!20](-5,-3)(5,7) \psframe*[linecolor=black!40](-5,3)(5,6)
5 \pscicle(-8.19,5.51){0.2}
6 \psframe[fillcolor=white,fillstyle=solid](-5.8,3.6)(4.3,5.8)
7 \psframe(-8.98,3.14)(-5.8,6.32)
8 \multido{\rA=-4.1+1.3}{5}{\rput(\rA,-2.4){\psframe[style=weiss](1.1,6)
9 \psline(0,0)(1.1,0.5)(0,1)(1.1,1.6)(0,2.2)(1.1,2.7)(0,3.2)(1.1,3.2)}}
10 \pspolygon*(-4.1,3.7)(-4.1,3)(-3,3)(-3.01,3.7)(-3.54,4.19)
11 \pspolygon*(1.09,3.7)(1.1,3)(2.2,3)(2.18,3.7)(1.65,4.24)
12 \pspolygon*(-2.78,3.7)(-2.8,3)(-1.7,3)(-1.71,3.7)(-2.27,4.04)
13 \pspolygon*(-1.51,3.7)(-1.5,3)(-0.4,3)(-0.41,3.7)(-1.02,4.17)
14 \pspolygon*(-0.21,3.7)(-0.2,3)(0.9,3)(0.89,3.7)(0.3,4.04)
15 \psline(-5,3.83)(-4.15,3.86)(-3.5,4.3)(-2.85,3.81)(-2.22,4.21)(-1.6,3.86)(-0.99,4.33)
16 (-0.28,3.83)(0.35,4.19)(0.97,3.83)(1.65,4.39)(2.2,4.01)(3.57,4.89)(2.41,5.8)
17 \psline(-5,5.8)(-5.78,5.8) \psline(-5.78,5.47)(2.85,5.47)
18 \psline(-5.8,3.52)(-5,3.5) \psline(3.57,4.89)(-5.8,4.89)
19 \psComment*[ref=r]{->}(-8.14,1.19)(-4.31,3.27){Mantelstift}
20 \psComment*[ref=r]{->}(-8.17,-0.56)(-4.37,1.59){Kernstift}[\rput]
21 \psComment*[ref=r]{->}(-7.91,-2.24)(-4.44,-0.23){Feder}[\rput]
22 \psComment[npos=-0.1]{->}(-3.48,8.72)(-1.33,5.46){Nur für Profil}
23 \end{pspicture}

```

6. \psChart: a pie chart

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

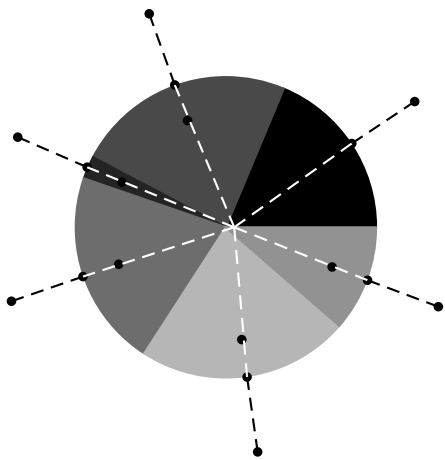
The special optional arguments for the \psChart macro are as follows:

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

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

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

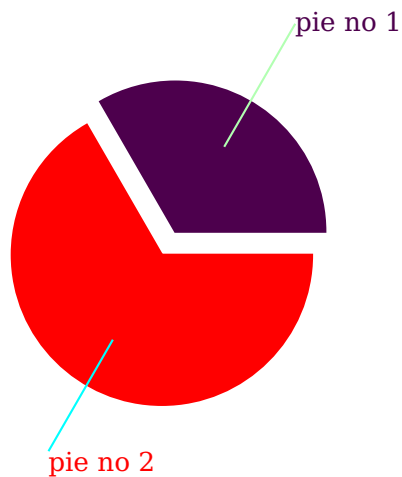
The other one is the node on the circle line. The origin is by default (0,0). Moving the pie to another position can be done as usual with the \rput-macro. The used colors are named internally as chartFillColor? and can be used by the user for coloring lines or text.



```

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

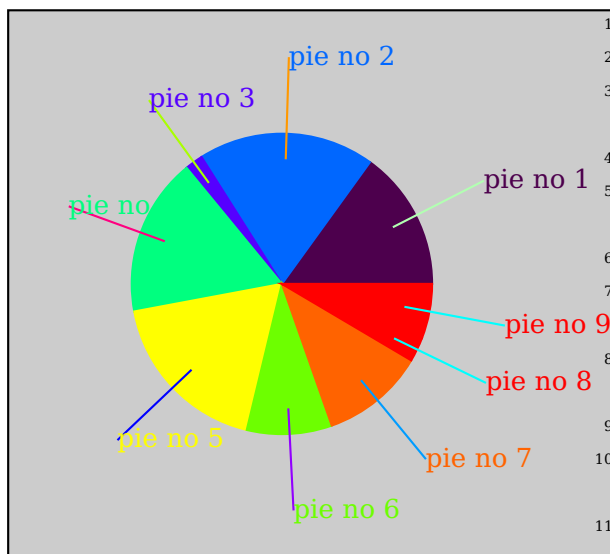
```



```

1 \begin{pspicture}(-3,-3)(3,3)
2 \psChart[chartColor=color]{45,90}{1}{2}
3 \ncline[linecolor=-chartFillColor1,
4   nodesepB=-20pt]{psChart01}{psChart1}
5 \rput[l](psChart01){%
6   \textcolor{chartFillColor1}{pie no 1}}
7 \ncline[linecolor=-chartFillColor2,
8   nodesepB=-20pt]{psChart02}{psChart2}
9 \rput[lt](psChart02){%
10  \textcolor{chartFillColor2}{pie no 2}}
11 \end{pspicture}

```



```

1 \psframebox[fillcolor=black!20,
2   fillstyle=solid]{%
3 \begin{pspicture}(-3.5,-3.5)
4   (4.25,3.5)
5 \psChart[chartColor=color]%
6   {23, 29, 3, 26, 28, 14, 17, 4,
7     9}{2}
8 \multido{\iA=1+1}{9}{%
9   \ncline[linecolor=-chartFillColor\iA,
10  nodesepB=-10pt]{psChart0\iA}{
11  psChart\iA}
12 \rput[l](psChart0\iA){%
13   \textcolor{chartFillColor\iA}{pie
14   no \iA}}
15 \end{pspicture}

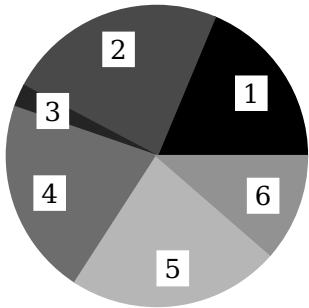
```



```

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

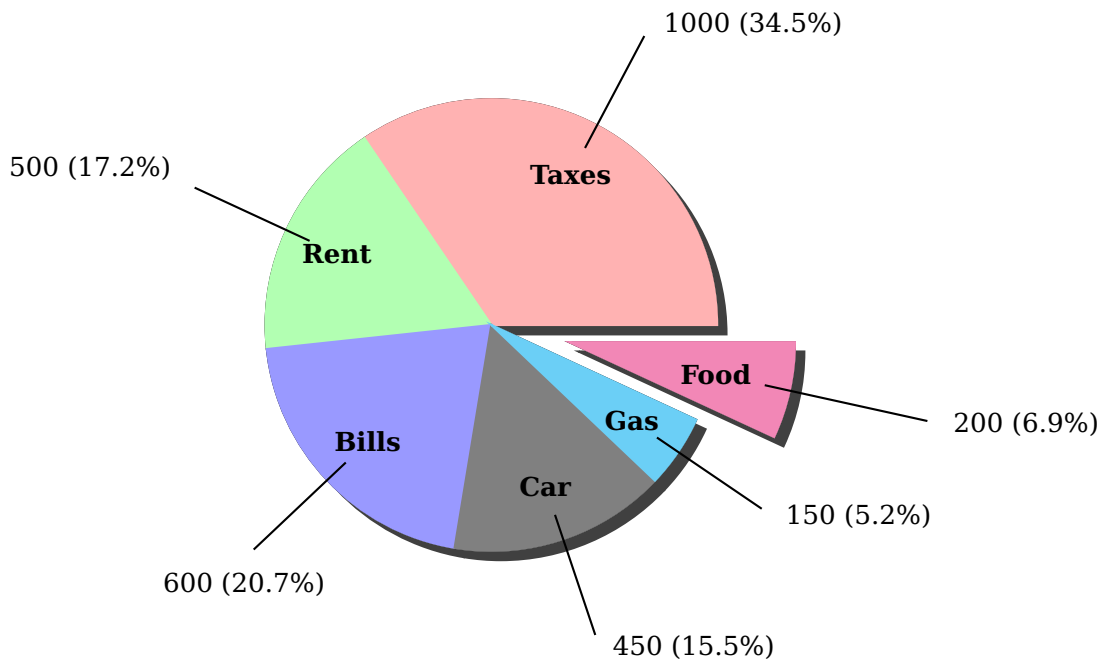
```



```

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

```



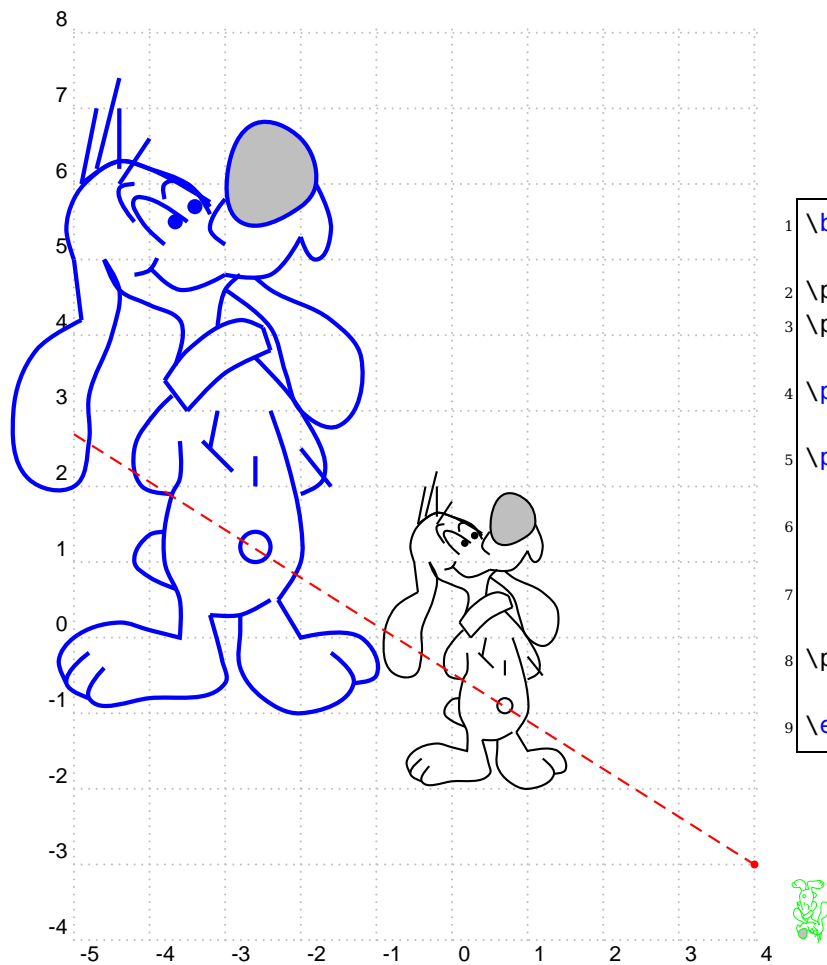
```

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

```

7. \psHomothetie: central dilatation

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

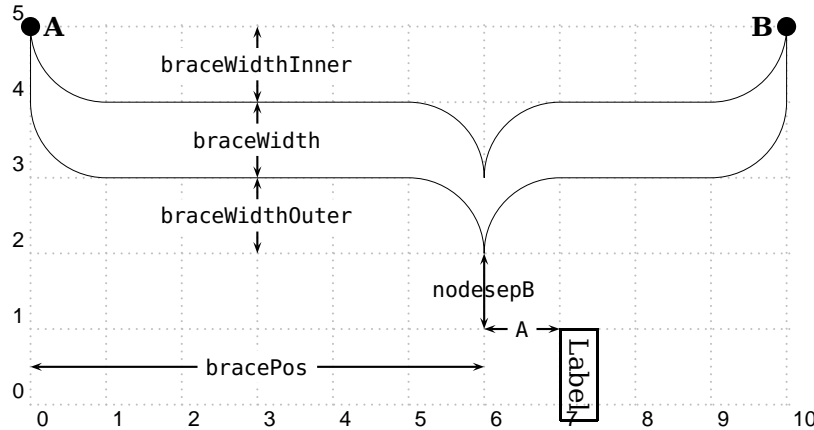


```
1 \begin{pspicture}[showgrid=
  true](-5,-4)(4,8)
2 \psBill% needs package pst-fun
3 \psHomothetie[linecolor=blue
  ](4,-3){2}{\psBill}
4 \psdots[dotsize=3pt,linecolor=
  red](4,-3)
5 \psplot[linestyle=dashed,
  linecolor=red]{-5}{4}%
6 [ /m -3 -0.85 sub 4 0.6 sub
  div def ]
7 { m x mul m 4 mul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

8. \psbrace

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

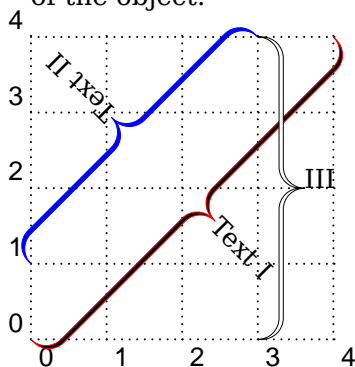
Additional to all other available options from pstricks or the other related packages, there are two new option, named braceWidth and bracePos. All important ones are shown in the following graphics and table.



A positive value for nodeSepA and nodeSepB shifts the label to the upper right and a negative value to the lower left. This does not depend on the value for the rotating of the label!

name	meaning
braceWidth	default is \pslinewidth
braceWidthInner	default is 10\pslinewidth
braceWidthOuter	default is 10\pslinewidth
bracePos	relative position (default is 0.5)
nodeSepA	x-separation (default is 0pt)
nodeSepB	y-separation (default is 0pt)
rot	additional rotating for the text (default is 0)
ref	reference point for the text (default is c)
fillcolor	default is black

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



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

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

```

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

```

```

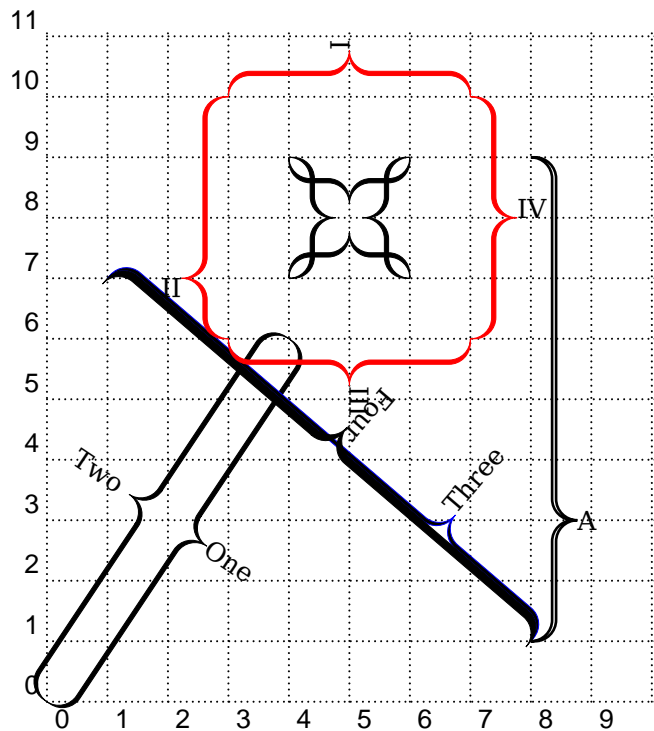
1 \def\someMath{\int\limits_1^{\infty}\frac{1}{x^2}\,dx=1$}
2 \begin{pspicture}(8,2.5)
3 \psbrace[ref=lC](0,0)(0,2){\someMath}%
4 \psbrace[rot=90](2,0)(2,2){\someMath}
5 \psbrace[ref=lC](4,0)(4,2){\someMath}
6 \psbrace[ref=lt,rot=90,nodesepB=-30pt](6,0)(6,2){\someMath}
7 \psbrace[ref=lt,rot=90,nodesepB=30pt](8,2)(8,0){\someMath}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

```

```

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

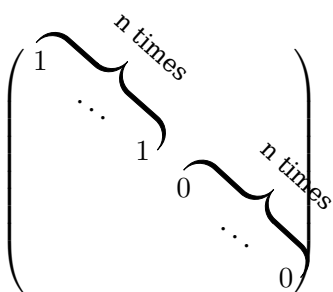
```

```

1 \psset{unit=0.8}
2 \begin{pspicture}(10,11)
3 \psgrid[subgriddiv=0,griddots=10]
4 \pnode(0,0){A}
5 \pnode(4,6){B}
6 \psbrace[ref=lc](A)(B){One}
7 \psbrace[rot=180,nodesepA=-5pt,ref=
  rb](B)(A){Two}
8 \psbrace[linecolor=blue,bracePos
  =0.25,ref=lb](8,1)(1,7){Three}
9 \psbrace[braceWidth=-1mm,rot=180,
  ref=rB](8,1)(1,7){Four}
10 \psbrace*[linearc=0.5,fillstyle=
  none,linewidth=1pt,braceWidth
  =1.5pt,
  bracePos=0.25,ref=lc](8,1)(8,9){A
  }
12 \psbrace(4,9)(6,9){}
13 \psbrace(6,9)(6,7){}
14 \psbrace(6,7)(4,7){}
15 \psbrace(4,7)(4,9){}
16 \psset{linecolor=red}
17 \psbrace*[ref=lb](7,10)(3,10){I}
18 \psbrace*[ref=lb,bracePos
  =0.75](3,10)(3,6){II}
19 \psbrace*[ref=lb](3,6)(7,6){III}
20 \psbrace*[ref=lb](7,6)(7,10){IV}
21 \end{pspicture}

```



```

1 \[
2 \begin{pmatrix}
3 \Rnode[vref=2ex]{A}{-1} \ \
4 & \ddots \ \
5 & & \Rnode[href=2]{B}{1} \ \
6 & & & \Rnode[vref=2ex]{C}{0} \ \
7 & & & & \ddots \ \
8 & & & & & \Rnode[href=2]{D}{0} \sim \ \
9 \end{pmatrix}
10 \]
11 \psbrace[rot=-90,nodesepB=-0.5,nodesepA=-0.2](B)(A)
  {\small n times}
12 \psbrace[rot=-90,nodesepB=-0.5,nodesepA=-0.2](D)(C)
  {\small n times}

```

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

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

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

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

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

```

1 Some nonsense text, which is nothing more than nonsense.
2 Some nonsense text, which is nothing more than nonsense.
3
4 \noindent\rnode{A}{
5
6 \vspace*{-1ex}
7 Some nonsense text, which is nothing more than nonsense.
8 Some nonsense text, which is nothing more than nonsense.
9 Some nonsense text, which is nothing more than nonsense.
10 Some nonsense text, which is nothing more than nonsense.
11 Some nonsense text, which is nothing more than nonsense.
12 Some nonsense text, which is nothing more than nonsense.
13 Some nonsense text, which is nothing more than nonsense.
14 Some nonsense text, which is nothing more than nonsense.
15
16 \vspace*{-2ex}\noindent\rnode{B}{}\psbrace[linecolor=red](A)(B){
17
18 Some nonsense text, which is nothing more than nonsense.
19 Some nonsense text, which is nothing more than nonsense.
20
21 \medskip\hfill\begin{minipage}{0.95\linewidth}
22 \noindent\rnode{A}{
23
24 \vspace*{-1ex}
25 Some nonsense text, which is nothing more than nonsense.
26 Some nonsense text, which is nothing more than nonsense.
27 Some nonsense text, which is nothing more than nonsense.
28 Some nonsense text, which is nothing more than nonsense.
29 Some nonsense text, which is nothing more than nonsense.
30 Some nonsense text, which is nothing more than nonsense.
31 Some nonsense text, which is nothing more than nonsense.
```

```
32 Some nonsense text, which is nothing more than nonsense.  
33  
34 \vspace*{-2ex}\noindent\rnode{B}{}\psbrace[linecolor=red](A)(B){}  
35 \end{minipage}
```

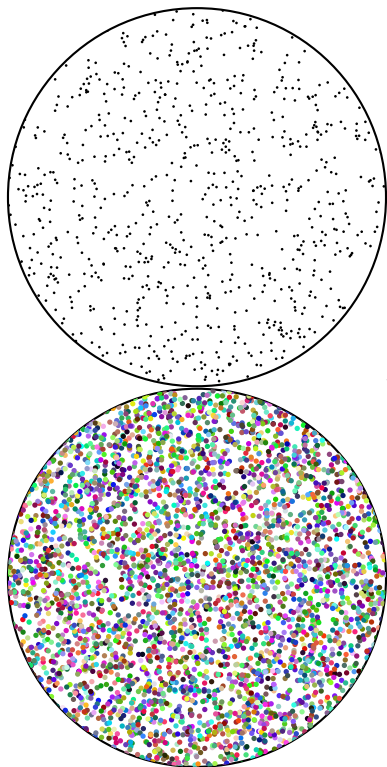
9. Random dots

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

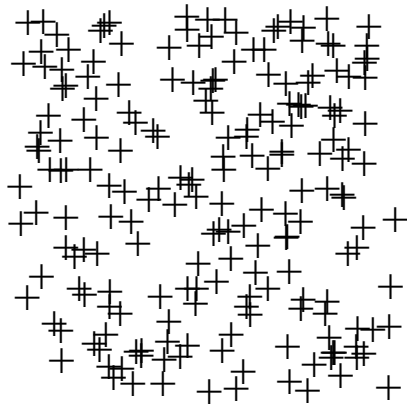
```
\psRandom [Options] {}
\psRandom [Options] (xMin,yMin) (xMax,yMax) {clip path}
```

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

name	default	
<code>randomPoints</code>	1000	number of random dots
<code>color</code>	false	random color



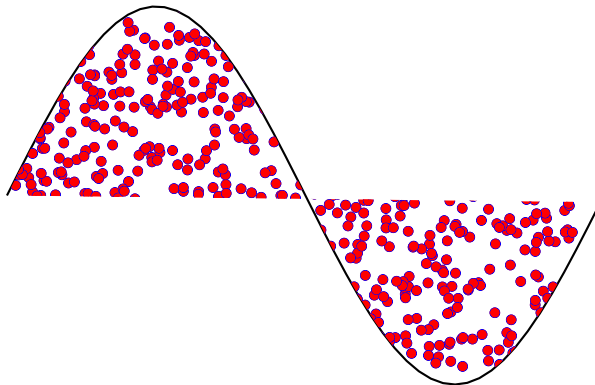
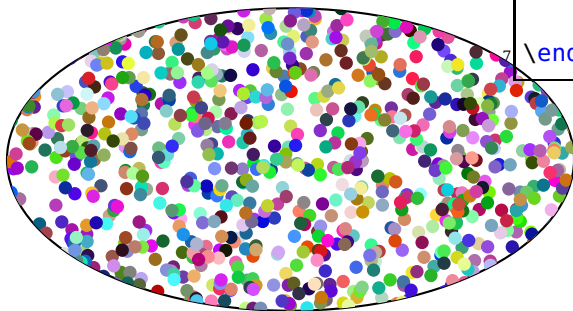
```
1 \psset{unit=5cm}
2 \begin{pspicture}(1,1)
3   \psRandom[dotsize=1pt,fillstyle=solid](1,1){\
4     \pscircle(0.5,0.5){0.5}}
5 \end{pspicture}
6 \begin{pspicture}(1,1)
7   \psRandom[dotsize=2pt,randomPoints=5000,color,%
8     fillstyle=solid](1,1){\pscircle(0.5,0.5){0.5}}
9 \end{pspicture}
```



```

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

```



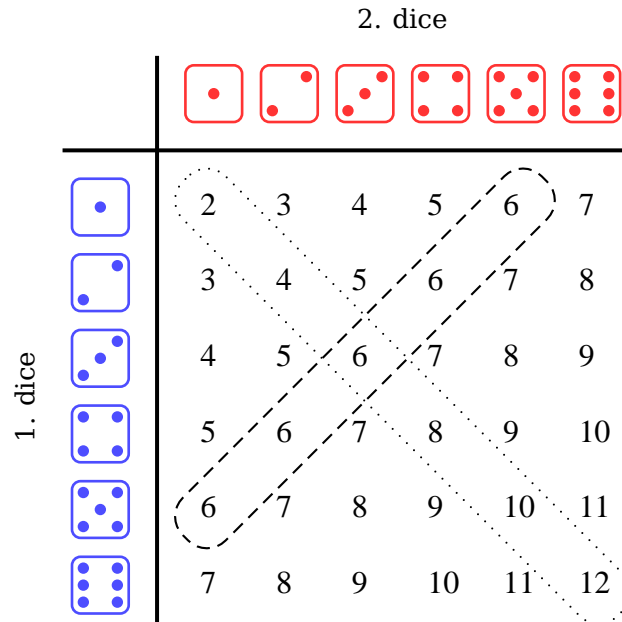
```

1 \psset{unit=2.5cm}
2 \begin{pspicture}(0,-1)(3,1)
3   \psRandom[dotsize=4pt, dotstyle=o,
4     linecolor=blue, fillcolor=red, %
5     fillstyle=solid, randomPoints
6     =1000] %
7     (0,-1)(3,1) { \psplot{0}{3.14}{ x
8       114 mul sin } }
9 \end{pspicture}

```

10. \psDice

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



```

1 \begin{pspicture}(-1,-1)(8,8)
2 \multido{\iA=1+1}{6}{%
3   \rput(\iA,7.5){\Huge\psdice[unit=0.75,linecolor=red!80]{\iA}}
4   \rput(! -0.5 7 \iA\space sub){\Huge\psdice[unit=0.75,linecolor=blue!70]{\iA}}
5   %
6   \multido{\iB=1+1}{6}{%
7     \rput(! \iA\space 7 \iB\space sub){%
8       \rnode[c]{p\iA\iB}{\makebox[1em][l]{\strut\psPrintValue[fontscale=12]{\iA\
9         space \iB\space add}}}%
10    }%
11  }%
12 }%
13 }%
14 }%
15 \end{pspicture}

```

11. \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 \psFormatInt{1234567}\\
1,234,567	2 \psFormatInt[intSeparator={,}]{1234567}\\
1.234.567	3 \psFormatInt[intSeparator=.]{1234567}\\
1.234.567	4 \psFormatInt[intSeparator=\$\cdot\$]{1234567}\\
1·234·567	5 \def\temp{965432}
965,432	6 \psFormatInt{\temp}

With the option `intSeparator` the symbol can be changed to any non-number character.

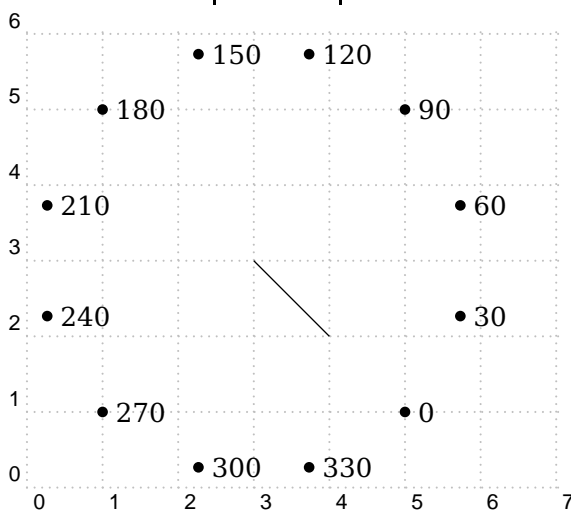
12. \psRelNode and \psDefPSPNodes

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

```
\psRelNode[P0][P1]{length factor}{end node name}
\psDefPSPNodes
```

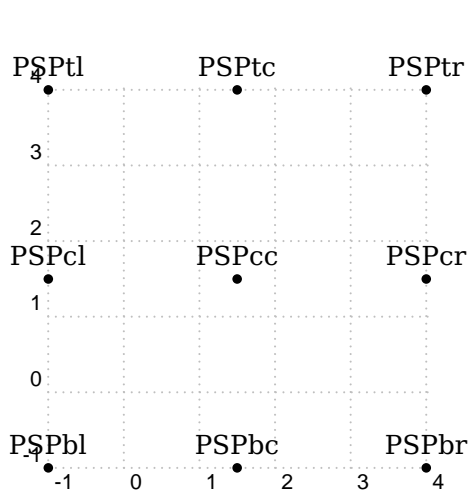
The length factor relates to the distance $\overline{P_0P_1}$ and the end node name must be a valid nodename and shouldn't contain any of the special PostScript characters. There are two valid options:

name	default	meaning
angle	0	angle between the given line $\overline{P_0P_1}$ and the new one $\overline{P_0P_{endNode}}$
trueAngle	false	defines whether the angle refers to the seen line or to the mathematical one, which respect the scaling factors xunit and yunit.



```
1 \begin{pspicture}[showgrid](7,6)
2   \pnode(3,3){A}\pnode(4,2){B}
3   \psline[nodesep=-3,linewidth=0.5pt](A)(
4     B)
5   \multido{\iA=0+30}{12}{%
6     \psRelNode[angle=\iA](A)(B){2}{C}%
7     \qdisk(C){2pt}
8     \uput[0](C){\iA}}
9 \end{pspicture}
```

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



```
1 \begin{pspicture}[showgrid=true](-1,-1)(4,4)
2   \psDefPSPNodes
3   \psdots(PSPbl)(PSPbc)(PSPbr)
4     (PSPcl)(PSPcc)(PSPcr)(PSPtl)(PSPtc)(PSPtr)
5   \uput[90](PSPbl){PSPbl} \uput[90](PSPbc){
6     PSPbc}
7   \uput[90](PSPbr){PSPbr} \uput[90](PSPcl){
8     PSPcl}
9   \uput[90](PSPcc){PSPcc} \uput[90](PSPcr){
10    PSPcr}
11 \uput[90](PSPtl){PSPtl} \uput[90](PSPtc){
12    PSPtc}
13 \uput[90](PSPtr){PSPtr}
14 \end{pspicture}
```

The name of the nodes are predefined as:


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

and can be modified in the same way.

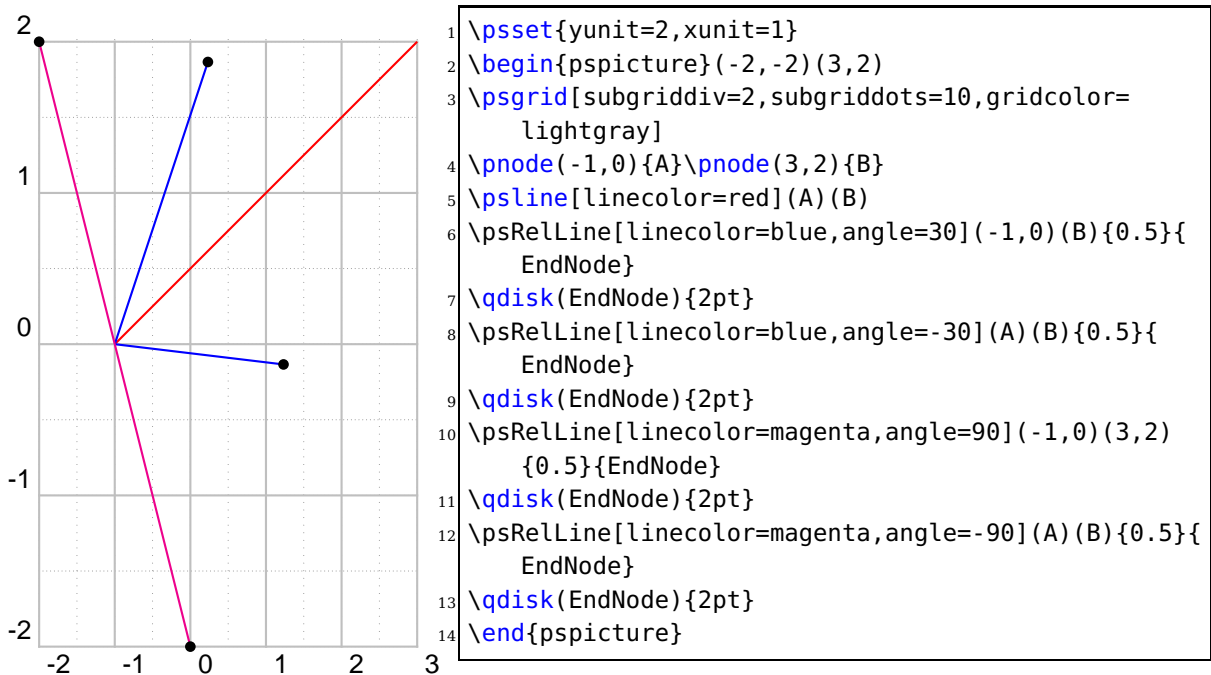
13. \psRelLine

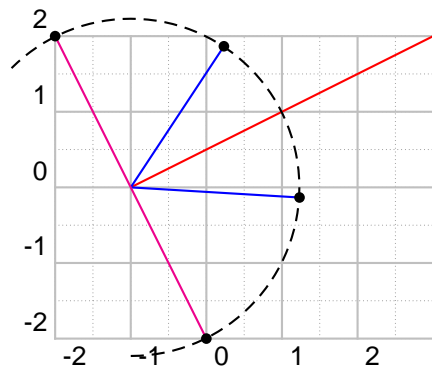
With this macro it is possible to plot lines relative to a given one. Parameter are the angle and the length factor:

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

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

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



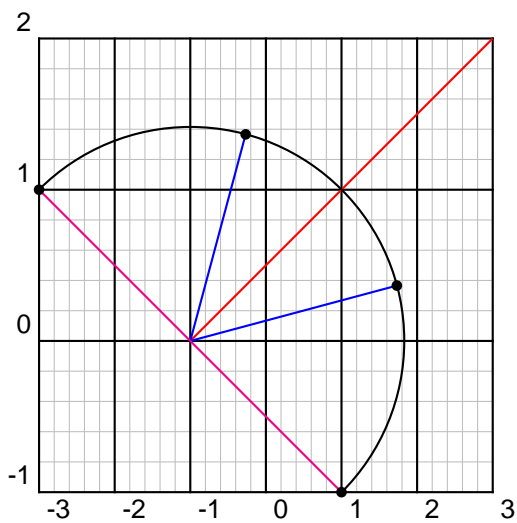


```

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

```

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

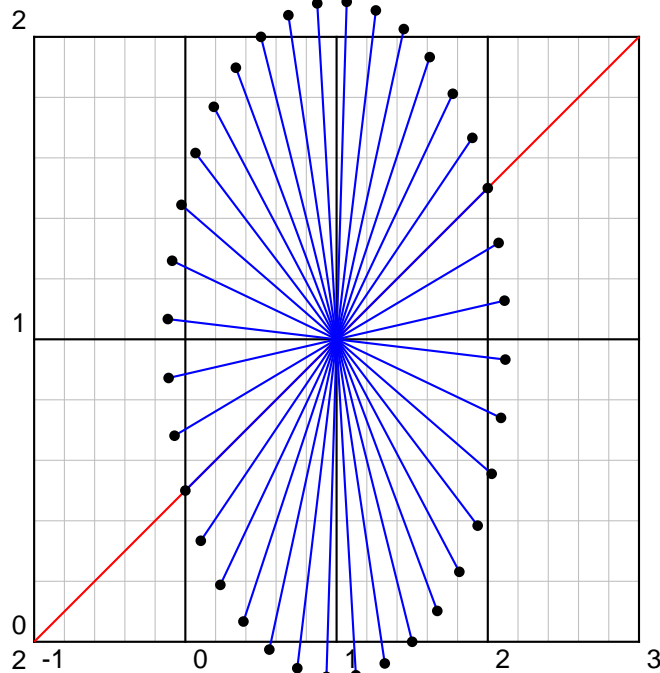


```

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

```

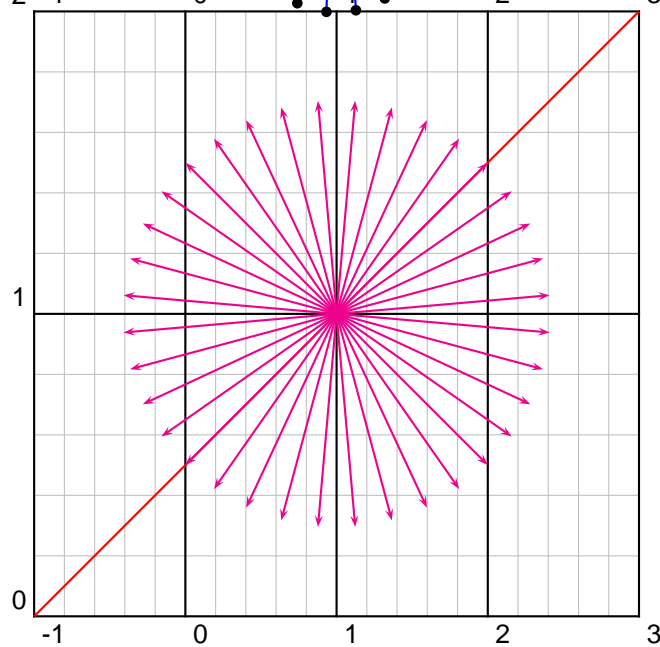
Two examples using `\multido` to show the behaviour of the options `trueAngle` and `angle`.



```

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

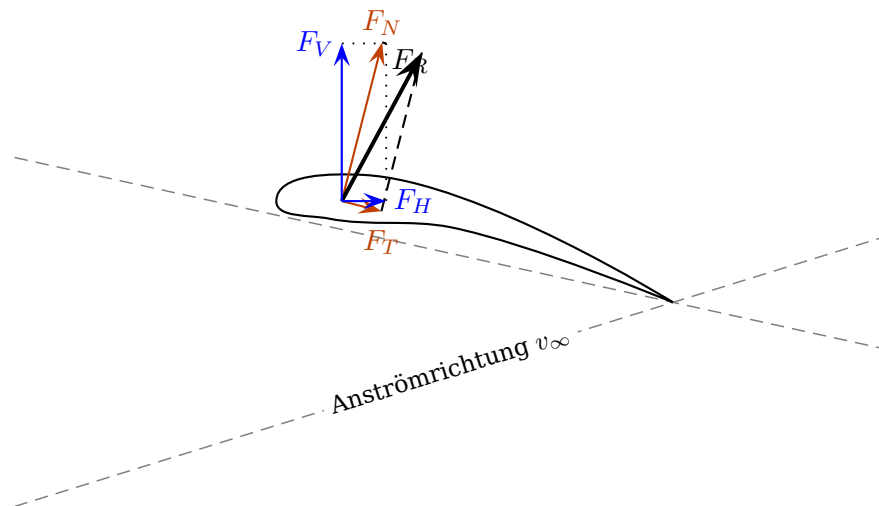
```



```

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

```



```

1 \psset{xunit=0.75\linewidth,yunit=0.75\linewidth,trueAngle}%
2 \end{center}
3 \begin{pspicture}(1,0.6)\psgrid
4 \pnode(.3,.35){Vk} \pnode(.375,.35){D} \pnode(0,.4){DST1} \pnode(1,.18){DST2}
5 \pnode(0,.1){A1} \pnode(1,.31){A1}
6 { \psset{linewidth=.02,linestyle=dashed,linicolor=gray}%
7 \pcline(DST1)(DST2) % <- Druckseitentangente
8 \pcline(A2)(A1) % <- Anstr"omrichtung
9 \lput*{:U}{\small Anstr"omrichtung $v_{\infty}$ }%
10 \psIntersectionPoint(A1)(A2)(DST1)(DST2){Hk}
11 \pscurve(Hk)(.4,.38)(Vk)(.36,.33)(.5,.32)(Hk)
12 \psParallelline[linicolor=red!75!green,arrows=->,arrowscale=2](Vk)(Hk)(D)
13 {4}{Fn}% why "4"?
14 \psParallelline[linestyle=dashed](D)(FtE)(Fn){.1}{Fnr1}
15 \psRelLine[linestyle=dashed,angle=90](FtE)(D){-4}{Fnr2} % why "-4"?
16 \psline[linewidth=1.5pt,arrows=->,arrowscale=2](D)(Fnr2)
17 \psIntersectionPoint(D)([nodesep=2]D)(Fnr1)([offset=-4]Fnr1){Fh}
18 \psIntersectionPoint(D)([offset=2]D)(Fnr1)([nodesep=4]Fnr1){Fv}
19 \psline[linicolor=blue,arrows=->,arrowscale=2](D)(Fh)
20 \psline[linicolor=blue,arrows=->,arrowscale=2](D)(Fv)
21 \psline[linestyle=dotted](Fh)(Fnr1) \psline[linestyle=dotted](Fv)(Fnr1)
22 \uput{.1}[0](Fh){\blue $F_{H}$} \uput{.1}[180](Fv){\blue $F_{V}$}
23 \uput{.1}[-45](Fnr1){$F_{R}$} \uput{.1}[90](Fn){\color{red!75!green}$F_{N}$}
24 \uput{.25}[-90](FtE){\color{red!75!green}$F_{T}$}
25 \end{pspicture}

```

14. \psParallelline

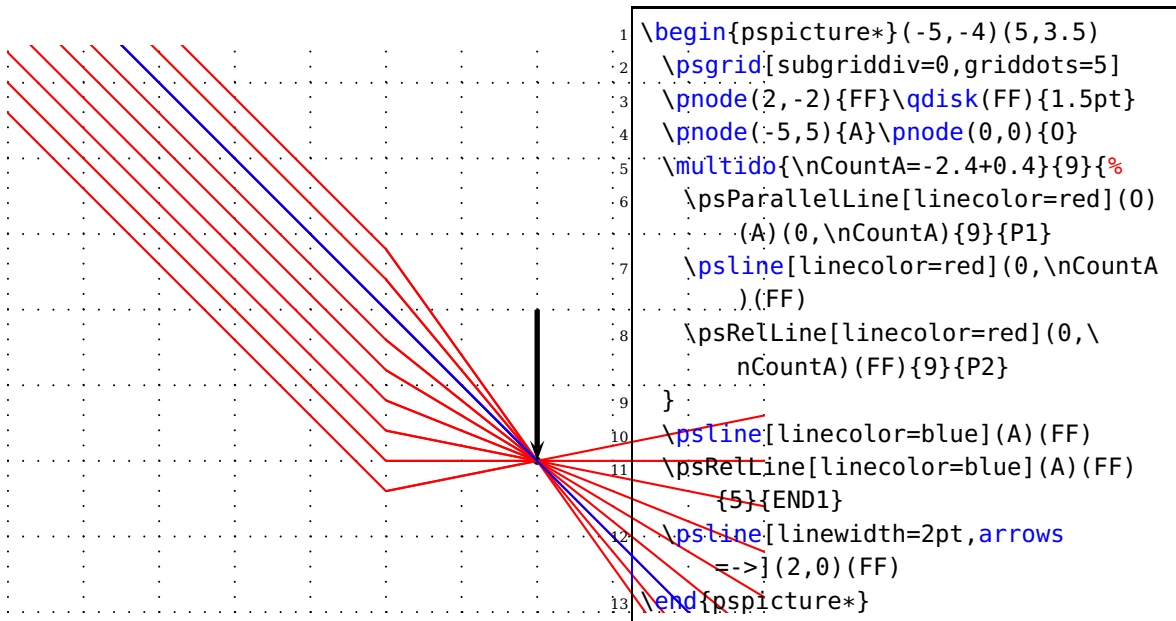
With this macro it is possible to plot lines relative to a given one, which is parallel. There is no special parameter here.

```

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

```

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



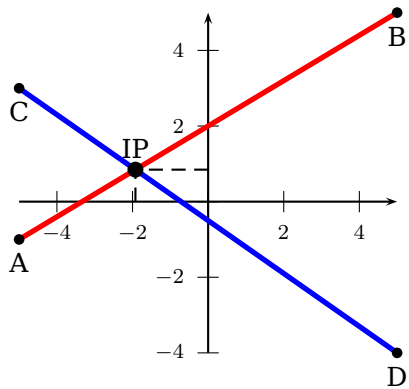
15. `\psIntersectionPoint`

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

```

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

```



```

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

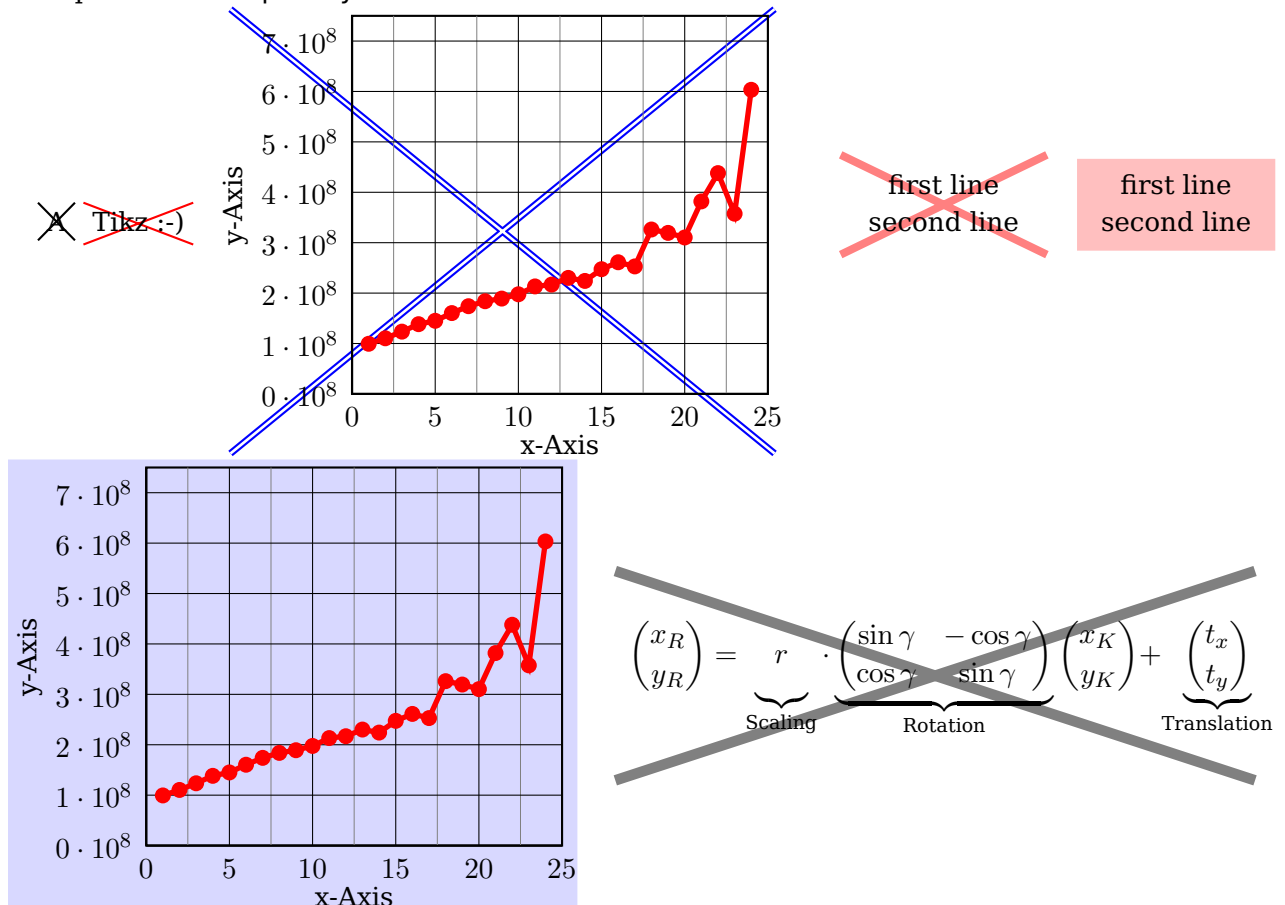
```

16. psCancel environment¹

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

```
\psCancel * [Options] {contents}
```

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



```

1 \psCancel{A} \psCancel[linecolor=red]{Tikz :-)} \quad
2 \psCancel[linecolor=blue,doubleline=true]{%
3 \readdata{\data}{demo1.data}
4 \psset{shift=*,xAxisLabel=x-Axis,yAxisLabel=y-Axis,llx=-13mm,lly=-7mm,
5   xAxisLabelPos={c,-1},yAxisLabelPos={-7,c}}
6 \pstScalePoints(1,0.00000001){}{}
7 \begin{psgraph}[axesstyle=frame,xticks=0 7.5,yticks=0 25,subticks=1,
8   ylabelFactor=\cdot 10^8,Dx=5,Dy=1,xsubticks=2](0,0)(25,7.5){5.5cm}{5cm}

```

¹ Thanks to by Stefano Baroni

```

9 \listplot[linecolor=red, linewidth=2pt, showpoints=true]{\data}
10 \end{psgraph}} \quad% end of Cancel
11 \psCancel[linewidth=3pt, linecolor=red,
12 strokeopacity=0.5]{\tabular[b]{c}first line\\second line\endtabular}\quad
13 \psCancel*[linecolor=red!50, opacity=0.5]{\tabular[b]{c}first line\\second line\
endtabular}
14 \quad
15 \psCancel*[linecolor=blue!30, opacity=0.5]{%
16 \readdata{\data}{demo1.data}
17 \psset{shift=*, xAxisLabel=x-Axis, yAxisLabel=y-Axis, llx=-15mm, lly=-7mm, urx=1mm
,
18 xAxisLabelPos={c, -1}, yAxisLabelPos={-7, c}}
19 \pstScalePoints(1, 0.00000001){}{}
20 \begin{psgraph}[axesstyle=frame, xticksize=0 7.5, yticksize=0 25, subticksize=1,
21 ylabelFactor=\cdot 10^8, Dx=5, Dy=1, xsubticks=2](0,0)(25,7.5){5.5cm}{5cm}
22 \listplot[linecolor=red, linewidth=2pt, showpoints=true]{\data}
23 \end{psgraph}} \quad% end of Cancel
24 \psCancel[linewidth=4pt, strokeopacity=0.5]{\parbox{8cm}{\l[
25 \binom{x_R}{y_R} = \underbrace{r\phantom{\binom{A}{B}}}_{\text{Scaling}}\
\cdot
26 \underbrace{\begin{pmatrix}
27 \sin\gamma & -\cos\gamma \\
28 \cos\gamma & \sin\gamma
29 \end{pmatrix}}_{\text{Rotation}} \binom{x_K}{y_K} +
30 \underbrace{\binom{t_x}{t_y}}_{\text{Translation}} \ \]} }% end of psCancel

```

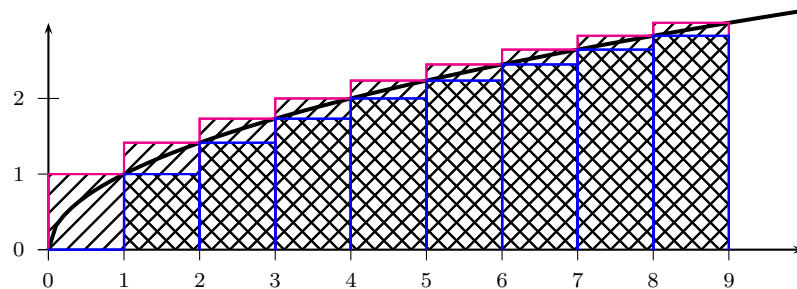

17. \psStep

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

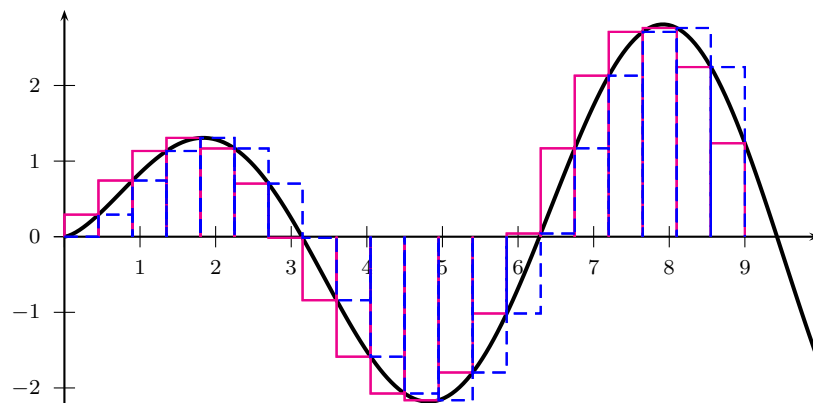
StepType=lower |upper|Riemann|infimum|supremum or alternative StepType=l |u|R|i|s with lower as the default setting. The syntax of the function is

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

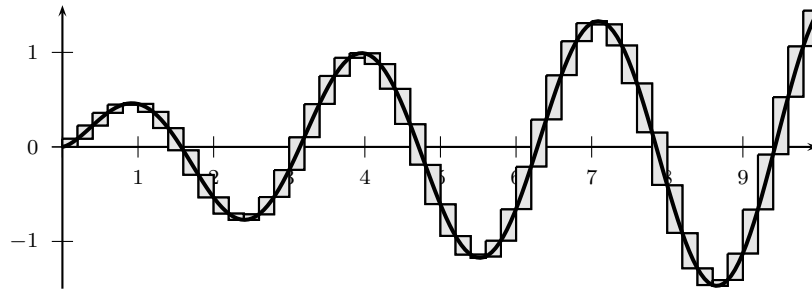
(x1,x2) is the given interval for the step wise calculated function, n is the number of the rectangles and *function* is the mathematical function in postfix or algebraic=true notation (with algebraic=true).



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



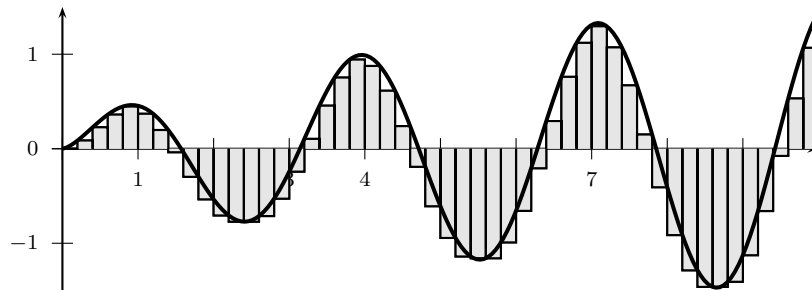
```
1 \psset{plotpoints=200}
2 \begin{pspicture}(-0.5,-2.25)(10,3)
3 \psaxes[labelFontSize=\scriptstyle]{->}(0,0)(0,-2.25)(10,3)
4 \psplot[linewidth=1.5pt,algebraic=true]{0}{10}{sqrt(x)*sin(x)}
5 \psStep[algebraic=true,linecolor=magenta,StepType=upper](0,9){20}{sqrt(x)*sin(x)}
6 \psStep[linecolor=blue,linestyle=dashed](0,9){20}{x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



```

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

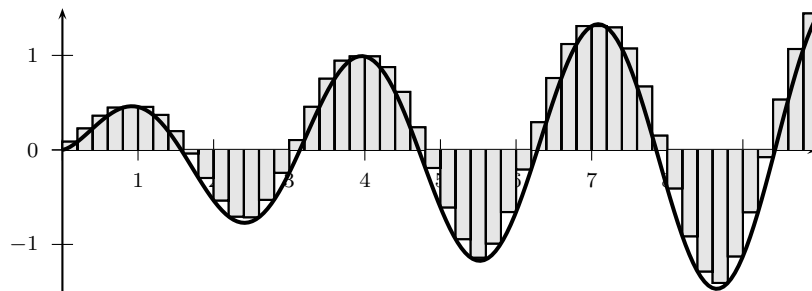
```



```

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

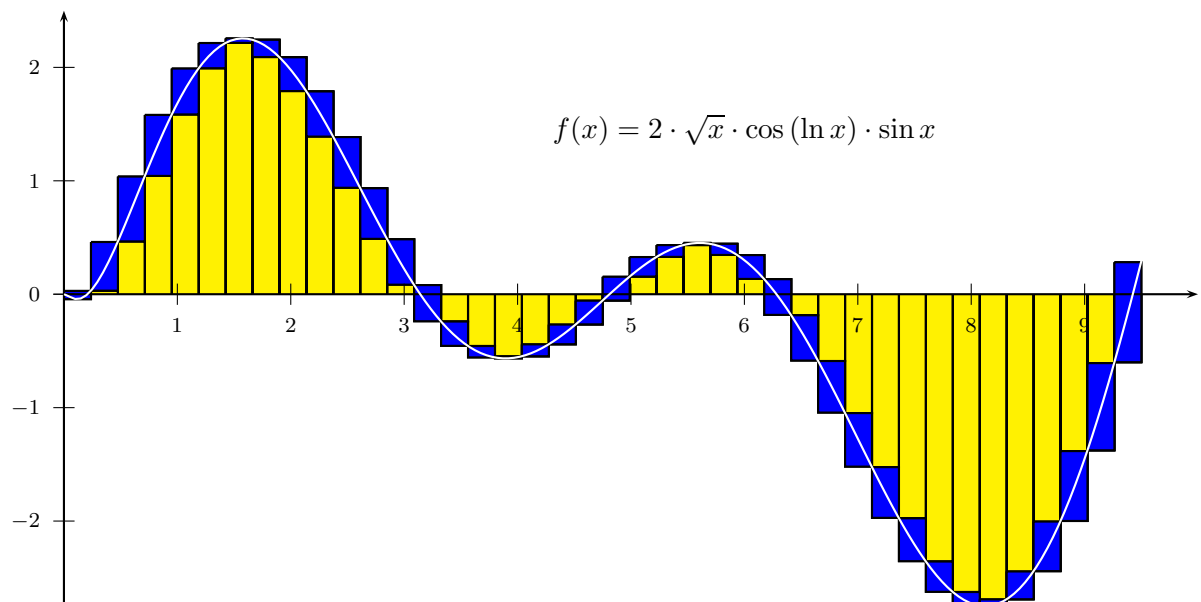
```



```

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

```



```

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

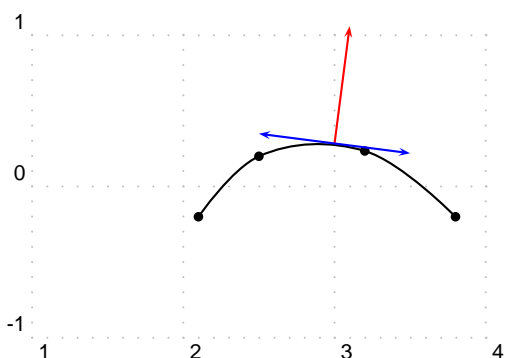
```

18. Tangent lines

There are two macros for plotting a tangent line or the tangent normal line. The first one is `\psTangentLine` which expects three pairs of coordinates, a x and a dx value. The second one is `\psplotTangent` which expects a function for the curve.

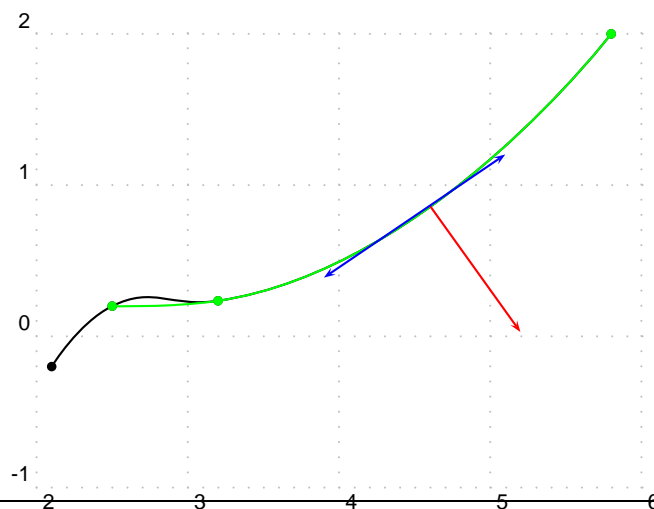
18.1. `\psTangentLine` and option `Tnormal`

```
\psTangentLine [Options] (x1,y1)(x2,y2)(x3,y3){x}{dx}
```



```
1 \psset{unit=2}
2 \begin{pspicture}[showgrid=true](1,-1)(4,1)
3   \pscurve[showpoints=true]
4     (2.1,-0.2)(2.5,0.2)(3.2,0.235)(3.8,-0.2)
5   \psTangentLine[Tnormal,arrows=->,
6     linecolor=red](2.5,0.2)(3.2,0.235)%
7     (3.8,-0.2){3}{0.1}
8   \psTangentLine[arrows=<->,
9     linecolor=blue](2.5,0.2)(3.2,0.235)%
10    (3.8,-0.2){3}{0.5}
11 \end{pspicture}
```

In special cases one has to use `curvature=1 1 1` for the macro `\pscurve` to get the same equation for the curve as `\psplotTangentLine` does.



```
1 \psset{unit=2}
2 \begin{pspicture}[showgrid=true](2,-1)(6,2)
3   \pscurve[showpoints=true,
4     curvature=1 1 1](2.1,-0.2)(2.5,0.2)(3.2,0.235)(5.8,2)
5   \pscurve[showpoints=true,linecolor=green,
6     curvature=1 1 1](2.5,0.2)(3.2,0.235)(5.8,2)
7   \psTangentLine[Tnormal,arrows=->,linecolor=red](2.5,0.2)(3.2,0.235)(5.8,2){4.6}{0.6}
8   \psTangentLine[arrows=<->,linecolor=blue](2.5,0.2)(3.2,0.235)(5.8,2){4.5}{0.6}
9 \end{pspicture}
```

18.2. `\psplotTangent` and option `Tnormal`

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

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

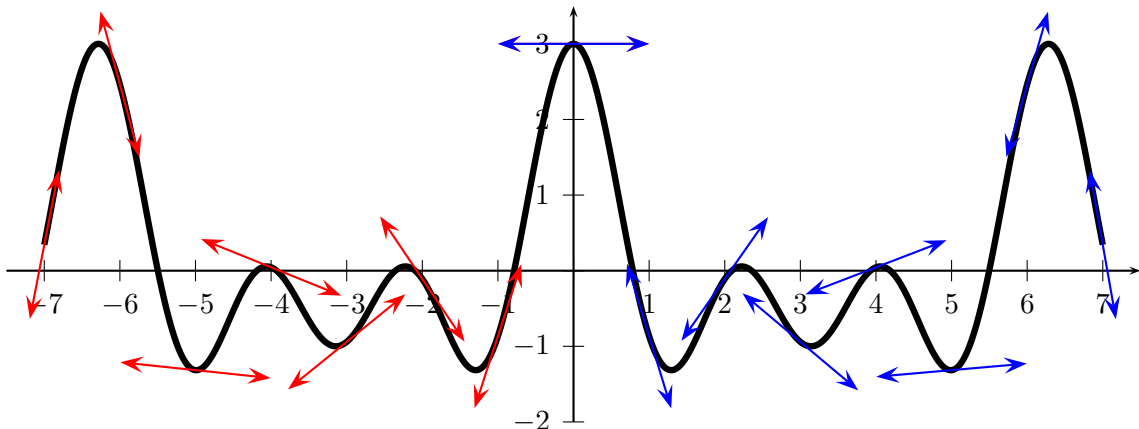
The macro expects three parameters:

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

dx : the dx to both sides of the x value

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

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



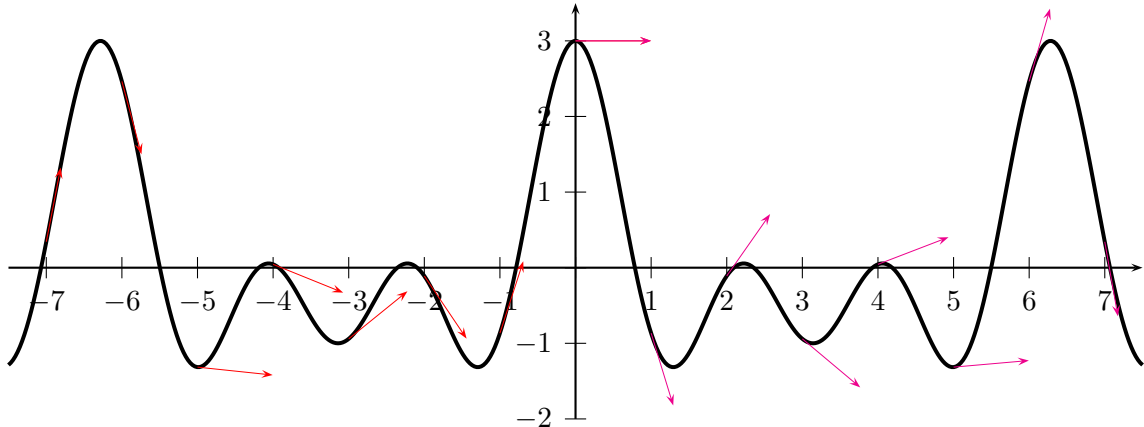
```
1 \def\F{x RadtoDeg dup dup cos exch 2 mul cos add exch 3 mul cos add}
2 \def\Fp{x RadtoDeg dup dup sin exch 2 mul sin 2 mul add exch 3 mul sin 3 mul
   add neg}
3 \psset{plotpoints=1001}
4 \begin{pspicture}(-7.5,-2.5)(7.5,4)%X\psgrid
5   \psaxes{->}(0,0)(-7.5,-2)(7.5,3.5)
6   \psplot[linewidth=3\pslinewidth]{-7}{7}{\F}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

```

10 \multido{\n=0+1}{8}{\psplotTangent[linecolor=blue, Derive=\Fp]{\n}{1}{\F}}
11 \end{pspicture}

```

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

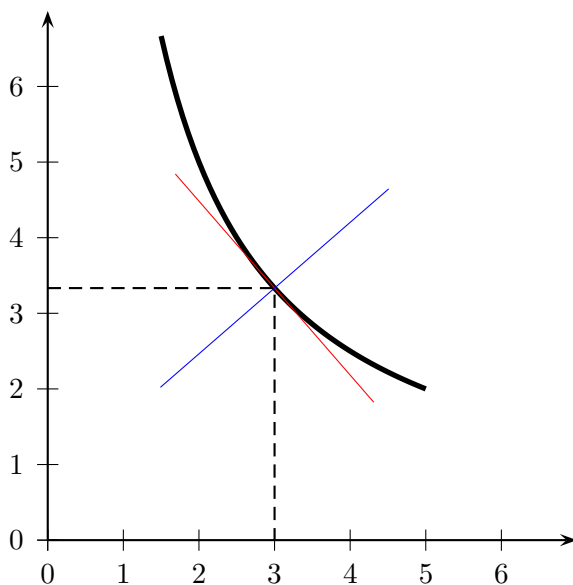


```

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

```

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

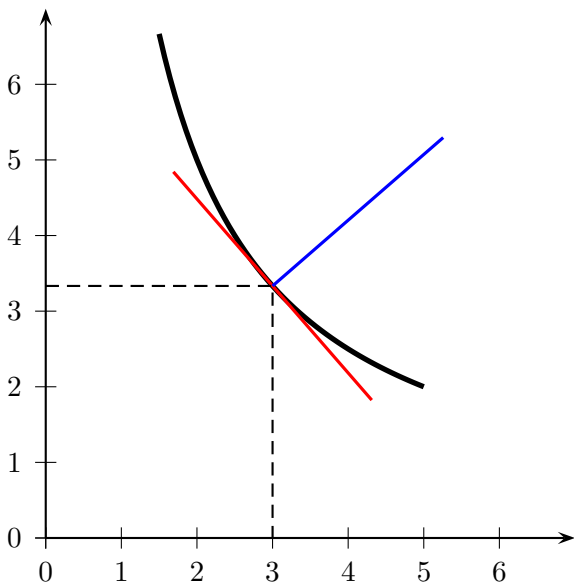


```

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

```

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



```

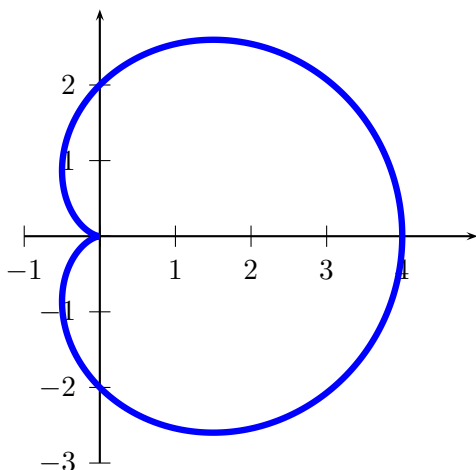
1 \begin{pspicture}(-0.5,-0.5)(7.25,7.25)
2   \def\Func{10 x div}
3   \psaxes[arrowscale=1.5]{->}(7,7)
4   \psplot[linewidth=2pt]{1.5}{5}{\Func}
5   \psplotTangent[linewidth=1.5\pslinewidth,
6     linecolor=red]{3}{2}{\Func}
7   \psplotTangent[linewidth=1.5\pslinewidth,
8     linecolor=blue,Tnormal]{3}{2}{\Func}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

```

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

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

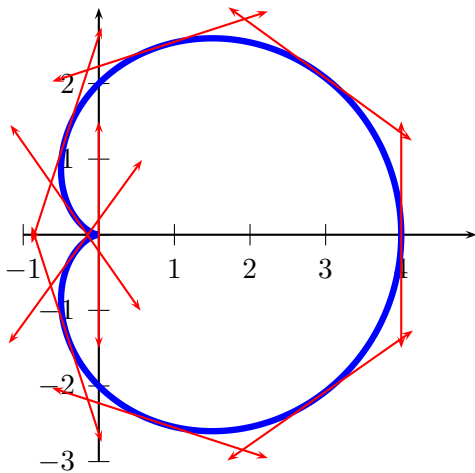
where $x = r \cdot \cos \theta$ and $y = r \cdot \sin \theta$



```

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

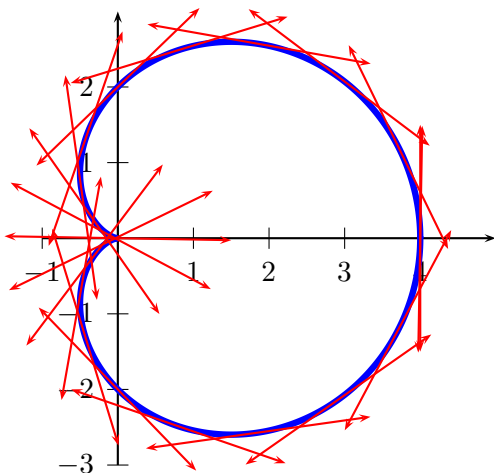
```



```

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

```



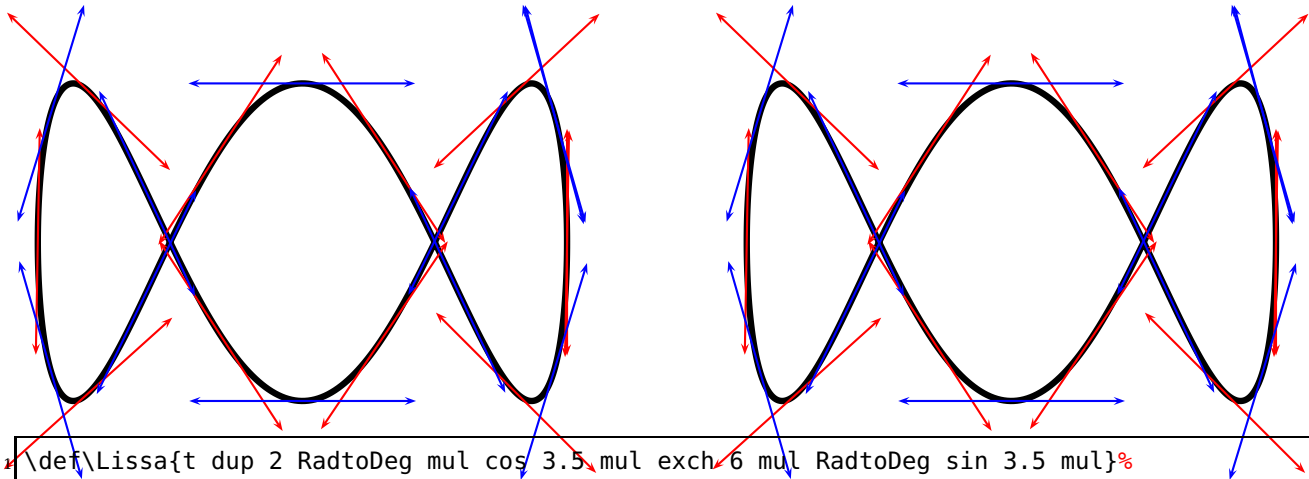
```

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

```

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

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



```

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

```

19. Successive derivatives of a function

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

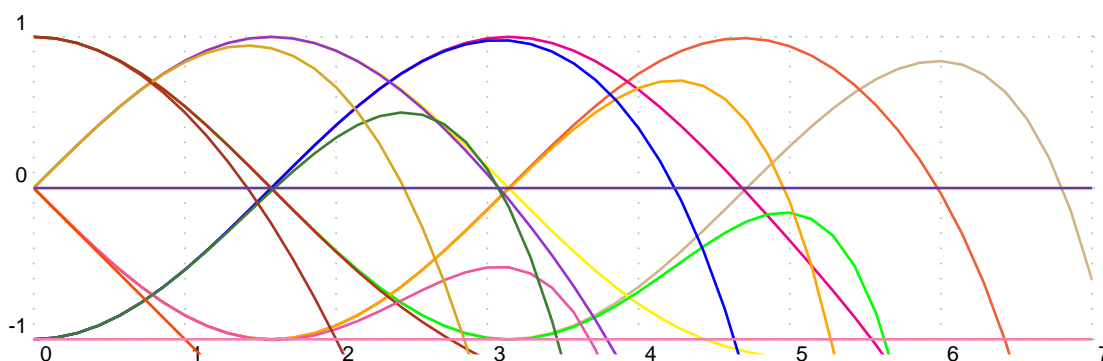
1. a positive integer which defines the order of the derivative; obviously 0 means the function itself!
2. a function of variable x which can be any function using common operators,

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

The following diagram contains the plot of the polynomial:

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

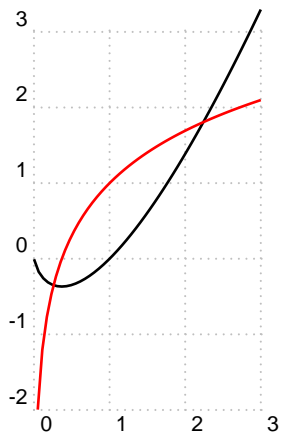
and of its first 15 derivatives. It is the sequence definition of the cosine.



```

1 \psset{unit=2}
2 \def\getColor#1{\ifcase#1 Tan\or Red\orange\or magenta\or yellow\or green\or Orange\or
3   blue\or
4   DarkOrchid\or BrickRed\or Rhodamine\or OliveGreen\or Goldenrod\or Mahogany\or
5   OrangeRed\or CarnationPink\or RoyalPurple\or Lavender\fi}
6 \begin{pspicture}[showgrid=true](0,-1.2)(7,1.5)
7   \psclip{\psframe[linestyle=none](0,-1.1)(7,1.1)}
8   \multido{\in=0+1}{16}{%
9     \psplot[linewidth=1pt,algebraic=true,linecolor=\getColor{\in}]{0}{7}
10    {Derive(\in,1-x^2/2+x^4/24-x^6/720+x^8/40320-x^10/3628800+x^12/479001600-x
11    ^14/87178291200)}}
12 \endpsclip
13 \end{pspicture}

```



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

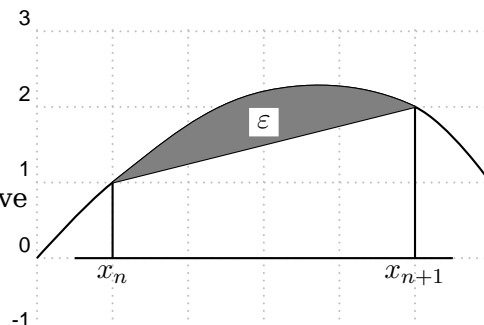
20. Variable step for plotting a curve

20.1. Theory

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

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

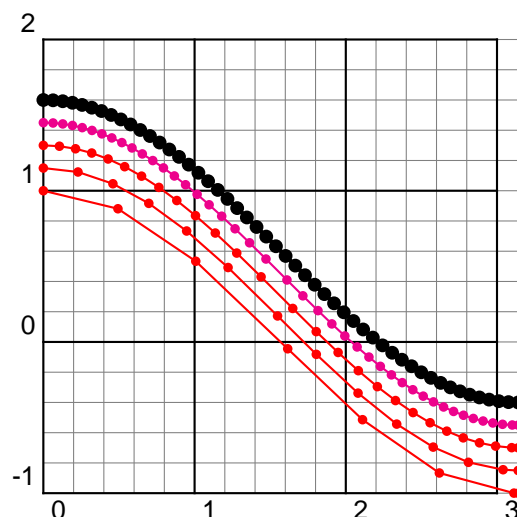
$M_2(f)$ is a majorant of the second derivative of f in the interval $[x_i; x_{i+1}]$.



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

20.2. The cosine

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



```

1 \psset{algebraic=true, VarStep=true, unit=2, showpoints=true, linecolor=red}
2 \begin{pspicture}[showgrid=true](-0,-1)(3.14,2)
3   \psplot[VarStepEpsilon=.01]{0}{3.14}{cos(x)}
4   \psplot[VarStepEpsilon=.001]{0}{3.14}{cos(x)+.15}
5   \psplot[VarStepEpsilon=.0001]{0}{3.14}{cos(x)+.3}

```

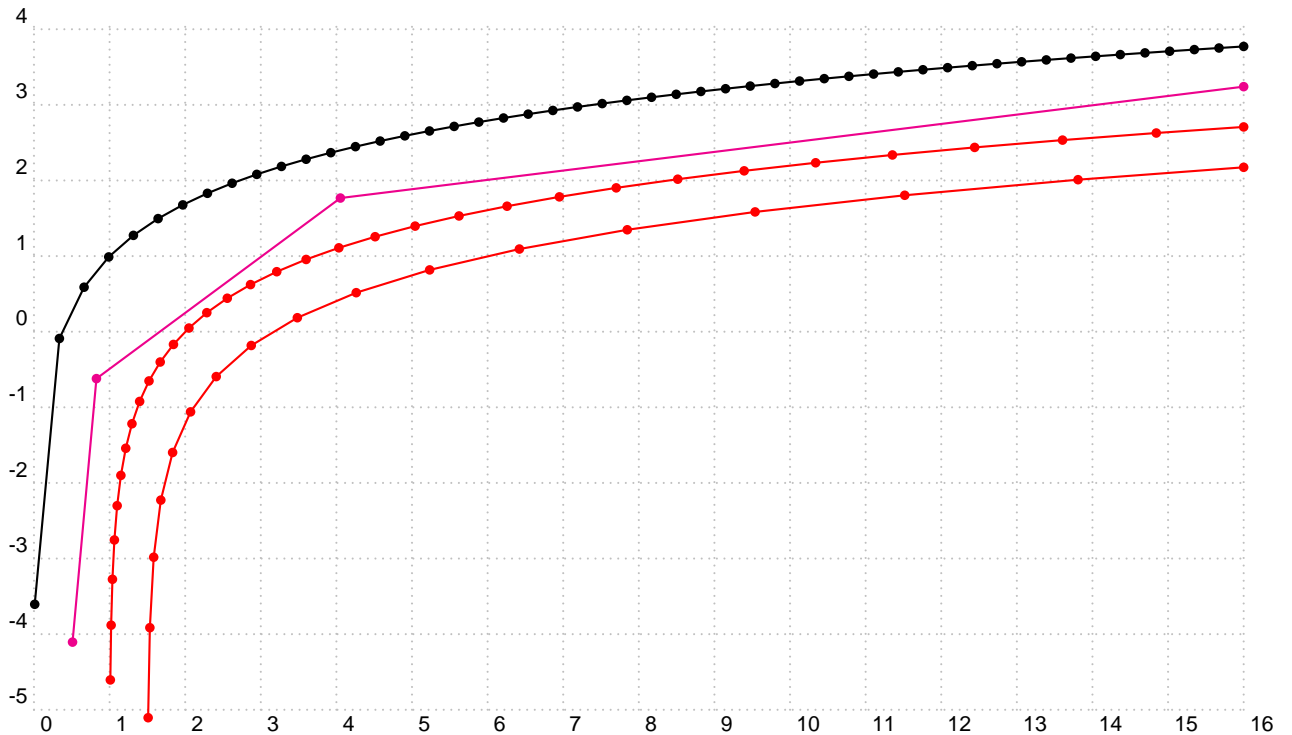
```

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

```

20.3. The Napierian Logarithm

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



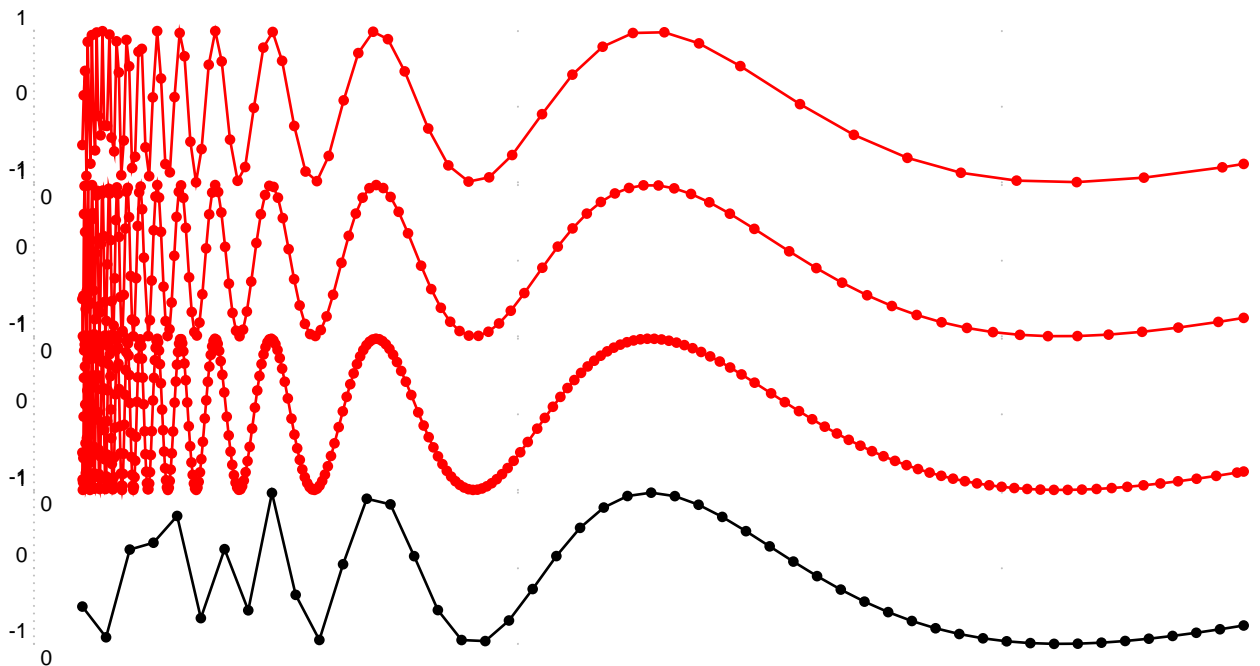
```

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

```

20.4. Sine of the inverse of x

Impossible to draw, but let's try!



```

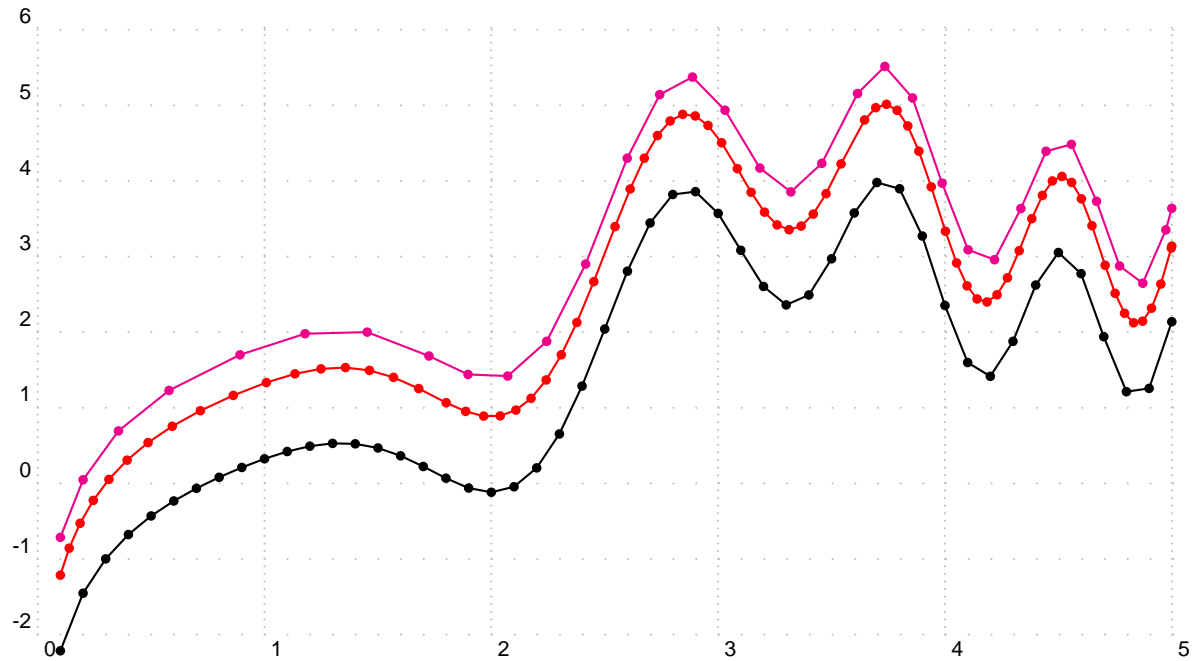
1 \psset{xunit=64,algebraic=true,VarStep,linecolor=red,showpoints=true,linewidth
   =1pt}
2 \begin{pspicture}[showgrid=true](0,-1)(.5,1)
3   \psplot[VarStepEpsilon=.0001]{.01}{.25}{sin(1/x)}
4 \end{pspicture}\
5 \begin{pspicture}[showgrid=true](0,-1)(.5,1)
6   \psplot[VarStepEpsilon=.00001]{.01}{.25}{sin(1/x)}
7 \end{pspicture}\
8 \begin{pspicture}[showgrid=true](0,-1)(.5,1)
9   \psplot[VarStepEpsilon=.000001]{.01}{.25}{sin(1/x)}
10 \end{pspicture}\
11 \begin{pspicture}[showgrid=true](0,-1)(.5,1)
12   \psplot[VarStep=false, linecolor=black]{.01}{.25}{sin(1/x)}
13 \end{pspicture}

```

20.5. A really complicated function

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

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

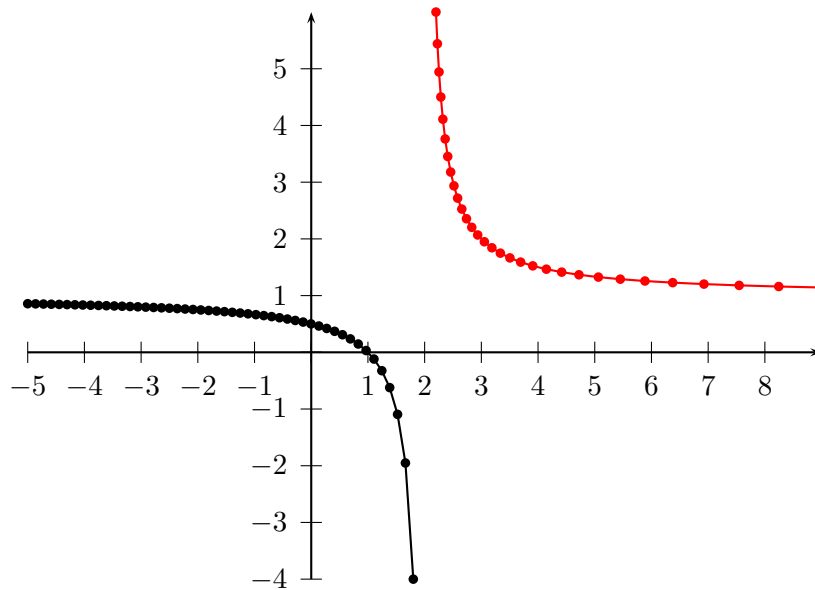


```

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

```

20.6. A hyperbola



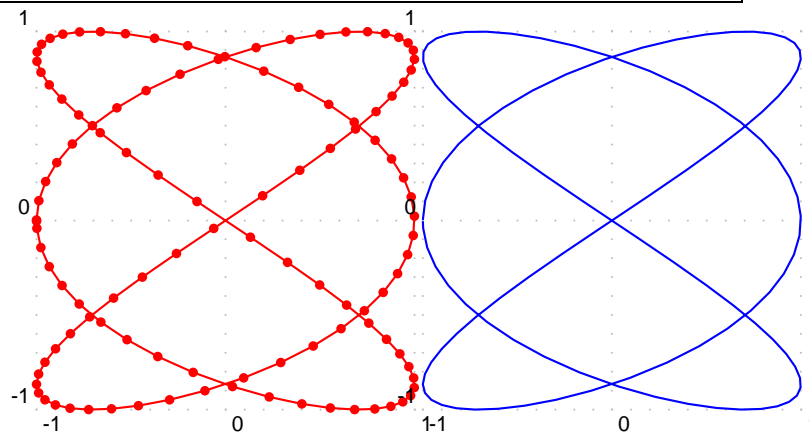
```

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

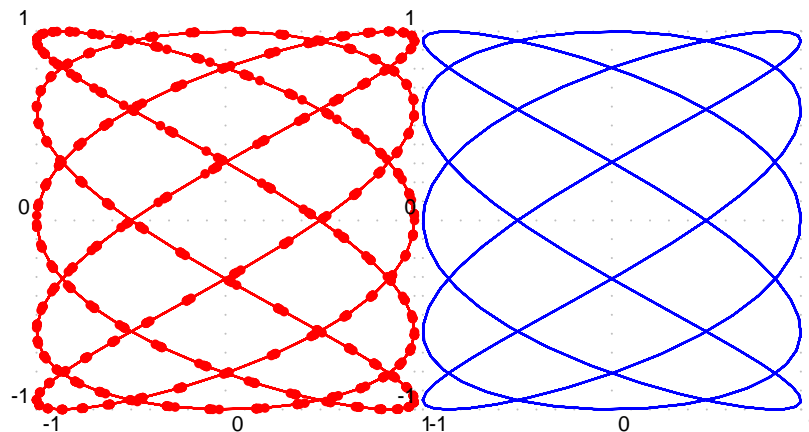
```


20.7. Using \parametricplot

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



```
1 \psset{unit=3}
2 \begin{pspicture}[showgrid=true](-1,-1)(1,1)
3 \parametricplot[algebraic=true,linecolor=red,VarStep=true, showpoints=true,
4   VarStepEpsilon=.0001]
5   {-3.14}{3.14}{cos(3*t)|sin(2*t)}
6 \end{pspicture}
7 \begin{pspicture}[showgrid=true](-1,-1)(1,1)
8 \parametricplot[algebraic=true,linecolor=blue,VarStep=true, showpoints=false,
9   VarStepEpsilon=.0001]
10  {-3.14}{3.14}{cos(3*t)|sin(2*t)}
11 \end{pspicture}
```

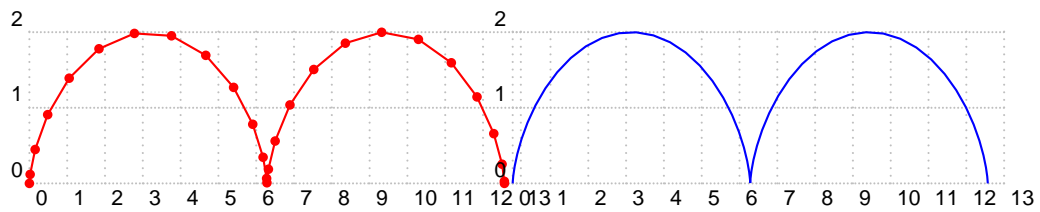


```
1 \psset{unit=2.5}
2 \begin{pspicture}[showgrid=true](-1,-1)(1,1)
3 \parametricplot[algebraic=true,linecolor=red,VarStep=true, showpoints=true,
4   VarStepEpsilon=.0001]
5   {0}{47.115}{cos(5*t)|sin(3*t)}
6 \end{pspicture}
7 \begin{pspicture}[showgrid=true](-1,-1)(1,1)
8 \parametricplot[algebraic=true,linecolor=blue,VarStep=true,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

```

10 {0}{47.115}{cos(5*t)|sin(3*t)}
11 \end{pspicture}

```



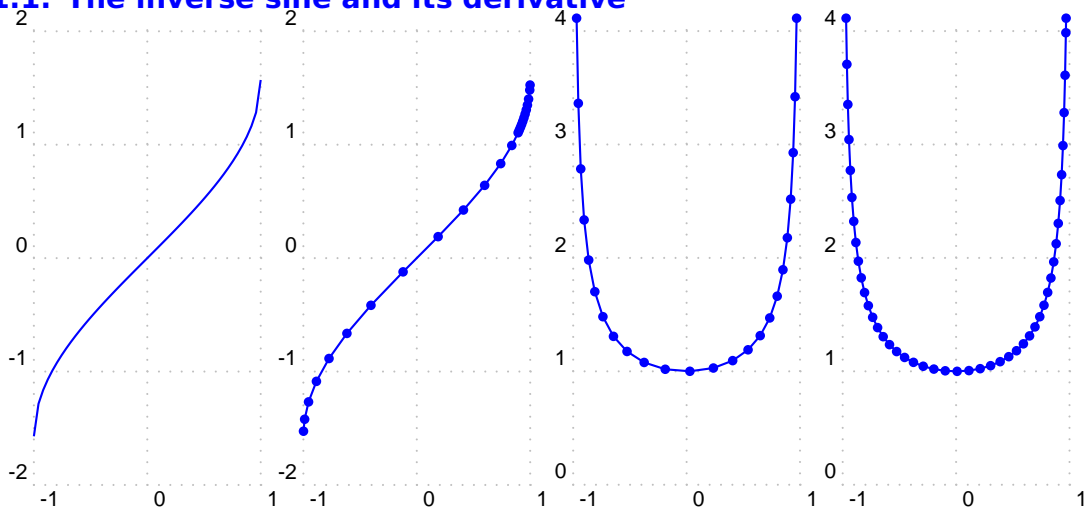
```

1 \psset{xunit=.5}
2 \begin{pspicture}[showgrid=true](0,0)(12.566,2)
3 \parametricplot[algebraic=true,linecolor=red,VarStep, showpoints=true,
4   VarStepEpsilon=.01]{0}{12.566}{t+cos(-t-Pi/2)|1+sin(-t-Pi/2)}
5 \end{pspicture}
6 %
7 \begin{pspicture}[showgrid=true](0,0)(12.566,2)
8 \parametricplot[algebraic=true,linecolor=blue,VarStep, showpoints=false,
9   VarStepEpsilon=.001]{0}{12.566}{t+cos(-t-Pi/2)|1+sin(-t-Pi/2)}
10 \end{pspicture}

```

21. New math functions and their derivatives

21.1. The inverse sine and its derivative



```

1 \psset{unit=1.5}
2 \begin{pspicture}[showgrid=true](-1,-2)(1,2)
3 \psplot[linecolor=blue,algebraic=true]{-1}{1}{asin(x)}
4 \end{pspicture}
5 \hspace{1em}
6 \psset{algebraic=true, VarStep, VarStepEpsilon=.001, showpoints=true}
7 \begin{pspicture}[showgrid=true](-1,-2)(1,2)
8 \psplot[linecolor=blue]{-.999}{.999}{asin(x)}
9 \end{pspicture}
10 \hspace{1em}

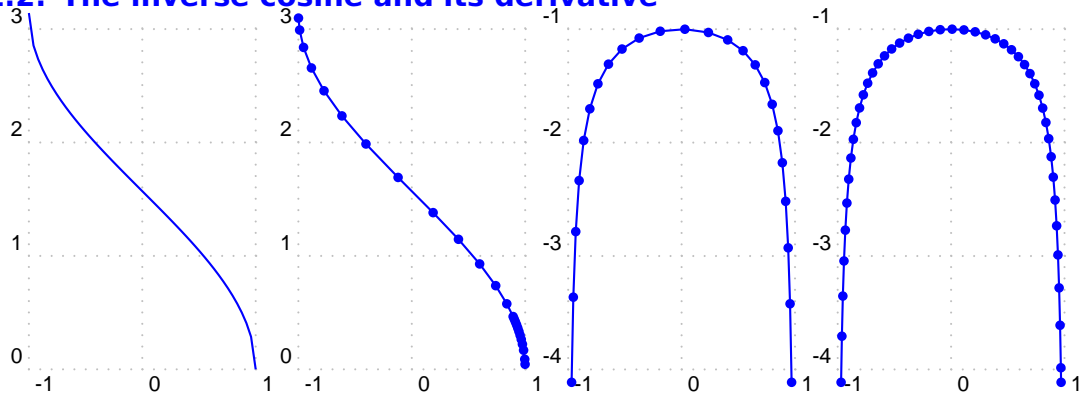
```

```

11 \begin{pspicture}[showgrid=true](-1,0)(1,4)
12   \psplot[linecolor=red]{-.97}{.97}{Derive(1,asin(x))}
13 \end{pspicture}
14 \hspace{1em}
15 \psset{algebraic=true, VarStep, VarStepEpsilon=.0001, showpoints=true}
16 \begin{pspicture}[showgrid=true](-1,0)(1,4)
17   \psplot[linecolor=red]{-.97}{.97}{Derive(1,asin(x))}
18 \end{pspicture}

```

21.2. The inverse cosine and its derivative

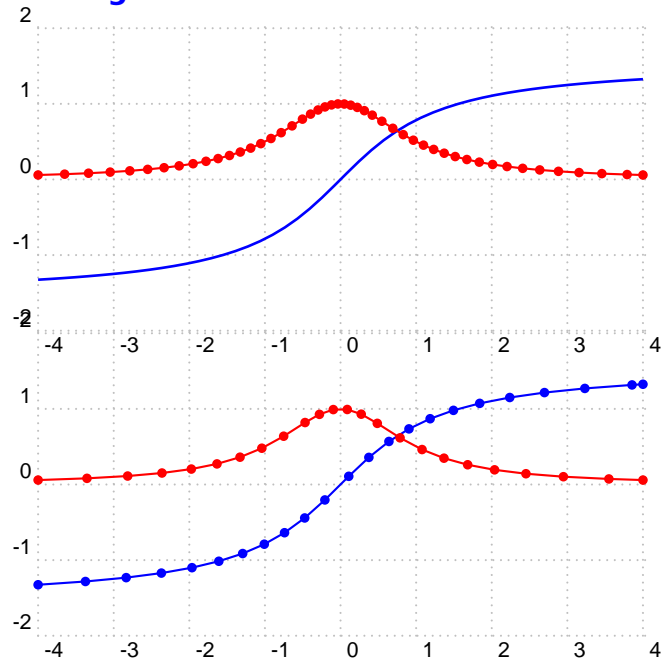


```

1 \psset{unit=1.5}
2 \begin{pspicture}[showgrid=true](-1,0)(1,3)
3   \psplot[linecolor=blue,algebraic=true]{-1}{1}{acos(x)}
4 \end{pspicture}
5 \hspace{1em}
6 \psset{algebraic=true, VarStep, VarStepEpsilon=.001, showpoints=true}
7 \begin{pspicture}[showgrid=true](-1,0)(1,3)
8   \psplot[linecolor=blue]{-.999}{.999}{acos(x)}
9 \end{pspicture}
10 \hspace{1em}
11 \begin{pspicture}[showgrid=true](-1,-4)(1,-1)
12   \psplot[linecolor=red]{-.97}{.97}{Derive(1,acos(x))}
13 \end{pspicture}
14 \hspace{1em}
15 \psset{algebraic=true, VarStep, VarStepEpsilon=.0001, showpoints=true}
16 \begin{pspicture}[showgrid=true](-1,-4)(1,-1)
17   \psplot[linecolor=red]{-.97}{.97}{Derive(1,acos(x))}
18 \end{pspicture}

```

21.3. The inverse tangent and its derivative

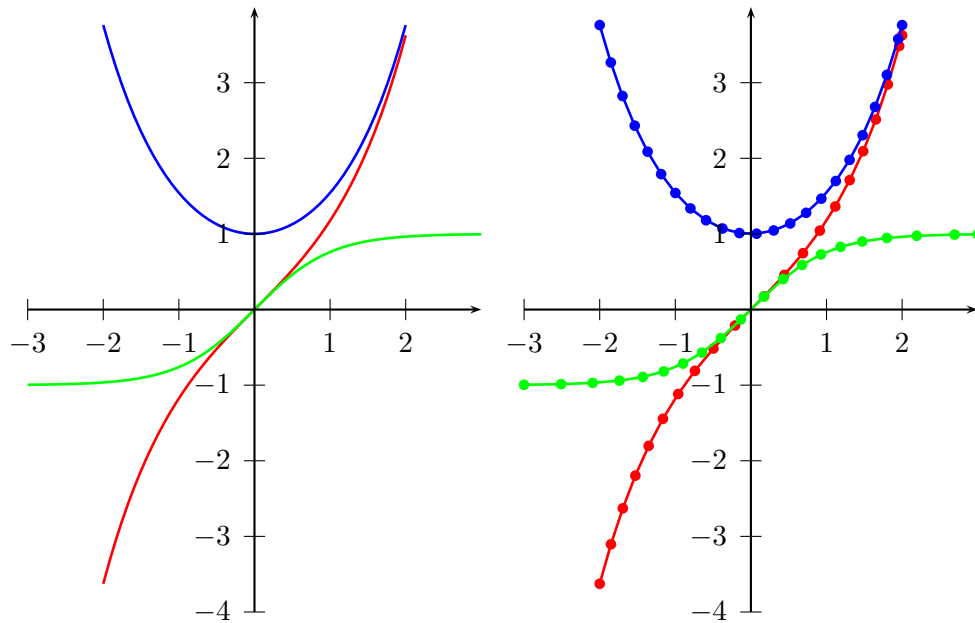


```

1 \begin{pspicture}[showgrid=true](-4,-2)(4,2)
2 \psset{algebraic=true}
3 \psplot[linecolor=blue,linewidth=1pt]{-4}{4}{atg(x)}
4 \psplot[linecolor=red,VarStep, VarStepEpsilon=.0001, showpoints=true]{-4}{4}{
5   Derive(1,atg(x))}
6 \end{pspicture}
7 \hspace{1em}
8 \begin{pspicture}[showgrid=true](-4,-2)(4,2)
9 \psset{algebraic=true, VarStep, VarStepEpsilon=.001, showpoints=true}
10 \psplot[linecolor=blue]{-4}{4}{atg(x)}
11 \psplot[linecolor=red]{-4}{4}{Derive(1,atg(x))}
12 \end{pspicture}

```

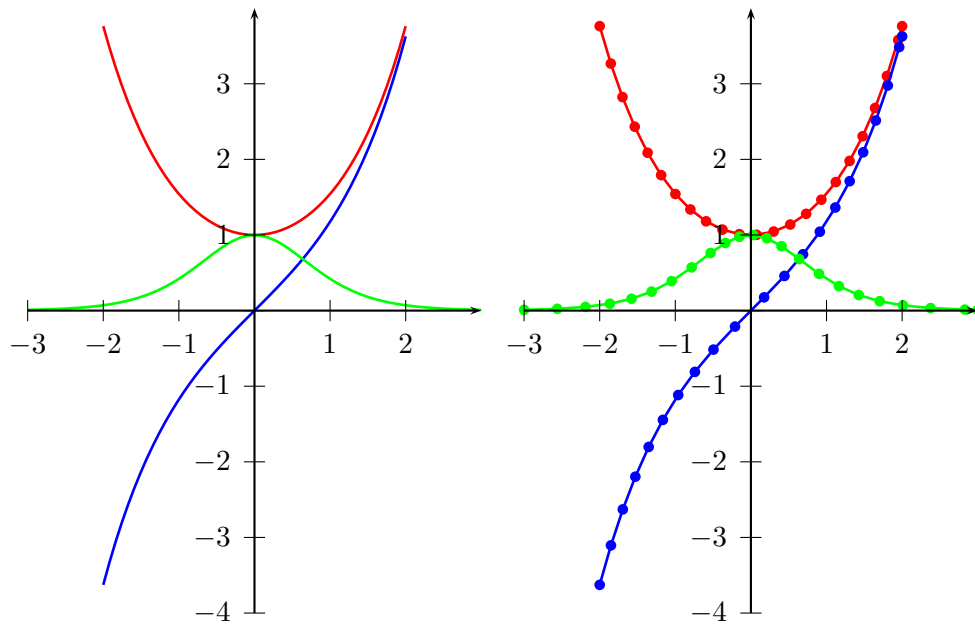
21.4. Hyperbolic functions



```

1 \begin{pspicture}(-3,-4)(3,4)
2 \psset{algebraic=true}
3 \psplot[linecolor=red,linewidth=1pt]{-2}{2}{sh(x)}
4 \psplot[linecolor=blue,linewidth=1pt]{-2}{2}{ch(x)}
5 \psplot[linecolor=green,linewidth=1pt]{-3}{3}{th(x)}
6 \psaxes{->}(0,0)(-3,-4)(3,4)
7 \end{pspicture}
8 \hspace{1em}
9 \begin{pspicture}(-3,-4)(3,4)
10 \psset{algebraic=true, VarStep=true, VarStepEpsilon=.001, showpoints=true}
11 \psplot[linecolor=red,linewidth=1pt]{-2}{2}{sh(x)}
12 \psplot[linecolor=blue,linewidth=1pt]{-2}{2}{ch(x)}
13 \psplot[linecolor=green,linewidth=1pt]{-3}{3}{th(x)}
14 \psaxes{->}(0,0)(-3,-4)(3,4)
15 \end{pspicture}

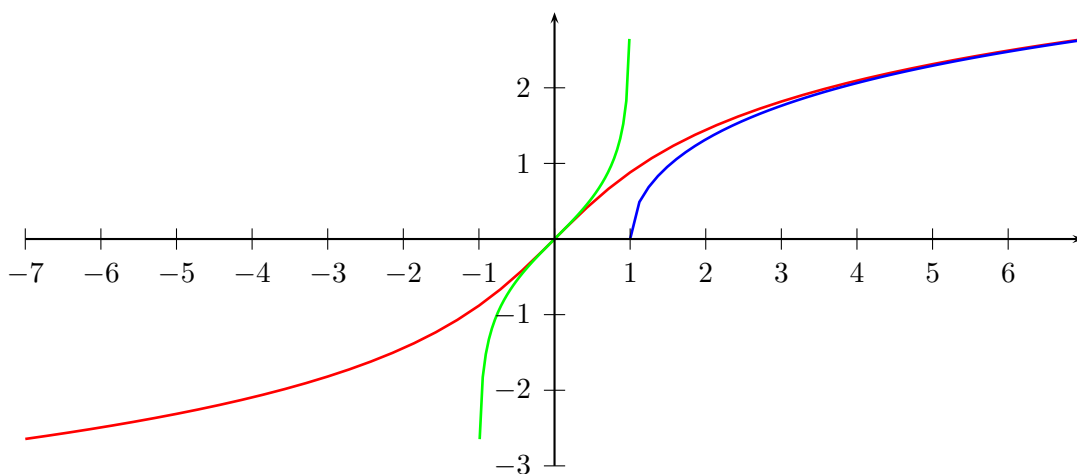
```

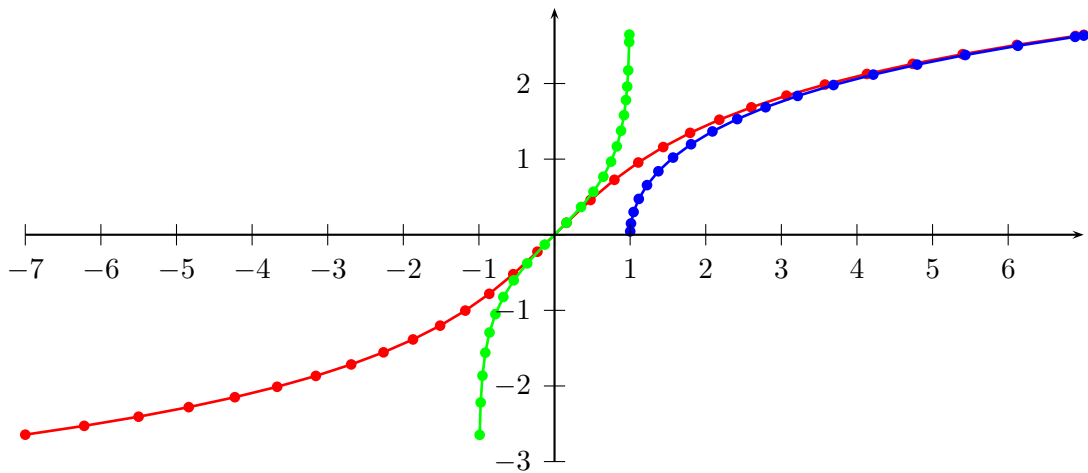


```

1 \begin{pspicture}(-3,-4)(3,4)
2 \psset{algebraic=true,linewidth=1pt}
3 \psplot[linecolor=red,linewidth=1pt]{-2}{2}{Derive(1,sh(x))}
4 \psplot[linecolor=blue,linewidth=1pt]{-2}{2}{Derive(1,ch(x))}
5 \psplot[linecolor=green,linewidth=1pt]{-3}{3}{Derive(1,th(x))}
6 \psaxes{->}(0,0)(-3,-4)(3,4)
7 \end{pspicture}
8 \hspace{1em}
9 \begin{pspicture}(-3,-4)(3,4)
10 \psset{algebraic=true,VarStep=true,VarStepEpsilon=.001,showpoints=true}
11 \psplot[linecolor=red,linewidth=1pt]{-2}{2}{Derive(1,sh(x))}
12 \psplot[linecolor=blue,linewidth=1pt]{-2}{2}{Derive(1,ch(x))}
13 \psplot[linecolor=green,linewidth=1pt]{-3}{3}{Derive(1,th(x))}
14 \psaxes{->}(0,0)(-3,-4)(3,4)
15 \end{pspicture}

```

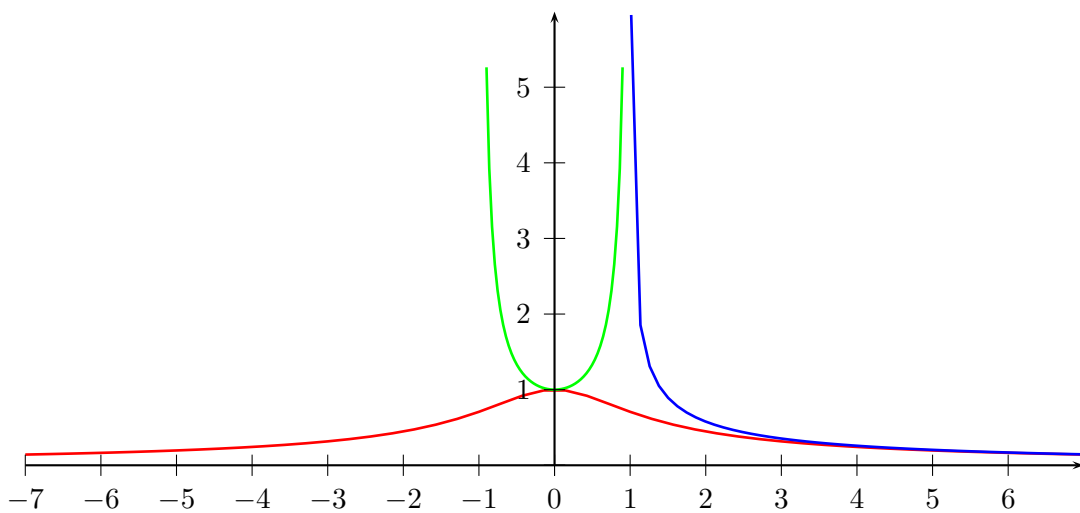


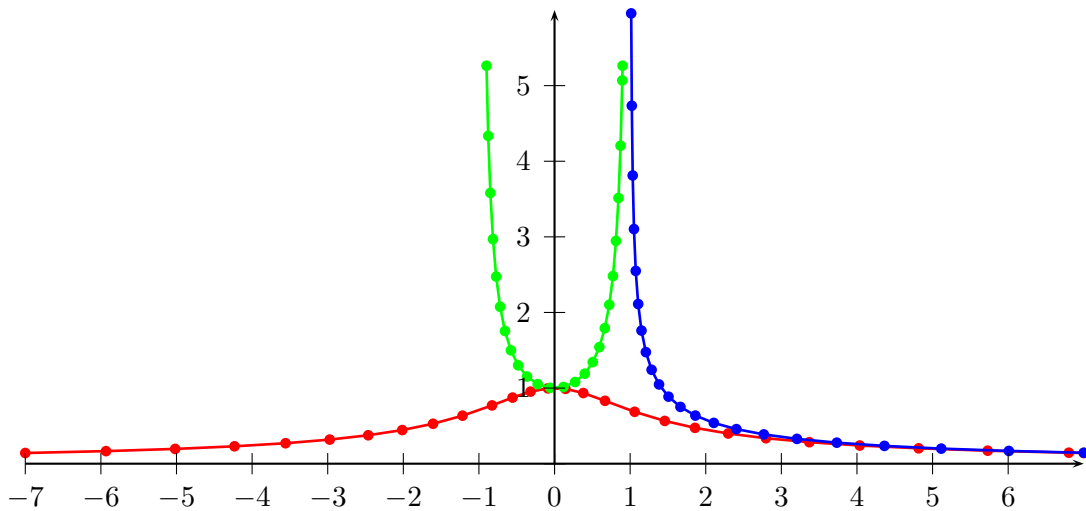


```

1 \begin{pspicture}(-7,-3)(7,3)
2 \psset{algebraic=true}
3 \psplot[linecolor=red,linewidth=1pt]{-7}{7}{Argsh(x)}
4 \psplot[linecolor=blue,linewidth=1pt]{1}{7}{Argch(x)}
5 \psplot[linecolor=green,linewidth=1pt]{-.99}{.99}{Argth(x)}
6 \psaxes{->}(0,0)(-7,-3)(7,3)
7 \end{pspicture}\[\baselineskip]
8 \begin{pspicture}(-7,-3)(7,3)
9 \psset{algebraic=true, VarStep, VarStepEpsilon=.001, showpoints=true}
10 \psplot[linecolor=red,linewidth=1pt]{-7}{7}{Argsh(x)}
11 \psplot[linecolor=blue,linewidth=1pt]{1.001}{7}{Argch(x)}
12 \psplot[linecolor=green,linewidth=1pt]{-.99}{.99}{Argth(x)}
13 \psaxes{->}(0,0)(-7,-3)(7,3)
14 \end{pspicture}

```





```

1 \begin{pspicture}(-7,-0.5)(7,6)
2 \psset{algebraic=true}
3 \psplot[linecolor=red,linewidth=1pt]{-7}{7}{Derive(1,Argsh(x))}
4 \psplot[linecolor=blue,linewidth=1pt]{1.014}{7}{Derive(1,Argch(x))}
5 \psplot[linecolor=green,linewidth=1pt]{-.9}{.9}{Derive(1,Argth(x))}
6 \psaxes{->}(0,0)(-7,0)(7,6)
7 \end{pspicture}\[\baselineskip]
8 \begin{pspicture}(-7,-0.5)(7,6)
9 \psset{algebraic=true}
10 \psset{algebraic=true, VarStep=true, VarStepEpsilon=.001, showpoints=true}
11 \psplot[linecolor=red,linewidth=1pt]{-7}{7}{Derive(1,Argsh(x))}
12 \psplot[linecolor=blue,linewidth=1pt]{1.014}{7}{Derive(1,Argch(x))}
13 \psplot[linecolor=green,linewidth=1pt]{-.9}{.9}{Derive(1,Argth(x))}
14 \psaxes{->}(0,0)(-7,0)(7,6)
15 \end{pspicture}

```


22. \psplotDiffEqn – solving differential equations

A differential equation of first order is like

$$y' = f(x, y, y') \quad (1)$$

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

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

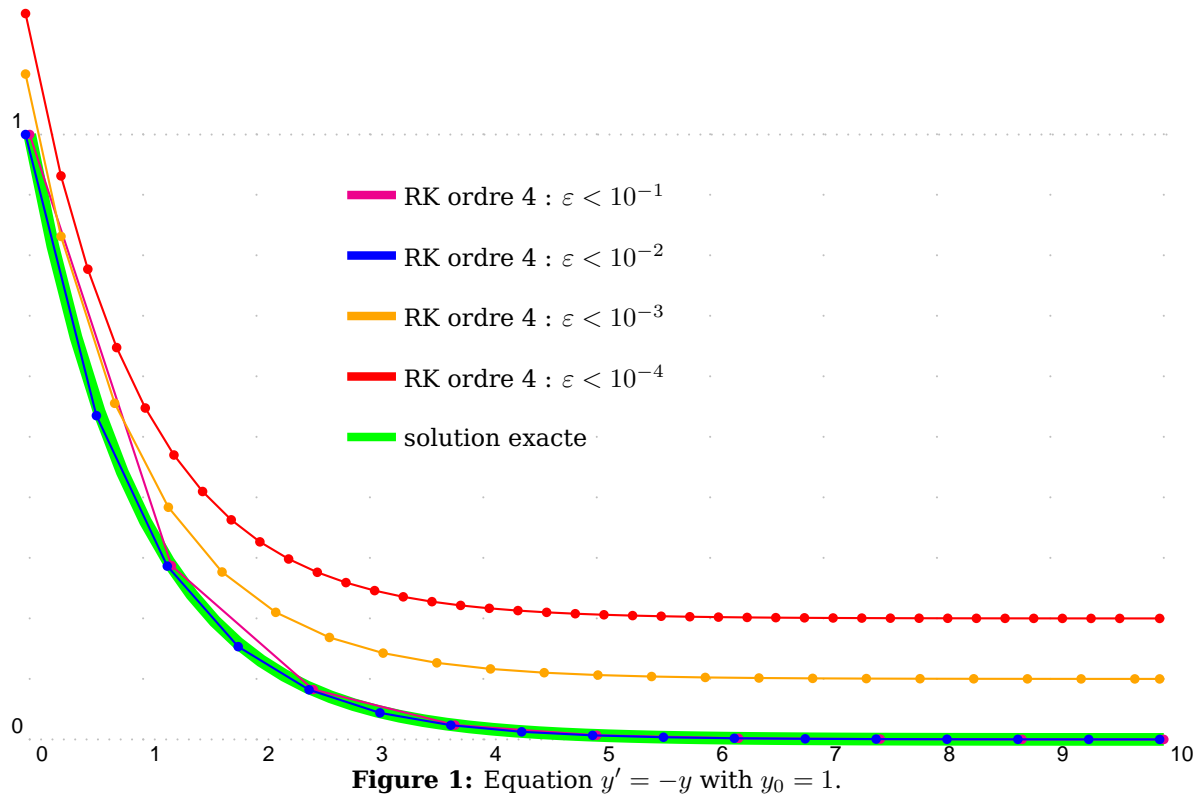
- options: the \psplotDiffEqn specific options and all other of PSTricks, which make sense;
- x_0 : the start value;
- x_1 : the end value of the definition interval;
- y_0 : the initial values for $y(x_0)$ $y'(x_0)$...;
- $f(x, y, y', \dots)$: the differential equation, depending to the number of initial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

The new options are:

- method: integration method (euler for order 1 euler method, rk4 for 4th order Runge-Kutta method);
- whichabs: select the abscissa for plotting the graph, by default it is x , but you can specify a number which represent a position in the vector y ;
- whichord: same as precedent for the ordinate, by default $y(0)$;
- plotfuncx: describe a ps function for the abscissa, parameter whichabs becomes useless;
- plotfuncy: idem for the ordinate;
- buildvector: boolean parameter for specifying the input-output of the f description:
 - true* (default): y is put on the stack element by element, y' must be given in the same way;
 - false* : y is put on the stack as a vector, y' must be returned in the same way;
- algebraic=true: algebraic=true description for f , buildvector parameter is useless when activating this option.

22.1. Variable step for differential equations

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



```

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

```

```

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

```

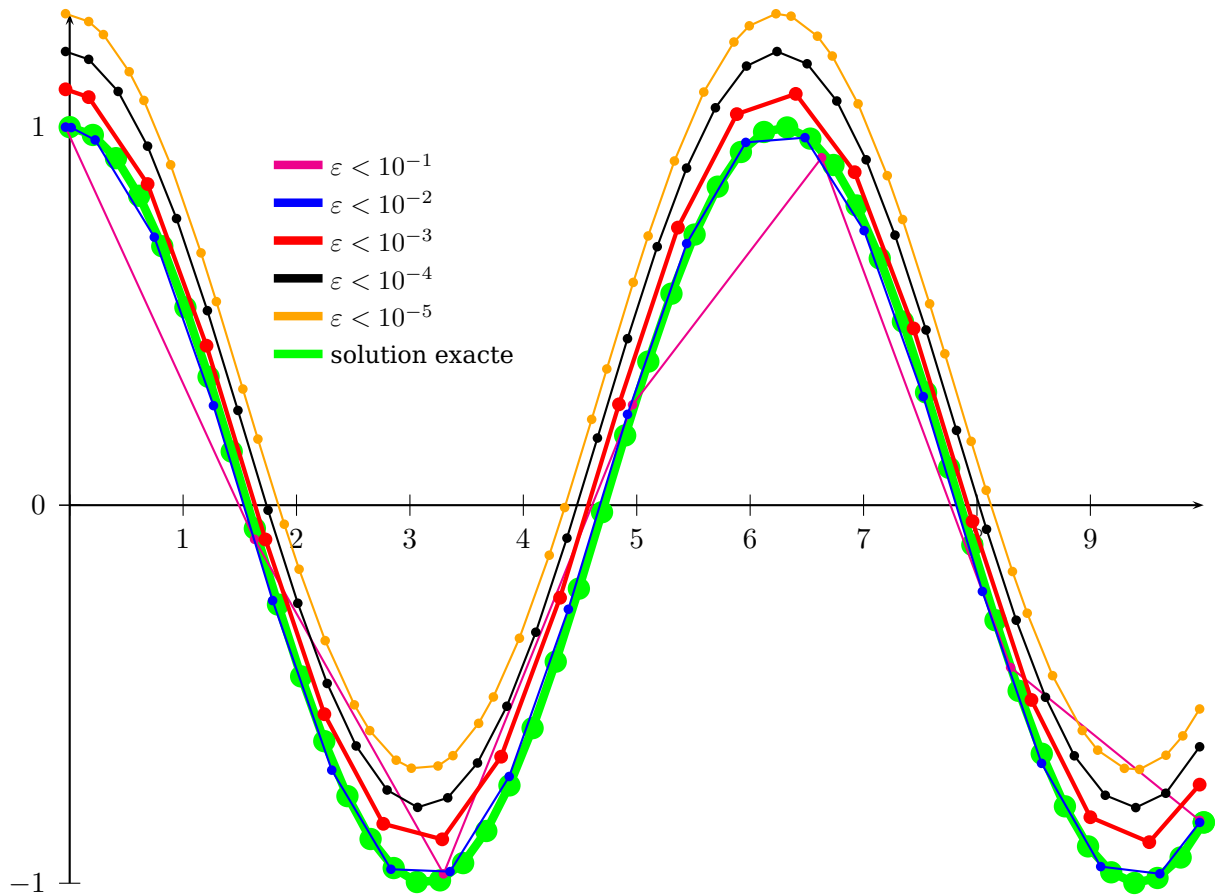


Figure 2: Equation $y'' = -y$

```

1 \def\Funct{exh neg}
2 \psset{xunit=1.5, yunit=5, method=varrkiv, showpoints=true}%%
3 \def\quatrepi{12.5663706144}
4 \begin{pspicture}(0,-1)(10,1.3)
5 \psaxes{->}(0,0)(0,-1)(10,1.3)
6 \psplot[linewidth=4\pslinewidth, linecolor=green, algebraic=true]{0}{10}{cos(
7 \rput(0,.0){\psplotDiffEqn[linecolor=magenta, plotpoints=7, varsteptol
8 \rput(0,.0){\psplotDiffEqn[linecolor=blue, plotpoints=201, varsteptol
9 \rput(0,.1){\psplotDiffEqn[linecolor=red, varsteptol
10 \rput(0,.2){\psplotDiffEqn[linecolor=black, varsteptol=.0001]{0}{10}{1 0}{\
Funct}}

```

```

11 \rput(0,.3){\psplotDiffEqn[linecolor=orange, varsteptol=.00001]{0}{10}{1 0}{\
    Funct}}
12 \psset{linewidth=4\pslinewidth,showpoints=false}
13 \rput*(2.3,.9){\psline[linecolor=magenta](-.75cm,0)}
14 \rput*[l](2.3,.9){\small $\varepsilon<10^{-1}$}
15 \rput*(2.3,.8){\psline[linecolor=blue](-.75cm,0)}
16 \rput*[l](2.3,.8){\small $\varepsilon<10^{-2}$}
17 \rput*(2.3,.7){\psline[linecolor=red](-.75cm,0)}
18 \rput*[l](2.3,.7){\small $\varepsilon<10^{-3}$}
19 \rput*(2.3,.6){\psline[linecolor=black](-.75cm,0)}
20 \rput*[l](2.3,.6){\small $\varepsilon<10^{-4}$}
21 \rput*(2.3,.5){\psline[linecolor=orange](-.75cm,0)}
22 \rput*[l](2.3,.5){\small $\varepsilon<10^{-5}$}
23 \rput*(2.3,.4){\psline[linecolor=green](-.75cm,0)}
24 \rput*[l](2.3,.4){\small solution exacte}
25 \end{pspicture}

```

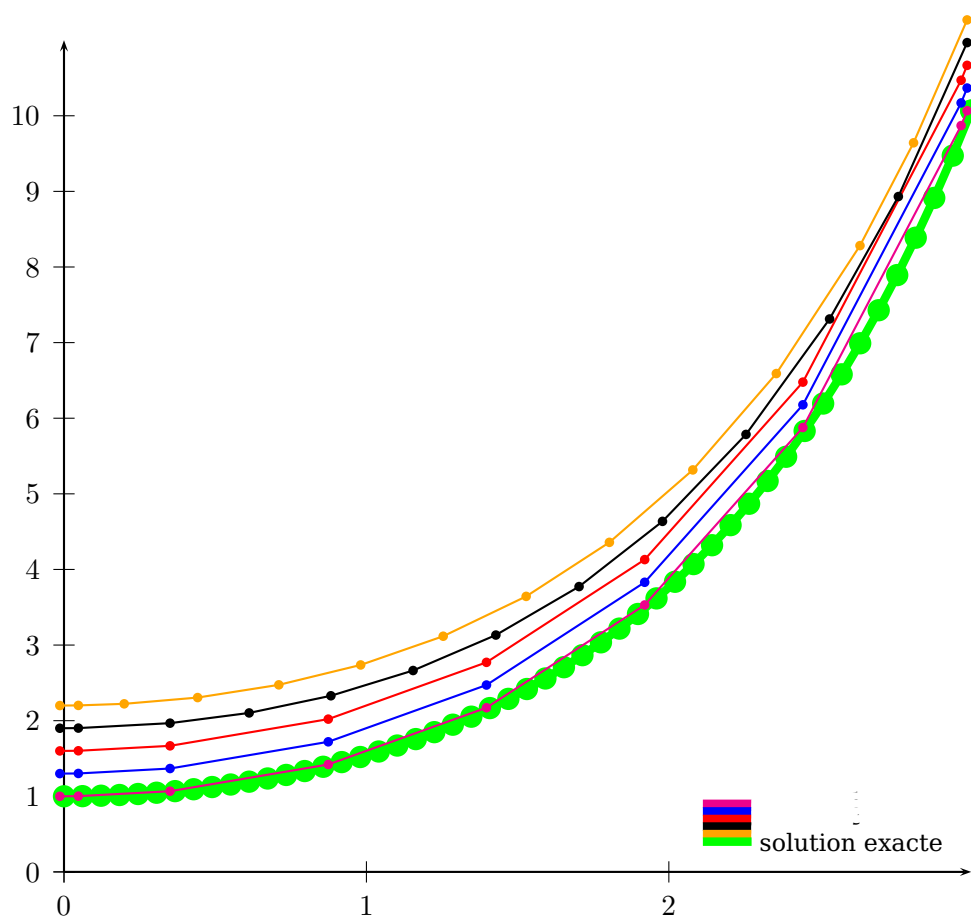


Figure 3: Equation $y'' = y$

```

1 \def\Funct{exch}
2 \psset{xunit=4, yunit=1, method=varrkiv, showpoints=true}%%
3 \def\quatrepi{12.5663706144}
4 \begin{pspicture}(0,-0.5)(3,11)

```

```

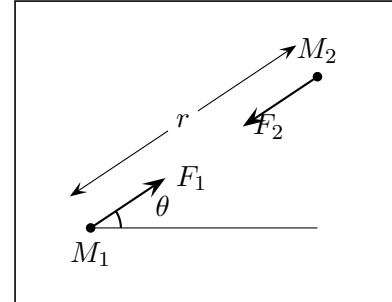
5 \psaxes{->}(0,0)(3,11)
6 \psplot[linewidth=4\pslinewidth, linecolor=green, algebraic=true]{0}{3}{ch(x)
  }
7 \rput(0,.0){\psplotDiffEqn[linecolor=magenta, varsteptol=.1]{0}{3}{1 0}{\
  Funct}}
8 \rput(0,.3){\psplotDiffEqn[linecolor=blue, varsteptol=.01]{0}{3}{1 0}{\Funct
  }}
9 \rput(0,.6){\psplotDiffEqn[linecolor=red, varsteptol=.001]{0}{3}{1 0}{\Funct
  }}
10 \rput(0,.9){\psplotDiffEqn[linecolor=black, varsteptol=.0001]{0}{3}{1 0}{\
  Funct}}
11 \rput(0,1.2){\psplotDiffEqn[linecolor=Orange, varsteptol=.00001]{0}{3}{1 0}{\
  Funct}}
12 \psset{linewidth=4\pslinewidth, showpoints=false}
13 \rput*(2.3,.9){\psline[linecolor=magenta](-.75cm,0)}
14 \rput*[l](2.3,.9){\small $\varepsilon < 10^{-1}$}
15 \rput*(2.3,.8){\psline[linecolor=blue](-.75cm,0)}
16 \rput*[l](2.3,.8){\small $\varepsilon < 10^{-2}$}
17 \rput*(2.3,.7){\psline[linecolor=red](-.75cm,0)}
18 \rput*[l](2.3,.7){\small $\varepsilon < 10^{-3}$}
19 \rput*(2.3,.6){\psline[linecolor=black](-.75cm,0)}
20 \rput*[l](2.3,.6){\small $\varepsilon < 10^{-4}$}
21 \rput*(2.3,.5){\psline[linecolor=Orange](-.75cm,0)}
22 \rput*[l](2.3,.5){\small $\varepsilon < 10^{-5}$}
23 \rput*(2.3,.4){\psline[linecolor=green](-.75cm,0)}
24 \rput*[l](2.3,.4){\small solution exacte}
25 \end{pspicture}

```

22.2. Equation of second order

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

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



```

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

```

Table 1: PostScript source code for the gravitational interaction

```

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

```

Table 2: Algebraic description for the gravitational interaction

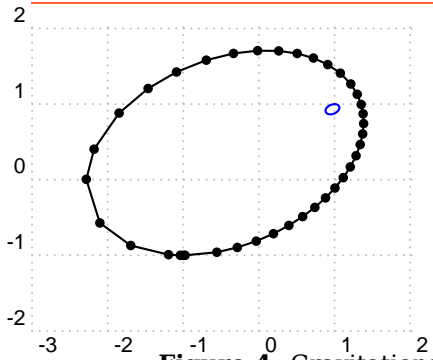


Figure 4: Gravitational interaction: fixed landmark, trajectory of the stars

```

1 \def\InitCond{ 1 1 .1 0 -1 -1 -2 0}
2 \begin{pspicture}[shift=-2,showgrid=true](-3,-1.75)(2,1.5)
3   \psplotDiffEqn[whichabs=0, whichord=1, linecolor=blue,
4     method=rk4, plotpoints=100]{0}{3.95}{\InitCond}{\Grav}
5   \psset{showpoints=true,whichabs=4, whichord=5}
6   \psplotDiffEqn[linecolor=black, method=varrkiv, varsteptol
7     =.0001, plotpoints=200]{0}{3.9}{\InitCond}{\Grav}
8 \end{pspicture}

```

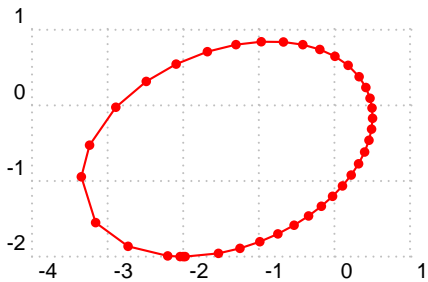


Figure 5: Gravitational interaction : landmark defined by one star

```

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

```

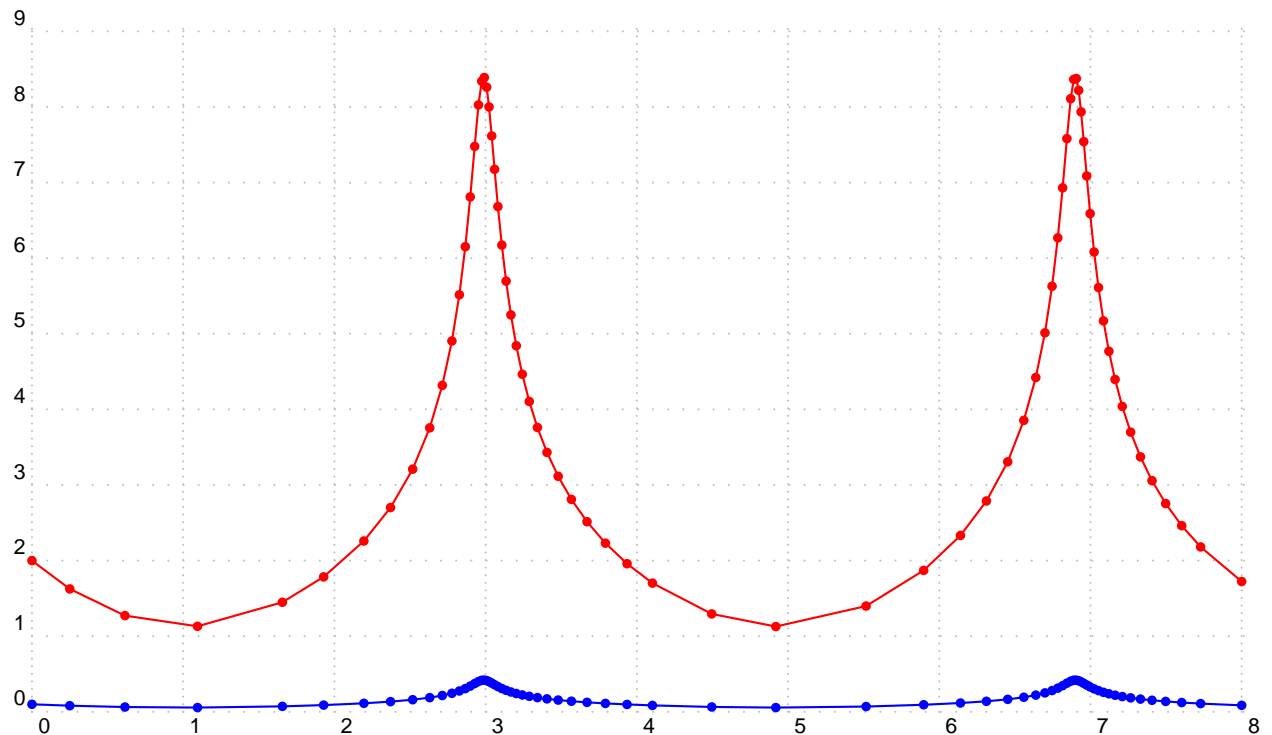


Figure 6: Gravitational interaction : speeds of the stars

```

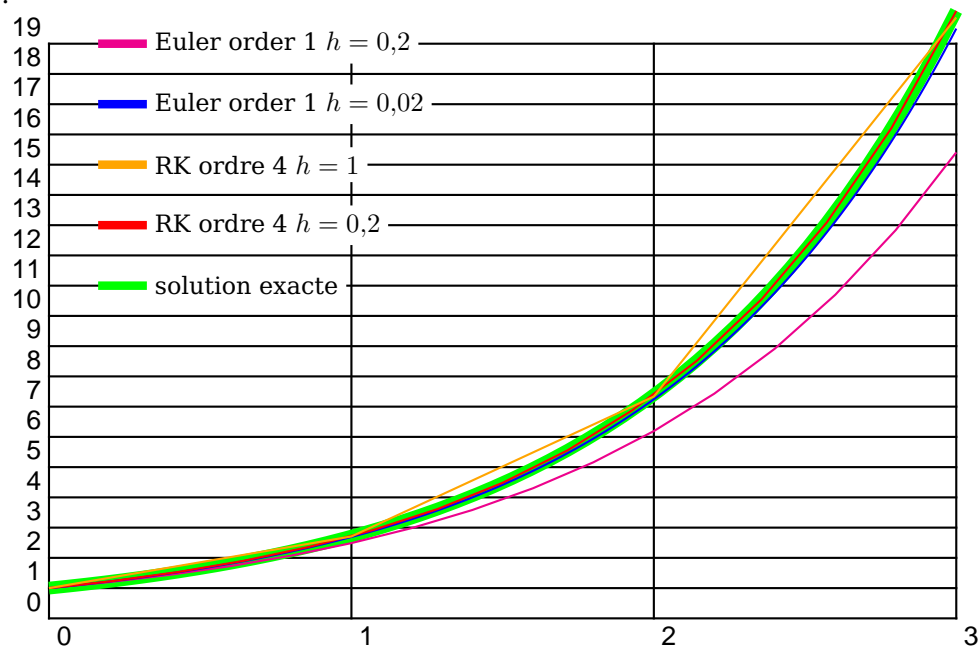
1 \psset{xunit=2}
2 \begin{pspicture}[showgrid=true](0,0)(8,9)
3   \psset{showpoints=true}

```

```
4 \psplotDiffEqn[linecolor=red, method=varrkiv, plotpoints=2, varsteptol=.0001,  
5   plotfuncy=dup 6 get dup mul exch 7 get dup mul add sqrt]{0}{8}{\InitCond  
   }\Grav}  
6 \psplotDiffEqn[linecolor=blue, method=varrkiv, plotpoints=2, varsteptol  
   =.0001,  
7   plotfuncy=dup 2 get dup mul exch 3 get dup mul add sqrt]{0}{8}{\InitCond  
   }\Grav}  
8 \end{pspicture}
```


Simple equation of first order $y' = y$

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



```

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

```

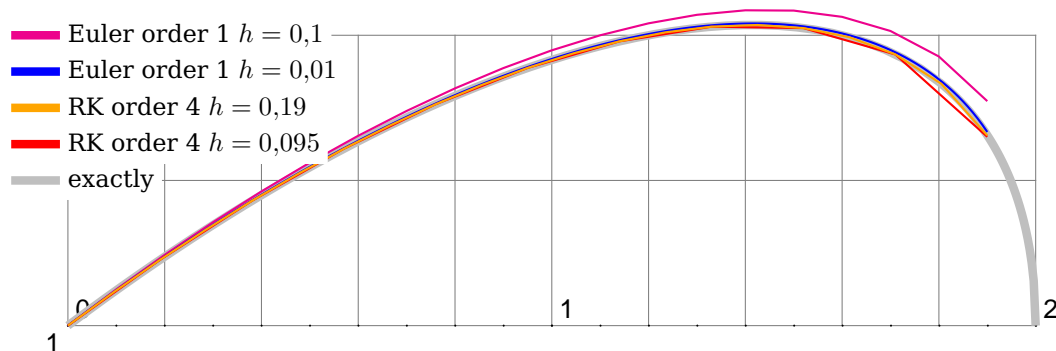
$$y' = \frac{2 - ty}{4 - t^2}$$

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

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

The following example uses $y_0 = 1$.

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

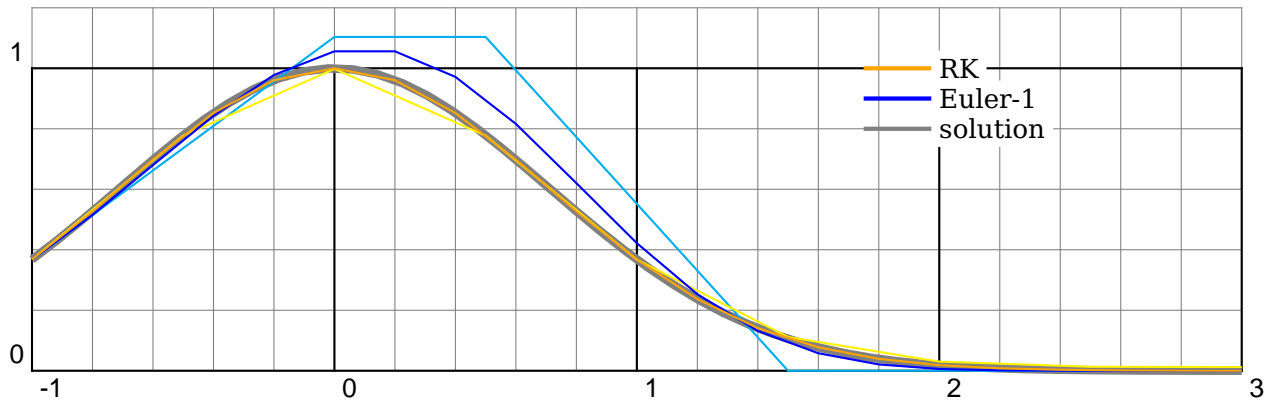


```
1 \psset{xunit=6.4, yunit=9.6, showpoints=false}
2 \begin{pspicture}(0,1)(2,1.7) \psgrid[subgriddiv=5]
3   { \psset{linewidth=4\pslinewidth, linecolor=lightgray}
4   \psplot{0}{1.8}{x dup dup mul 4 exch sub sqrt add 2 div}
5   \psplot{1.8}{2}{x dup dup mul 4 exch sub sqrt add 2 div} }
6 \def\InitCond{1}
7 \def\Func{x mul 2 exch sub 4 x dup mul sub div}
8 \psplotDiffEqn[linecolor=magenta, plotpoints=20]{0}{1.9}{\InitCond}{\Func}
9 \psplotDiffEqn[linecolor=blue, plotpoints=191]{0}{1.9}{\InitCond}{\Func}
10 \psplotDiffEqn[linecolor=red, method=rk4, plotpoints=11,%
11   algebraic=true]{0}{1.9}{\InitCond}{(2-x*y[0])/(4-x^2)}
12 \psplotDiffEqn[linecolor=orange, method=rk4, plotpoints=21,%
13   algebraic=true]{0}{1.9}{\InitCond}{(2-x*y[0])/(4-x^2)}
14 \psset{linewidth=4\pslinewidth}
15 \rput*(0.3,1.6){\psline[linecolor=magenta](-.75cm,0)}\rput*[l](0.3,1.6){\small Euler
16   order 1 $h=0{,}1$}
17 \rput*(0.3,1.55){\psline[linecolor=blue](-.75cm,0)}\rput*[l](0.3,1.55){\small Euler
18   order 1 $h=0{,}01$}
19 \rput*(0.3,1.5){\psline[linecolor=orange](-.75cm,0)}\rput*[l](0.3,1.5){\small RK order 4
20   $h=0{,}19$}
```

```
18 \rput*(0.3,1.45){\psline[linecolor=red](-.75cm,0)}\rput*[l](0.3,1.45){\small RK order 4 $  
    h=0{,}095$}  
19 \rput*(0.3,1.4){\psline[linecolor=lightgray](-.75cm,0)}\rput*[l](0.3,1.4){\small exactly}  
20 \end{pspicture}
```

$$y' = -2xy$$

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



```

1 \psset{unit=4}
2 \begin{pspicture}(-1,0)(3,1.1)\psgrid
3 \psplot[linewidth=4\pslinewidth,linecolor=gray]{-1}{3}{Euler x dup mul neg
   exp}
4 \psset{plotpoints=9}
5 \psplotDiffEqn[linecolor=cyan]{-1}{3}{1 Euler div}{x -2 mul mul}
6 \psplotDiffEqn[linecolor=yellow, method=rk4]{-1}{3}{1 Euler div}{x -2 mul mul
   }
7 \psset{plotpoints=21}
8 \psplotDiffEqn[linecolor=blue]{-1}{3}{1 Euler div}{x -2 mul mul}
9 \psplotDiffEqn[linecolor=orange, method=rk4]{-1}{3}{1 Euler div}{x -2 mul mul
   }
10 \psset{linewidth=2\pslinewidth}
11 \rput*(2,1){\psline[linecolor=orange](-0.25,0)}
12 \rput*[l](2,1){RK}
13 \rput*(2,.9){\psline[linecolor=blue](-0.25,0)}
14 \rput*[l](2,.9){\textsc{Euler}-1}
15 \rput*(2,.8){\psline[linecolor=gray](-0.25,0)}
16 \rput*[l](2,.8){solution}
17 \end{pspicture}

```

Spiral of Cornu

The integrals of Fresnel:

$$x = \int_0^t \cos \frac{\pi t^2}{2} dt \quad (2)$$

$$y = \int_0^t \sin \frac{\pi t^2}{2} dt \quad (3)$$

with

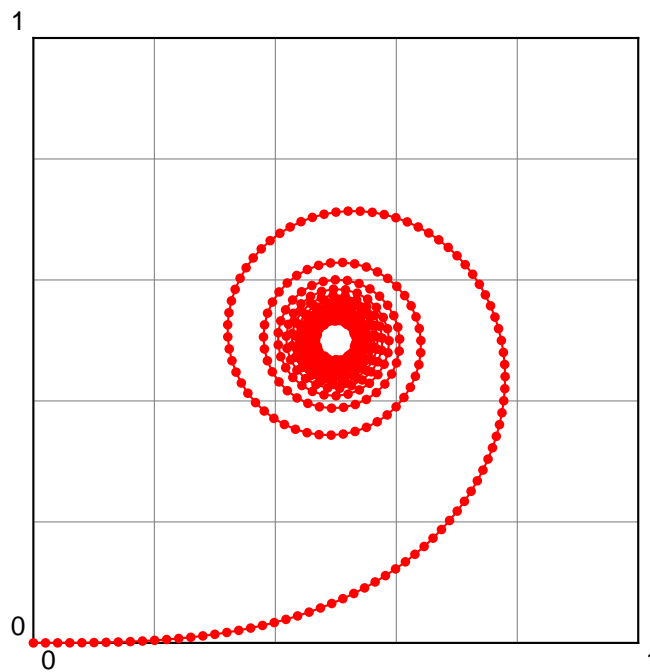
$$\dot{x} = \cos \frac{\pi t^2}{2} \quad (4)$$

$$\dot{y} = \sin \frac{\pi t^2}{2} \quad (5)$$

```

1 \psset{unit=8}
2 \begin{pspicture}(1,1)\psgrid[subgriddiv=5]
3 \psplotDiffEqn[whichabs=0,whichord=1,linecolor=red,method=rk4,algebraic=true,
4   %
5   plotpoints=500,showpoints=true]{0}{10}{0 0}{cos(Pi*x^2/2)|sin(Pi*x^2/2)}
6 \end{pspicture}

```



Lotka-Volterra

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

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

$$\dot{R} = a \cdot R - b \cdot R \cdot F \quad (6)$$

$$\dot{F} = e \cdot b \cdot R \cdot F - c \cdot F \quad (7)$$

where the parameters are defined by:

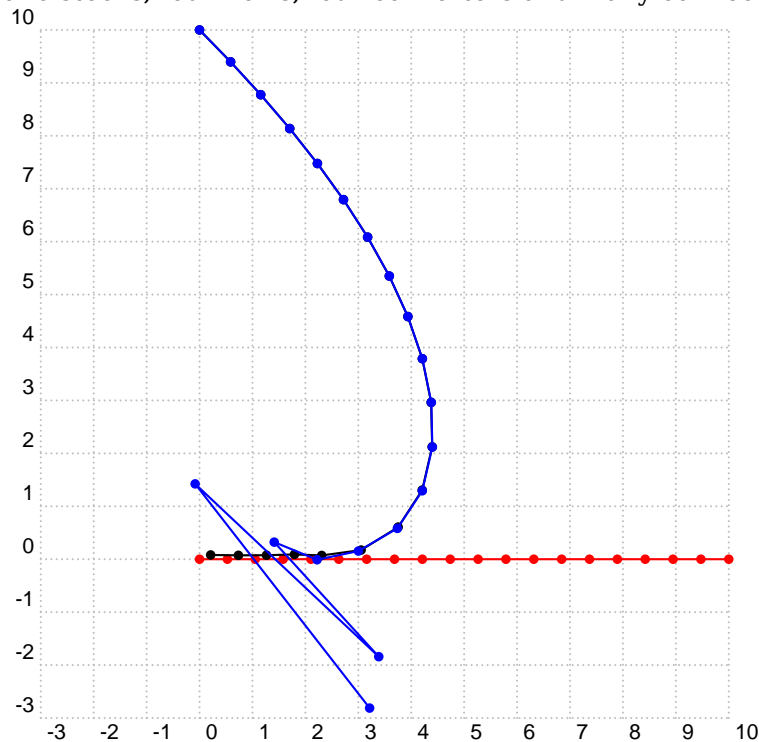
a is the natural growth rate of rabbits in the absence of predation,

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

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

e is the efficiency of turning predated rabbits into foxes.

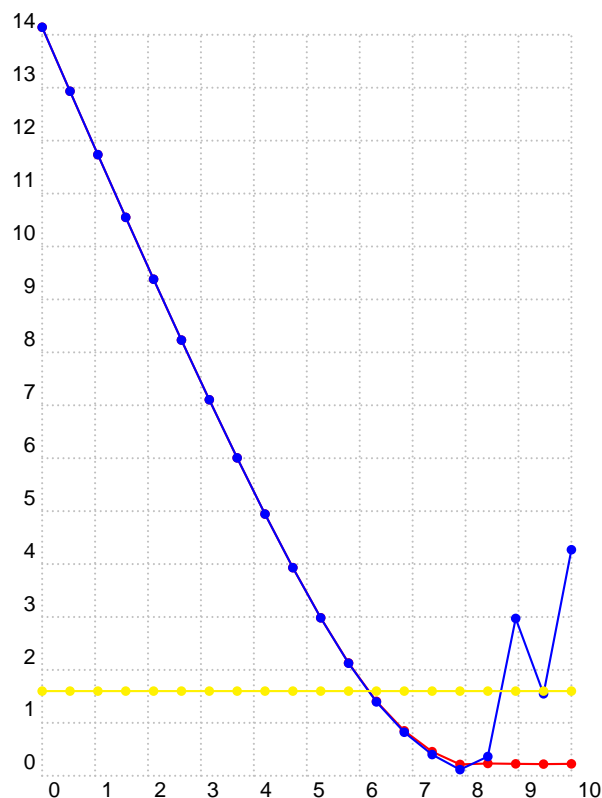
The Stella model representing the Lotka-Volterra model will be slightly more complex than the single species models we've dealt with before. The main difference is that our model will have two stocks (reservoirs), one for each species. Each species will have its own birth and death rates. In addition,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



```

1 \def\InitCond{ 0 10 10}%% xa ya xl
2 \def\Faiglelapin{\Vaigle*(y[2]-y[0])/sqrt(y[1]^2+(y[2]-y[0])^2)|%
3     -\Vaigle*y[1]/sqrt(y[1]^2+(y[2]-y[0])^2)|%
4     -\Vlapin}
5 \def\Vlapin{1} \def\Vaigle{1.6}
6 \psset{unit=.7,subgriddiv=0,gridcolor=lightgray,method=adams,algebraic=true,%
7     plotpoints=20,showpoints=true}
8 \begin{pspicture}[showgrid=true](-3,-3)(10,10)
9 \psplotDiffEqn[plotfuncy=pop 0,whichabs=2,linicolor=red]{0}{10}{\InitCond}{\
10     Faiglelapin}
11 \psplotDiffEqn[whichabs=0,whichord=1,linicolor=black,method=rk4]{0}{10}{\InitCond}{\
12     Faiglelapin}
13 \psplotDiffEqn[whichabs=0,whichord=1,linicolor=blue]{0}{10}{\InitCond}{\Faiglelapin}
14 \end{pspicture}

```



```

1 \def\InitCond{ 0 10 10}%% xa ya xl
2 \def\Faiglelapin{\Vaigle*(y[2]-y[0])/sqrt(y[1]^2+(y[2]-y[0])^2)|%
3     -\Vaigle*y[1]/sqrt(y[1]^2+(y[2]-y[0])^2)|%
4     -\Vlapin}
5 \def\Vlapin{1} \def\Vaigle{1.6}
6 \psset{unit=.7,subgriddiv=0,gridcolor=lightgray,method=adams,algebraic=true,%
7     plotpoints=20,showpoints=true}
8 \begin{pspicture}[showgrid=true](10,12)
9 \psplotDiffEqn[plotfuncy=dup 1 get dup mul exch dup 0 get exch 2 get sub dup
10     mul add sqrt,linicolor=red,method=rk4]{0}{10}{\InitCond}{\Faiglelapin}
11 \psplotDiffEqn[plotfuncy=dup 1 get dup mul exch dup 0 get exch 2 get sub dup

```

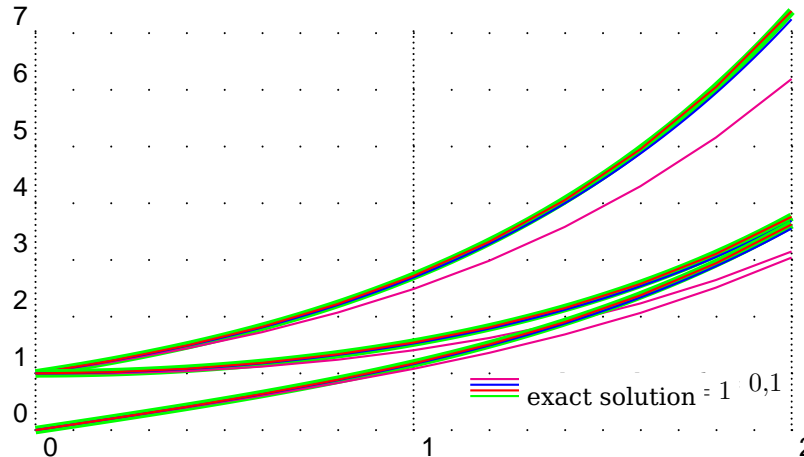
```

12 \mul add sqrt, linecolor=blue]{0}{10}{\InitCond}{\Faiglelapin}
13 \psplotDiffEqn[plotfuncy=pop Func aload pop pop dup mul exch dup mul add sqrt,
14   linecolor=yellow]{0}{10}{\InitCond}{\Faiglelapin}
15 \end{pspicture}

```

$$y'' = y$$

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

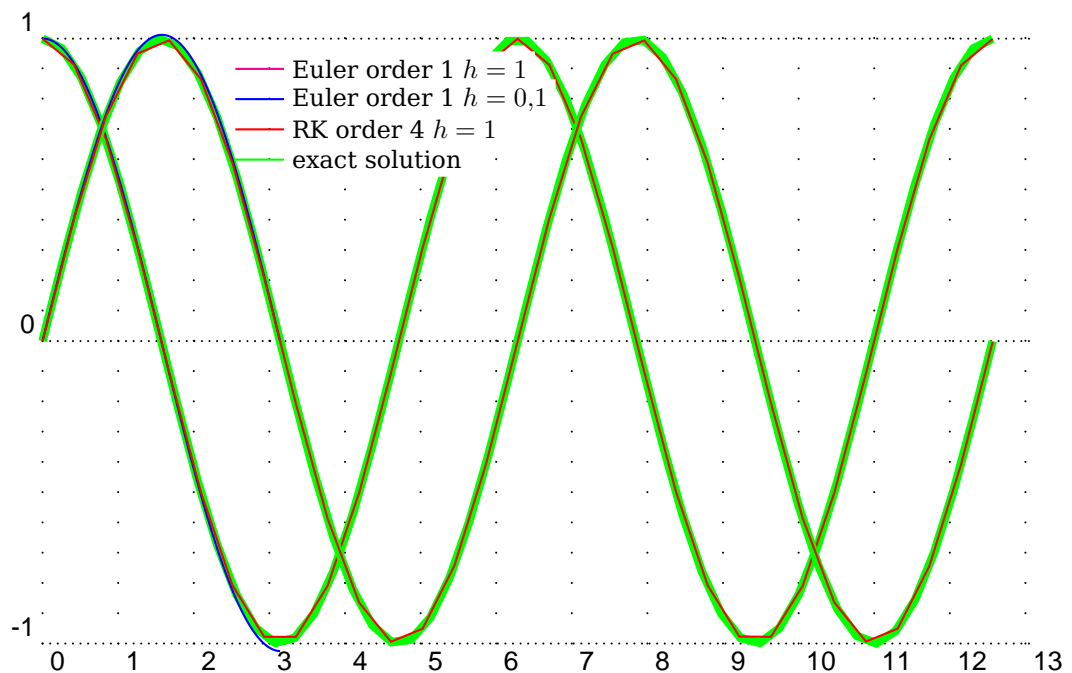


```

1 \def\Funct{exch} \psset{xunit=5cm, yunit=0.75cm}
2 \begin{pspicture}(0,-0.25)(2,7)\psgrid[subgriddiv=1,griddots=10]
3 \psplot[linecolor=green]{0}{2}{Euler x exp}%e^x
4 \psplotDiffEqn[linecolor=magenta,plotpoints=11]{0}{2}{1 1}{\Funct}
5 \psplotDiffEqn[linecolor=blue,plotpoints=101]{0}{2}{1 1}{\Funct}
6 \psplotDiffEqn[linecolor=red,method=rk4,plotpoints=11]{0}{2}{1 1}{\Funct}
7 \psplot[linecolor=green]{0}{2}{Euler dup x exp}%ch(x)
8   exch x neg exp add 2 div}
9 \psplotDiffEqn[linecolor=magenta,plotpoints=11]{0}{2}{1 0}{\Funct}
10 \psplotDiffEqn[linecolor=blue,plotpoints=101]{0}{2}{1 0}{\Funct}
11 \psplotDiffEqn[linecolor=red,method=rk4,plotpoints=11]{0}{2}{1 0}{\Funct}
12 \psplot[linecolor=green]{0}{2}{Euler dup x exp
13   exch x neg exp sub 2 div}%sh(x)
14 \psplotDiffEqn[linecolor=magenta,plotpoints=11]{0}{2}{0 1}{\Funct}
15 \psplotDiffEqn[linecolor=blue,plotpoints=101]{0}{2}{0 1}{\Funct}
16 \psplotDiffEqn[linecolor=red,method=rk4,plotpoints=11]{0}{2}{0 1}{\Funct}
17 \rput*(1.3,.9){\psline[linecolor=magenta](-.75cm,0)}\rput*[l](1.3,.9){\small\textsc{Euler}
18   order 1 $h=1$}
19 \rput*(1.3,.8){\psline[linecolor=blue](-.75cm,0)}\rput*[l](1.3,.8){\small\textsc{Euler}
20   order 1 $h=0{,}1$}
21 \rput*(1.3,.7){\psline[linecolor=red](-.75cm,0)}\rput*[l](1.3,.7){\small RK order 4 $h=1$}
22 \rput*(1.3,.6){\psline[linecolor=green](-.75cm,0)}\rput*[l](1.3,.6){\small exact
23   solution}
24 \end{pspicture}

```


$$y'' = -y$$



```

1 \def\Funct{exch neg}
2 \psset{xunit=1, yunit=4}
3 \def\quatrepi{12.5663706144}%%4pi=12.5663706144
4 \begin{pspicture}(0, -1.25)(\quatrepi,1.25)\psgrid[subgriddiv=1,griddots=10]
5 \psplot[linewidth=4\pslinewidth,linecolor=green]{0}{\quatrepi}{x RadtoDeg cos}
   %%cos(x)
6 \psplotDiffEqn[linecolor=blue, plotpoints=201]{0}{3.1415926}{1 0}{\Funct}
7 \psplotDiffEqn[linecolor=red, method=rk4, plotpoints=31]{0}{\quatrepi}{1 0}{\Funct}
8 \psplot[linewidth=4\pslinewidth,linecolor=green]{0}{\quatrepi}{x RadtoDeg sin}
   %%sin(x)
9 \psplotDiffEqn[linecolor=blue,plotpoints=201]{0}{3.1415926}{0 1}{\Funct}
10 \psplotDiffEqn[linecolor=red,method=rk4, plotpoints=31]{0}{\quatrepi}{0 1}{\Funct}
11 \rput*(3.3,.9){\psline[linecolor=magenta]{-.75cm,0}}\rput*[l](3.3,.9){\small
   Euler order 1 $h=1$}
12 \rput*(3.3,.8){\psline[linecolor=blue]{-.75cm,0}}\rput*[l](3.3,.8){\small
   Euler order 1 $h=0{,}1$}
13 \rput*(3.3,.7){\psline[linecolor=red]{-.75cm,0}}\rput*[l](3.3,.7){\small RK
   order 4 $h=1$}
14 \rput*(3.3,.6){\psline[linecolor=green]{-.75cm,0}}\rput*[l](3.3,.6){\small
   exact solution}
15 \end{pspicture}

```

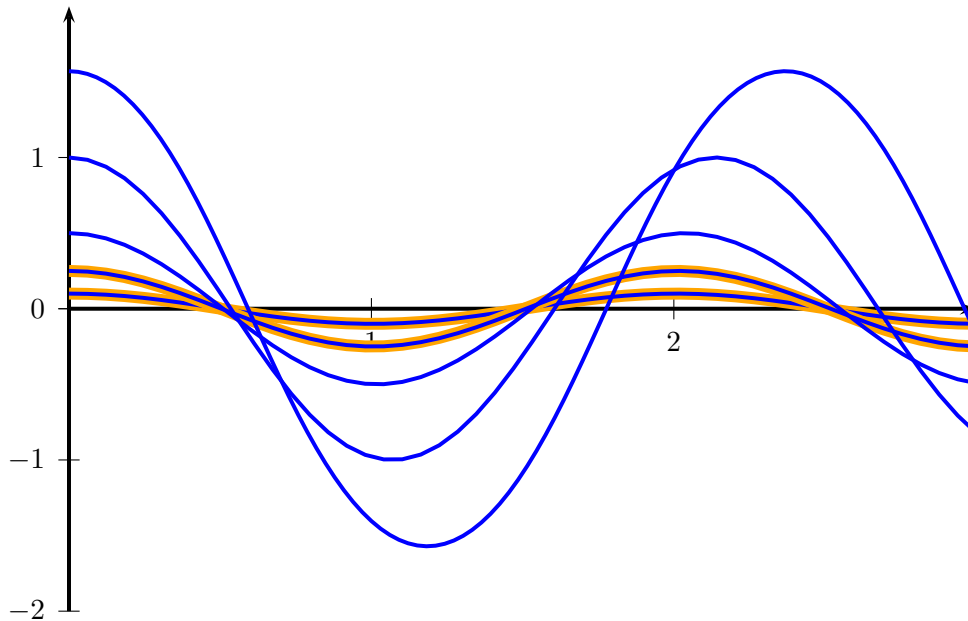
The mechanical pendulum: $y'' = -\frac{g}{l} \sin(y)$

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

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

The function f is written in PostScript code:

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



```

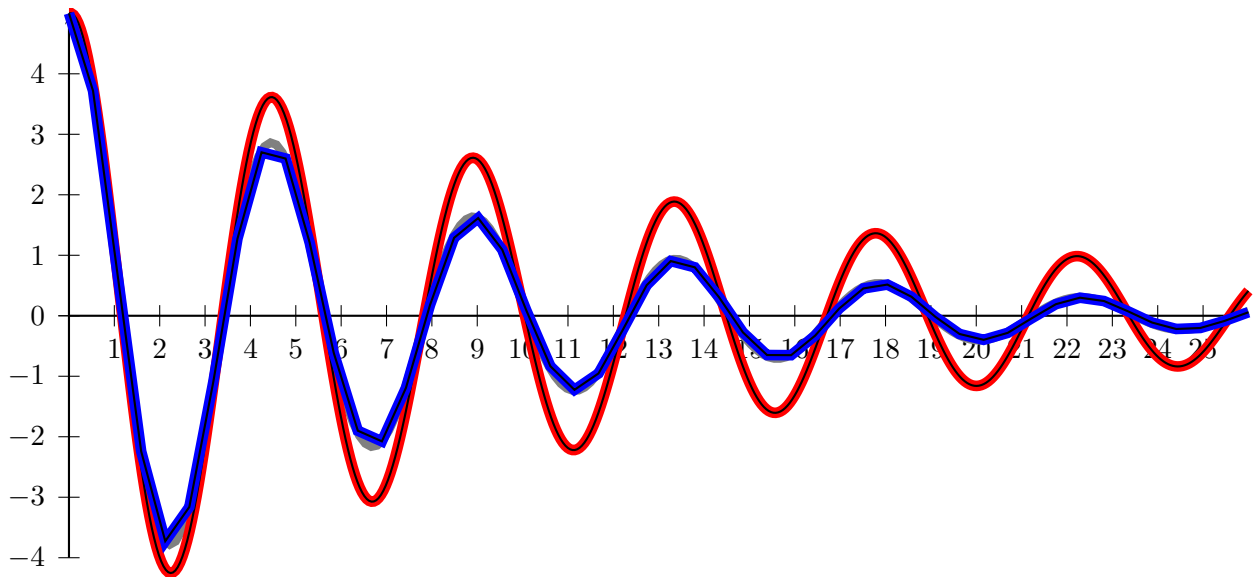
1 \def\Func{y[1]|-9.8*sin(y[0])}
2 \psset{yunit=2,xunit=4,algebraic=true,linewidth=1.5pt}
3 \begin{pspicture}(0,-2.25)(3,2.25)
4   \psaxes{->}(0,0)(0,-2)(3,2)
5   \psplot[linewidth=3\pslinewidth, linecolor=Orange]{0}{3}{.1*cos(sqrt(9.8)*x)}
6   \psset{method=rk4,plotpoints=50,linecolor=blue}
7   \psplotDiffEqn{0}{3}{.1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

```

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

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

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



```

1 \psset{xunit=.6,yunit=0.8,plotpoints=500}
2 \begin{pspicture}(0,-4.25)(26,5.25)
3   \psaxes{->}(0,0)(0,-4)(26,5)
4   \psplot[plotpoints=200,linewidth=4\pslinewidth,linecolor=gray]{0}{26}{%
5     Euler x -8 div exp x 127 sqrt 8 div mul RadtoDeg dup cos 5 mul exch sin 127
6     sqrt div 5 mul add mul}
7   \psplotDiffEqn[linecolor=red,linewidth=5\pslinewidth]{0}{26}{5 0}
8     {dup 3 1 roll -4 div exch 2 mul sub}
9   \psplotDiffEqn[linecolor=black,algebraic=true]{0}{26}{5 0} {y[1]|-y[1]/4-2*y
10    [0]}
11   \psset{method=rk4, plotpoints=50}
12   \psplotDiffEqn[linecolor=blue,linewidth=5\pslinewidth]{0}{26}{5 0}{%
13     dup 3 1 roll -4 div exch 2 mul sub}
14   \psplotDiffEqn[linecolor=black,algebraic=true]{0}{26}{5 0}{y[1]|-y[1]/4-2*y
15     [0]}
16 \end{pspicture}

```

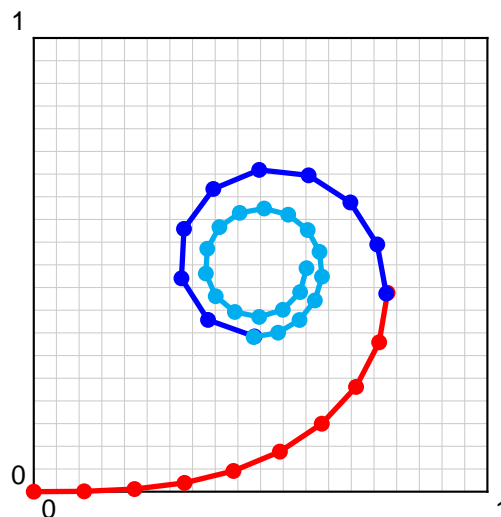
22.3. Save final state of a equation

With the macros `\BeginSaveFinalState` and `\EndSaveFinalState` the end values of a differential equation can be saved and then used with the optional argument `GetFinalState` as starting values for another equation.

```

1 \psset{unit=10cm,linewidth=2pt}
2 \begin{pspicture}(1,1)\psgrid[subgridcolor=black!20,subgriddiv=20]
3 \BeginSaveFinalState
4 \psplotDiffEqn[
5   whichabs=0,whichord=1,linecolor=red,method=rk4,
6   plotpoints=10,showpoints=true]{0}{1}{0 0}{
7   pop pop
8   x dup mul 2 div 180 mul cos %% dx/dt
9   x dup mul 2 div 180 mul sin %% dy/dt
10  }
11 \psplotDiffEqn[GetFinalState,
12   whichabs=0,whichord=1,linecolor=blue,method=rk4,%SaveFinalState,
13   plotpoints=10,showpoints=true]{1}{2}{0 0}{
14   pop pop
15   x dup mul 2 div 180 mul cos %% dx/dt
16   x dup mul 2 div 180 mul sin %% dy/dt
17  }
18 \psplotDiffEqn[GetFinalState,
19   whichabs=0,whichord=1,linecolor=cyan,method=rk4,%SaveFinalState,
20   plotpoints=19,showpoints=true]{2}{3}{0 0 }{
21   pop pop
22   x dup mul 2 div 180 mul cos %% dx/dt
23   x dup mul 2 div 180 mul sin %% dy/dt
24  }
25 \EndSaveFinalState
26 \end{pspicture}

```



23. \psMatrixPlot

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

```

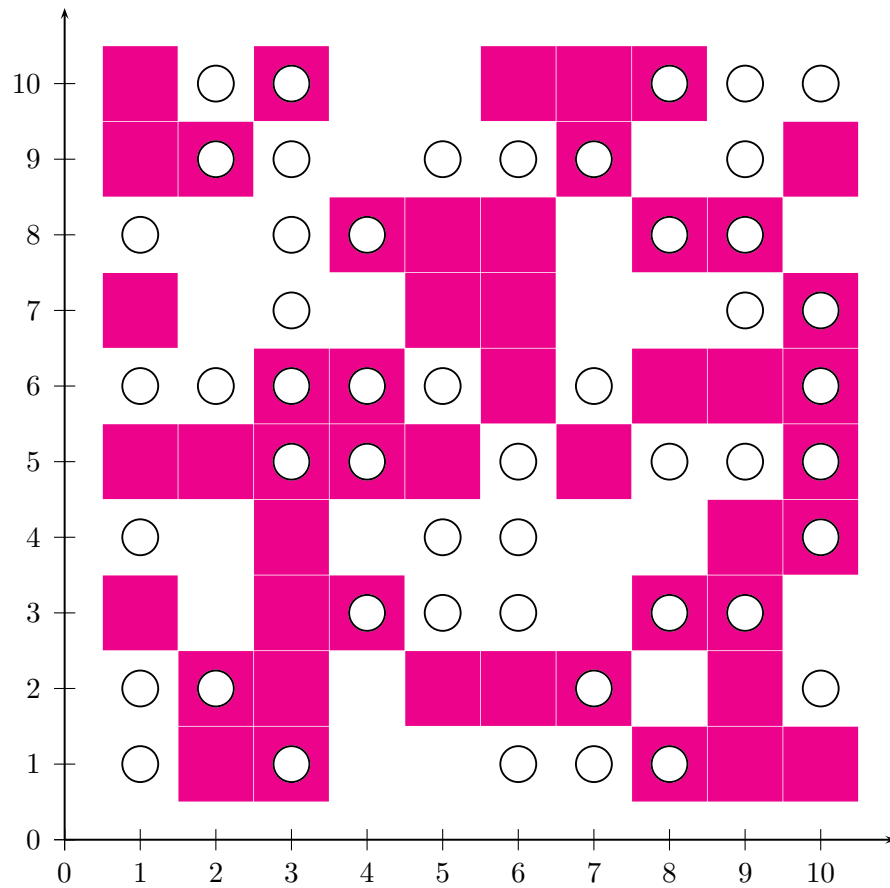
/dotmatrix [ % <----- important line
0 1 1 0 0 0 0 1 1 1
0 1 1 0 1 1 1 0 1 0
1 0 1 1 0 0 0 1 1 0
0 0 1 0 0 0 0 0 1 1
1 1 1 1 1 0 1 0 0 1
0 0 1 1 0 1 0 1 1 1
1 0 0 0 1 1 0 0 0 1
0 0 0 1 1 1 0 1 1 0
1 1 0 0 0 0 1 0 0 1
1 0 1 0 0 1 1 1 0 0
] def      % <----- important line

```

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

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

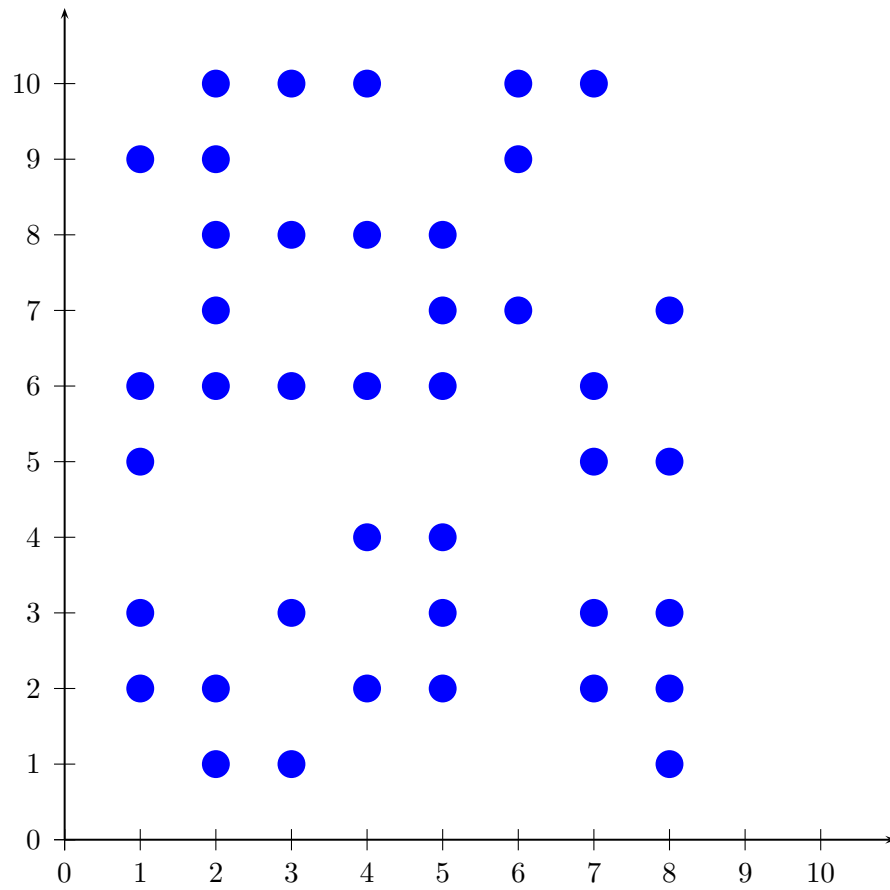
The matrix is scanned line by line from the the first one to the last. In general it appears as a bottom-to-top version of the above listed matrix, the first row 0110000111 is the first plotted line ($y = 1$). With the option ChangeOrder=true it looks exactly like the above view.



```

1 \begin{pspicture}(-0.5,-0.75)(11,11)
2   \psaxes{->}(11,11)
3   \psMatrixPlot[dotsize=1.1cm, dotstyle=square*, linecolor=magenta]%
4     {10}{10}{matrix.data}
5   \psMatrixPlot[dotsize=.5cm, dotstyle=o, ChangeOrder]{10}{10}{matrix.data}
6 \end{pspicture}

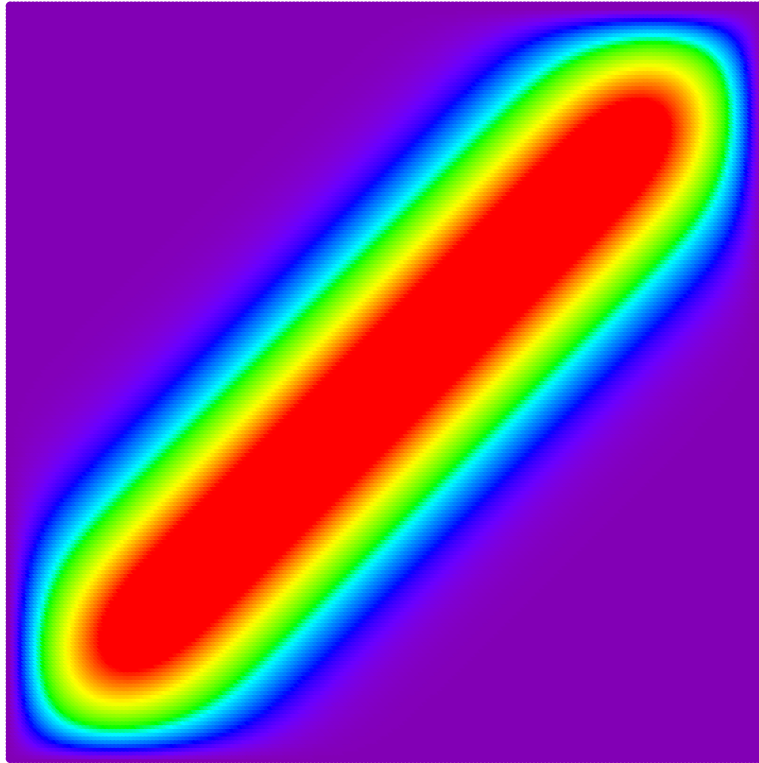
```



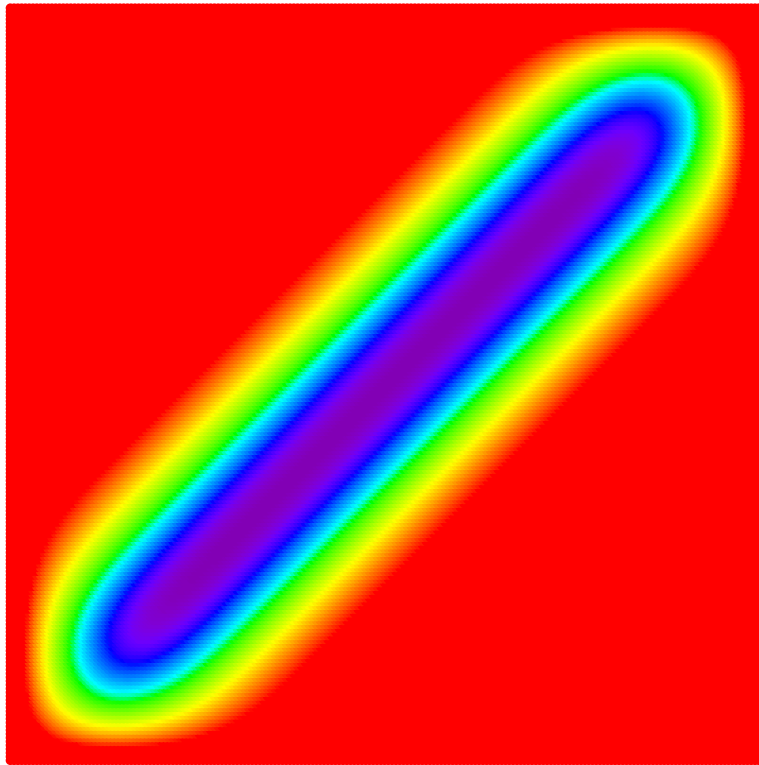
```
1 \begin{pspicture}(-0.5,-0.75)(11,11)
2   \psaxes{->}(11,11)
3   \psMatrixPlot[dotstyle=*,linecolor=blue]{10}{8}{matrix.data}
4 \end{pspicture}
```

With the `colorType=1` the data is printed as continuous color in the range of the wavelength. The smallest value of the data array is set to red and the biggest value is set to violet. All other values are substituted by the corresponding color of the wavelength. `colorType=2` ist the same, but vice versa with the color, from violet to red. `colorType=3` is the grayscale image and `colorType=4` the same invers.

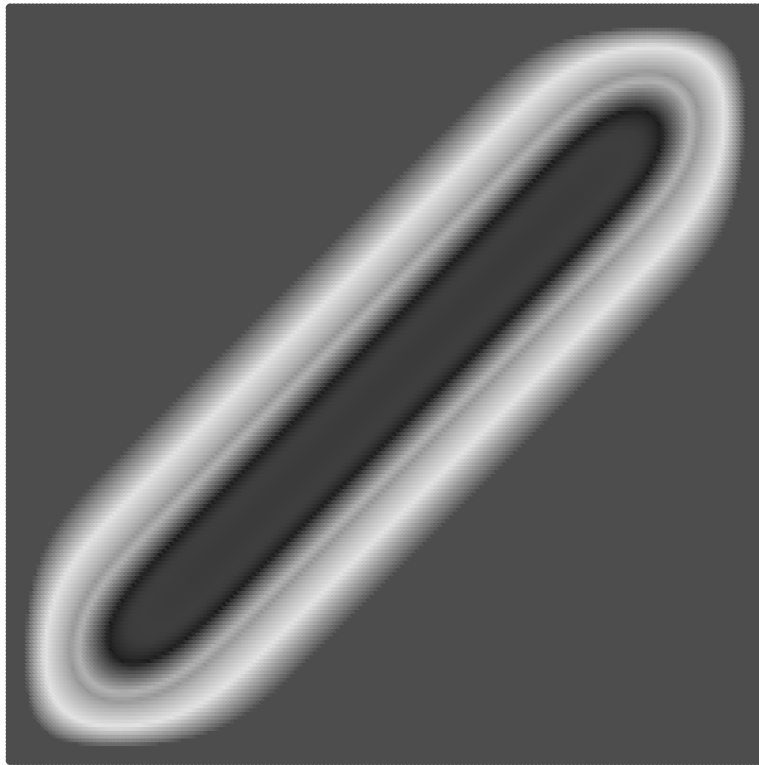
The following examples use a 200×200 matrix data, which is saved as `/dotmatrix [...]` in the file `pstricks-add-doc.dat`.



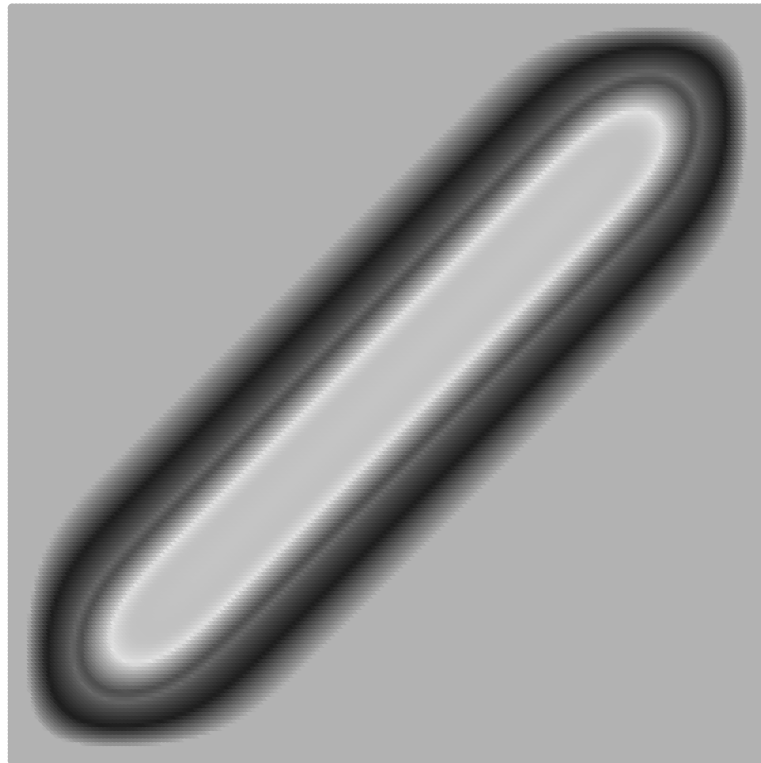
```
1 \begin{pspicture}(10,10)
2   \psMatrixPlot[colorType=1,xStep=0.05,yStep=0.05]{200}{200}{dotmatrix.data}
3 \end{pspicture}
```

```
1 \begin{pspicture}(10,10)  
2   \psMatrixPlot[colorType=2,xStep=0.05,yStep=0.05]{200}{200}{dotmatrix.data}  
3 \end{pspicture}
```



```
1 \begin{pspicture}(10,10)  
2   \psMatrixPlot[colorType=3,xStep=0.05,yStep=0.05]{200}{200}{dotmatrix.data}  
3 \end{pspicture}
```

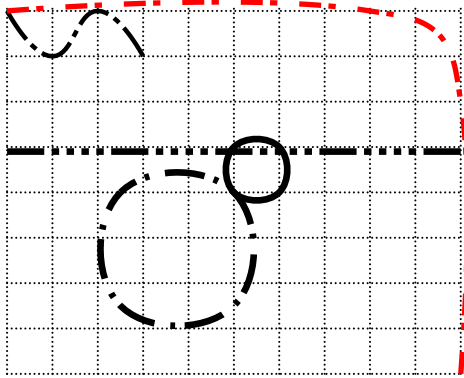


```
1 \begin{pspicture}(10,10)
2   \psMatrixPlot[colorType=4,xStep=0.05,yStep=0.05]{200}{200}{dotmatrix.data}
3 \end{pspicture}
```

24. Dashed Lines

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

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



```

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

```

25. Arrows

25.1. Definition

psstricks-add defines the following "arrows":

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

You can also mix and match, e.g., ->, *-) and [-> are all valid values of the arrows parameter. The parameter can be set with

```
\psset{arrows=<type>}
```

or for some macros with a special option, like

```
\psline[<general options>]{<arrow type>}(A)(B)
```

```
\psline[linecolor=red,linewidth=2pt]{|->}(0,0)(0,2)
```



25.2. Multiple arrows

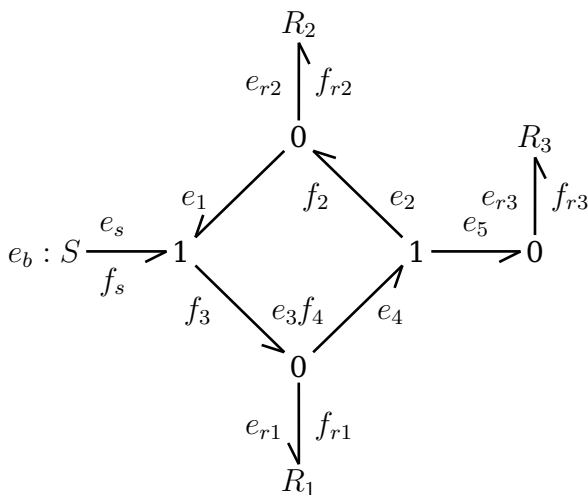
There are two new options which are only valid for the arrow type << or >>. nArrow sets both, the nArrowA and the nArrowB parameter. The meaning is declared in the following

tables. Without setting one of these parameters the behaviour is like the one described in the old PSTricks manual.

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

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

25.3. hookarrow



```
\psset{arrowsize=8pt,arrowlength=1,linewidth=1pt,nodesep=2pt,shortput=tablr}
```

```

2 \large
3 \begin{psmatrix}[colsep=12mm,rowsep=10mm]
4   & & $R_2$ & & \\
5   & & 0 & & $R_3$ \\
6 $e_b:S$ & 1 & & 1 & 0 \\
7   & & 0 & & \\
8   & & $R_1$ & & \\
9 \end{psmatrix}
10 \ncline{-h}{1,3}{2,3}<{$e_{r2}$}>{$f_{r2}$}\ncline{-h}{2,3}{3,2}<{$e_1$}
11 \ncline{-h}{3,1}{3,2}^{$e_s$}_{$f_s$} \ncline{-h}{3,2}{4,3}>{$e_3$}<{$f_3$}
12 \ncline{-h}{4,3}{3,4}>{$e_4$}<{$f_4$} \ncline{-h}{3,4}{2,3}>{$e_2$}<{$f_2$}
13 \ncline{-h}{3,4}{3,5}^{$e_5$}
14 \ncline{-h}{3,5}{2,5}<{$e_{r3}$}>{$f_{r3}$}
15 \ncline{-h}{4,3}{5,3}<{$e_{r1}$}>{$f_{r1}$}

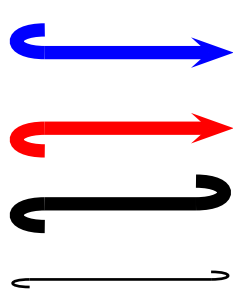
```

25.4. hookrightarrow and hookleftarrow

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

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

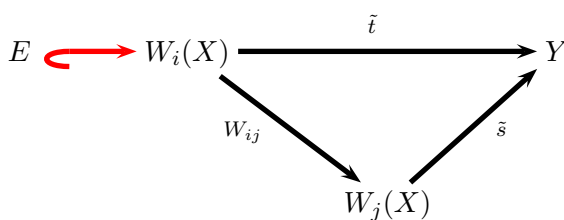
If the line begins with a right hook then the line ends with a left hook and vice versa:



```

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

```



```

1 $\begin{psmatrix}
2 E&W_i(X)&&Y \\
3 &&W_j(X) & \\
4 \psset{arrows=->,nodesep=3pt,linewidth
   =2pt}
5 \everypsbox{\scriptstyle}
6 \ncline[linecolor=red,arrows=H->,%
   hooklength=4mm,hookwidth=2mm
   ]{1,1}{1,2}
7 \ncline{1,2}{1,4}^{\tilde{t}}
8 \ncline{1,2}{2,3}<{W_{ij}}
9 \ncline{2,3}{1,4}>{\tilde{s}}
10 \end{psmatrix}$
11

```

25.5. ArrowInside Option

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

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

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

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

25.6. ArrowFill Option

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

are overpainted by the line.



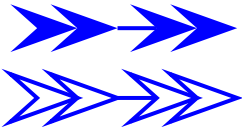
```
1 \psset{arrowscale=2.5}
2 \psline[linecolor=red,arrowinset=0]{<->}(-1,0)(2,0)
```



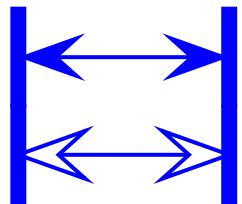
```
1 \psset{arrowscale=2.5}
2 \psline[linecolor=red,arrowinset=0,ArrowFill=false]
  <->}(-1,0)(2,0)
```



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



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



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

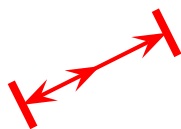
```
1 \psline[linecolor=blue,arrowscale=4,
  ArrowFill]{>|->|}(-1,0)(2,0)
```

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

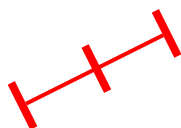
25.7. Examples

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

`\psline`




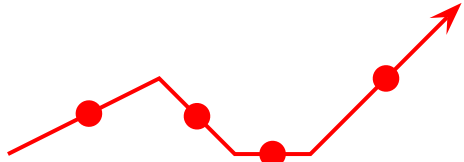
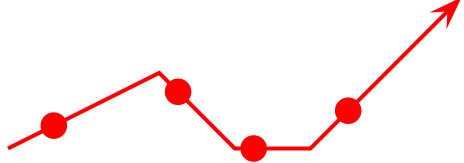
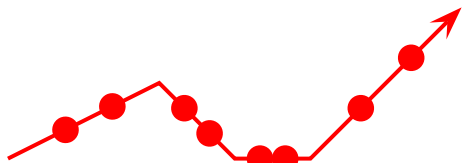

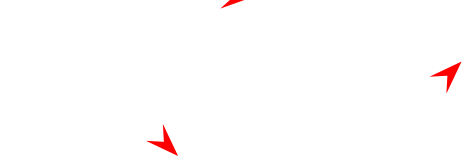

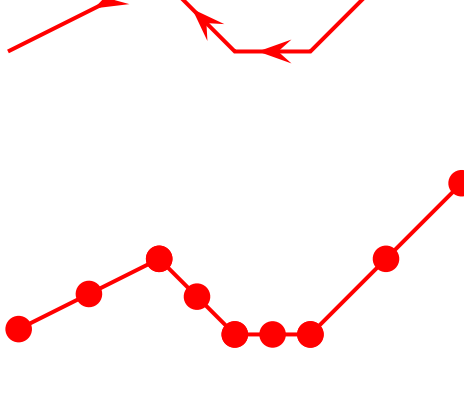
```
1 \begin{pspicture}(2,2)
2 \psset{arrowscale=2,ArrowFill=true}
3 \psline[ArrowInside=->]{|<->|}(2,1)
4 \end{pspicture}
```

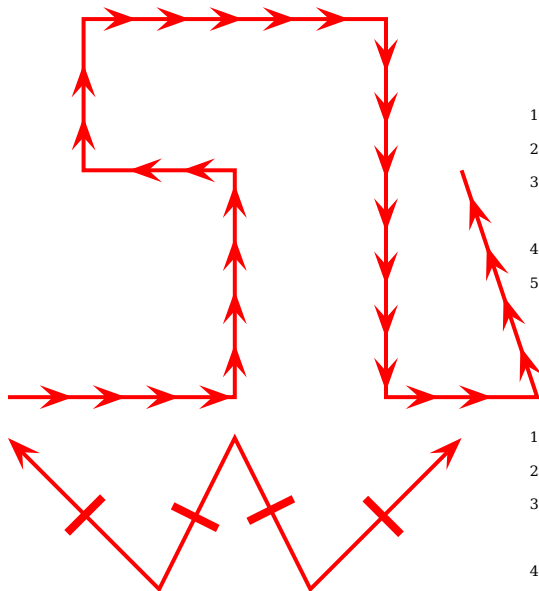


```
1 \begin{pspicture}(2,2)
2 \psset{arrowscale=2,ArrowFill=true}
3 \psline[ArrowInside=-|]{|-|}(2,1)
4 \end{pspicture}
```



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

	<pre> 1 \begin{pspicture}(2,2) 2 \psset{arrowscale=2,ArrowFill=true} 3 \psline[ArrowInside=->,ArrowInsideNo=2,ArrowInsideOffset =0.1]{->}(2,1) 4 \end{pspicture} </pre>
	<pre> 1 \begin{pspicture}(6,2) 2 \psset{arrowscale=2,ArrowFill=true} 3 \psline[ArrowInside=-*]{->}(0,0)(2,1)(3,0) (4,0)(6,2) 4 \end{pspicture} </pre>
	<pre> 1 \begin{pspicture}(6,2) 2 \psset{arrowscale=2,ArrowFill=true} 3 \psline[ArrowInside=-*,ArrowInsidePos =0.25]{->}(0,0)(2,1)(3,0)(4,0)(6,2) 4 \end{pspicture} </pre>
	<pre> 1 \begin{pspicture}(6,2) 2 \psset{arrowscale=2,ArrowFill=true} 3 \psline[ArrowInside=-*,ArrowInsidePos=0.25, ArrowInsideNo=2]{->}% 4 (0,0)(2,1)(3,0)(4,0)(6,2) 5 \end{pspicture} </pre>
	<pre> 1 \begin{pspicture}(6,2) 2 \psset{arrowscale=2,ArrowFill=true} 3 \psline[ArrowInside=->, ArrowInsidePos =0.25]{->}% 4 (0,0)(2,1)(3,0)(4,0)(6,2) 5 \end{pspicture} </pre>
	<pre> 1 \begin{pspicture}(6,2) 2 \psset{arrowscale=2,ArrowFill=true} 3 \psline[linestyle=none,ArrowInside=->, ArrowInsidePos=0.25]{->}% 4 (0,0)(2,1)(3,0)(4,0)(6,2) 5 \end{pspicture} </pre>
	<pre> 1 \begin{pspicture}(6,2) 2 \psset{arrowscale=2,ArrowFill=true} 3 \psline[ArrowInside=-<, ArrowInsidePos =0.75]{->}% 4 (0,0)(2,1)(3,0)(4,0)(6,2) 5 \end{pspicture} </pre>
	<pre> 1 \begin{pspicture}(6,2) 2 \psset{arrowscale=2,ArrowFill=true, ArrowInside=-*} 3 \psline(0,0)(2,1)(3,0)(4,0)(6,2) 4 \psset{linestyle=none} 5 \psline[ArrowInsidePos=0](0,0)(2,1)(3,0) (4,0)(6,2) 6 \psline[ArrowInsidePos=1](0,0)(2,1)(3,0) (4,0)(6,2) 7 \end{pspicture} </pre>



```

1 \begin{pspicture}(6,5)
2 \psset{arrowscale=2,ArrowFill=true}
3 \psline[ArrowInside=->,ArrowInsidePos
4   =20](0,0)(3,0)%
5   (3,3)(1,3)(1,5)(5,5)(5,0)(7,0)(6,3)
6 \end{pspicture}

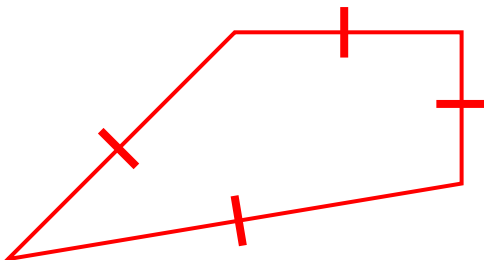
```

```

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

```

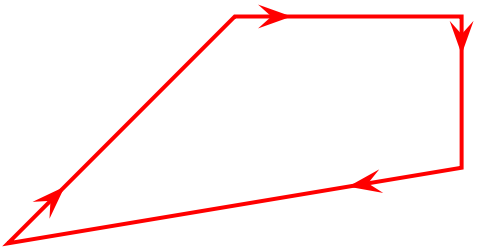
\pspolygon



```

1 \begin{pspicture}(6,3)
2 \psset{arrowscale=2}
3 \pspolygon[ArrowInside=-|](0,0)(3,3)(6,3)
4   (6,1)
5 \end{pspicture}

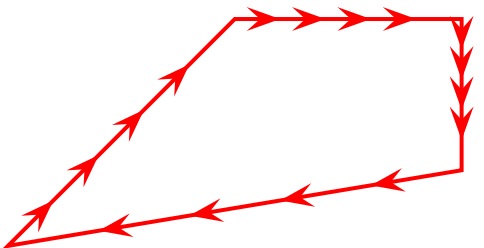
```



```

1 \begin{pspicture}(6,3)
2 \psset{arrowscale=2}
3 \pspolygon[ArrowInside=->,ArrowInsidePos
4   =0.25](0,0)(3,3)(6,3)(6,1)
5 \end{pspicture}

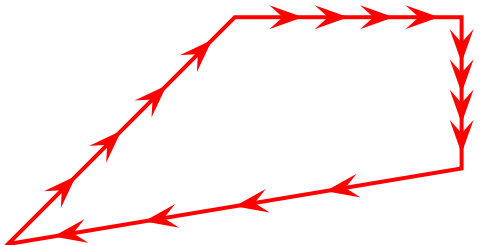
```



```

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

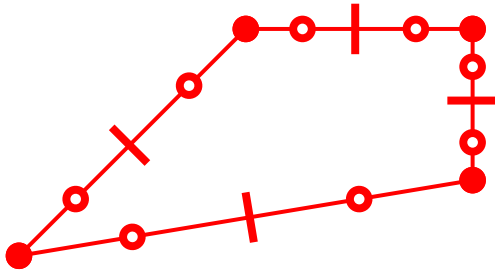
```



```

1 \begin{pspicture}(6,3)
2 \psset{arrowscale=2}
3 \pspolygon[ArrowInside=->,ArrowInsideNo=4,%
4   ArrowInsideOffset=0.1](0,0)(3,3)(6,3)
5   (6,1)
6 \end{pspicture}

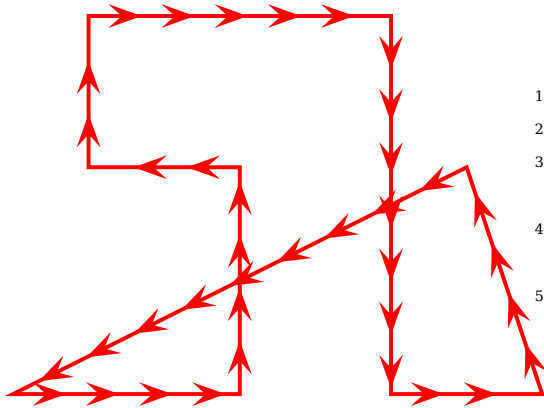
```



```

1 \begin{pspicture}(6,3)
2 \psset{arrowscale=2}
3 \pspolygon[ArrowInside=-|](0,0)(3,3)(6,3)
4   (6,1)
5 \psset{linestyle=none,ArrowInside=-*}
6 \pspolygon[ArrowInsidePos=0](0,0)(3,3)
7   (6,3)(6,1)
8 \pspolygon[ArrowInsidePos=1](0,0)(3,3)
9   (6,3)(6,1)
10 \psset{ArrowInside=-o}
11 \pspolygon[ArrowInsidePos=0.25](0,0)(3,3)
12   (6,3)(6,1)
13 \pspolygon[ArrowInsidePos=0.75](0,0)(3,3)
14   (6,3)(6,1)
15 \end{pspicture}

```

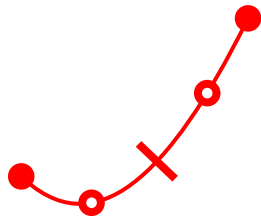


```

1 \begin{pspicture}(6,5)
2 \psset{arrowscale=2}
3 \pspolygon[ArrowInside=->,ArrowInsidePos
4   =20]%(0,0)(3,0)(3,3)(1,3)(1,5)(5,5)(5,0)(7,0)
5   (6,3)
6 \end{pspicture}

```

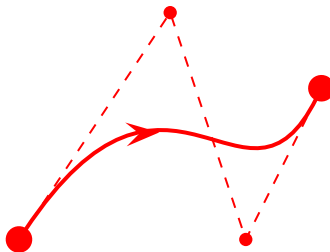
\psbezier



```

1 \begin{pspicture}(3,3)
2 \psset{arrowscale=2}
3 \psbezier[ArrowInside=-|](0,1)(1,0)(2,1)(3,3)
4 \psset{linestyle=none,ArrowInside=-o}
5 \psbezier[ArrowInsidePos=0.25](0,1)(1,0)(2,1)(3,3)
6 \psbezier[ArrowInsidePos=0.75](0,1)(1,0)(2,1)(3,3)
7 \psset{linestyle=none,ArrowInside=-*}
8 \psbezier[ArrowInsidePos=0](0,1)(1,0)(2,1)(3,3)
9 \psbezier[ArrowInsidePos=1](0,1)(1,0)(2,1)(3,3)
10 \end{pspicture}

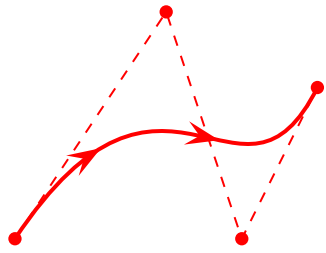
```



```

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

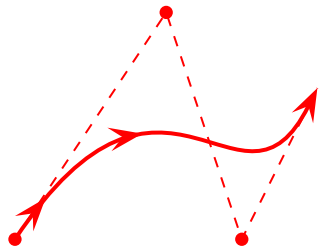
```



```

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

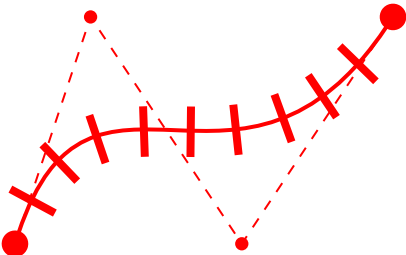
```



```

1 \begin{pspicture}(4,3)
2 \psset{arrowscale=2}
3 \psbezier[ArrowInside=->,showpoints=true,
4 ArrowInsideNo=2,ArrowInsideOffset=-0.2]
5 {->}(0,0)(2,3)(3,0)(4,2)
6 \end{pspicture}

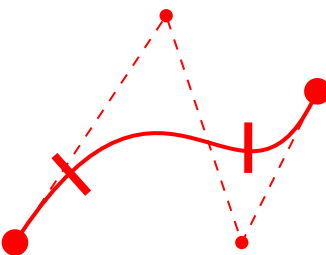
```



```

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

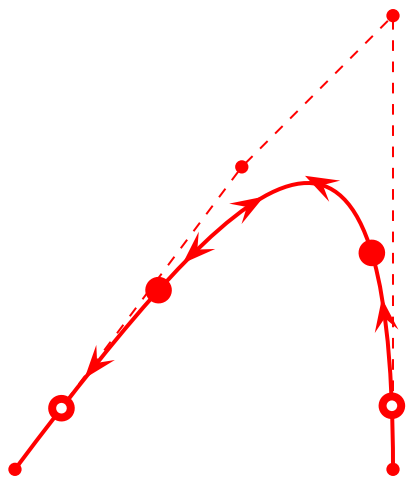
```



```

1 \begin{pspicture}(4,3)
2 \psset{arrowscale=2}
3 \psset{ArrowInside=-|}
4 \psbezier[ArrowInsidePos=0.25,showpoints=true
5 ]{*-}(2,3)(3,0)(4,2)
6 \psset{linestyle=none}
7 \psbezier[ArrowInsidePos=0.75](0,0)(2,3)(3,0)(4,2)
8 \end{pspicture}

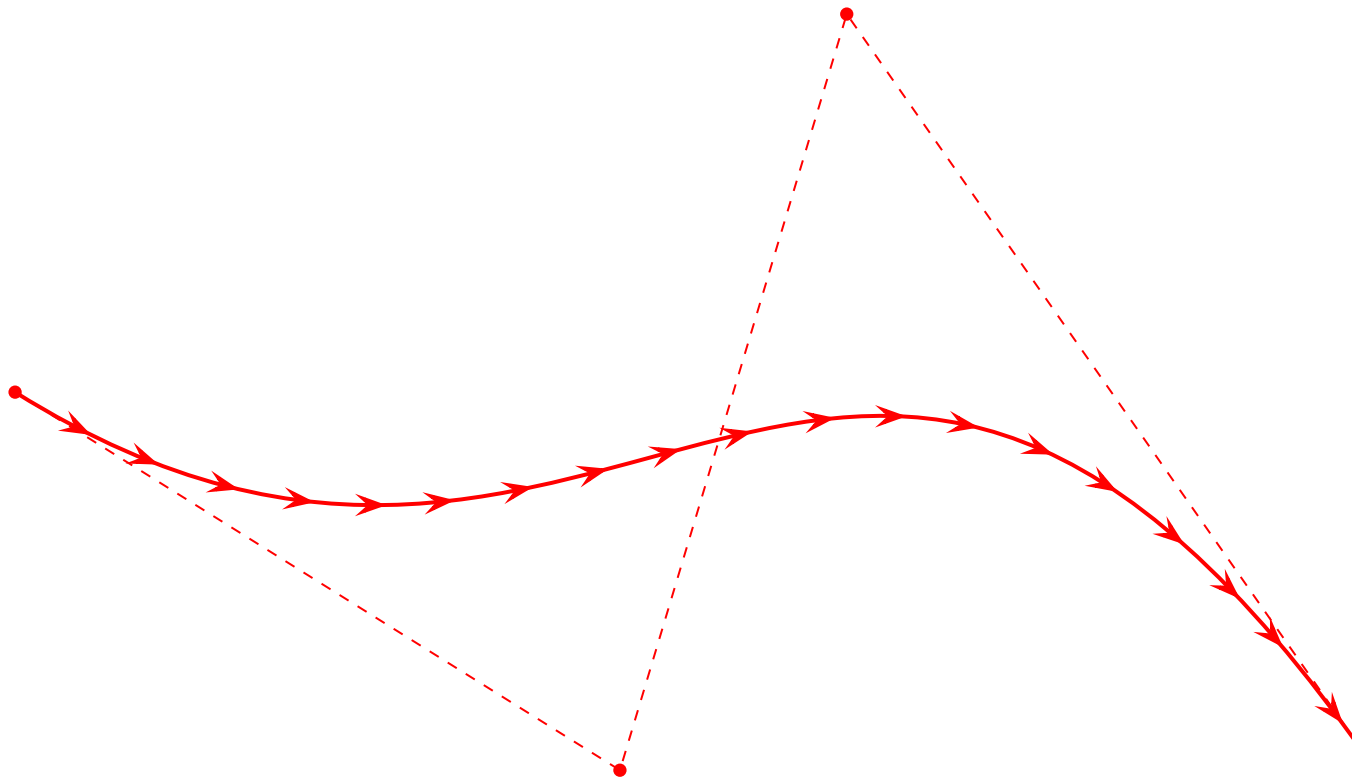
```



```

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

```



```

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

```

`\pcline`

These examples need the package `pst-node`.



```

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

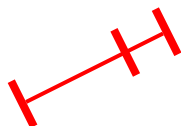
```



```

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

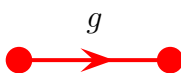
```



```

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

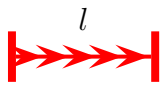
```



```

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

```



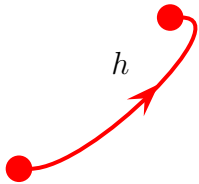
```

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

```

`\pccurve`

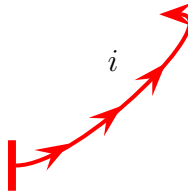
These examples also need the package `pst-node`.



```

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

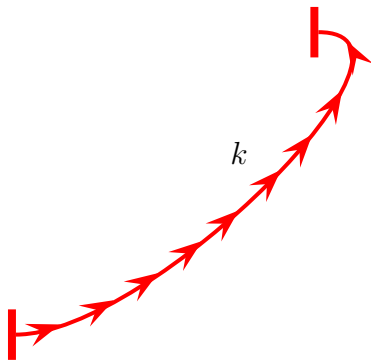
```



```

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

```



```

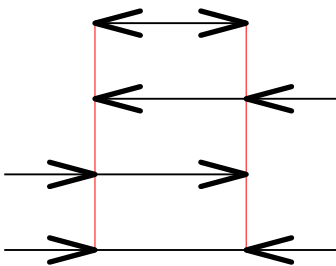
1 \begin{pspicture}(4,4)
2 \psset{arrowscale=2}
3 \pccurve[ArrowInside=->,ArrowInsidePos=20]{|-|}(0,0)
  (4,4)
4 \naput[labelsep=0.3]{\large{k$}}
5 \end{pspicture}

```

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

Possible optional arguments are

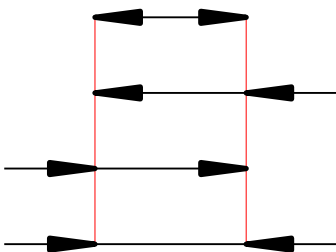
name	meaning
veearrowlength	default is 3mm
veearrowangle	default is 30
veearrowlinewidth	default is 0.35mm
filledveearrowlength	default is 3mm
filledveearrowangle	default is 15
filledveearrowlinewidth	default is 0.35mm
tickarrowlength	default is 1.5mm
tickarrowlinewidth	default is 0.35mm



```

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

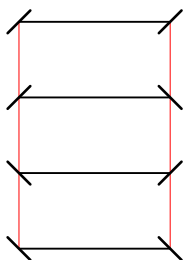
```



```

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

```



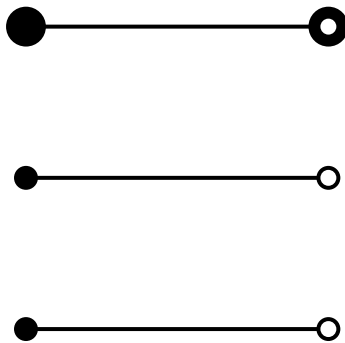
```

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

```


25.9. Special arrow option arrowLW

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



```

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

```

26. Transparent colors

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

27. „Manipulating transparent colors“

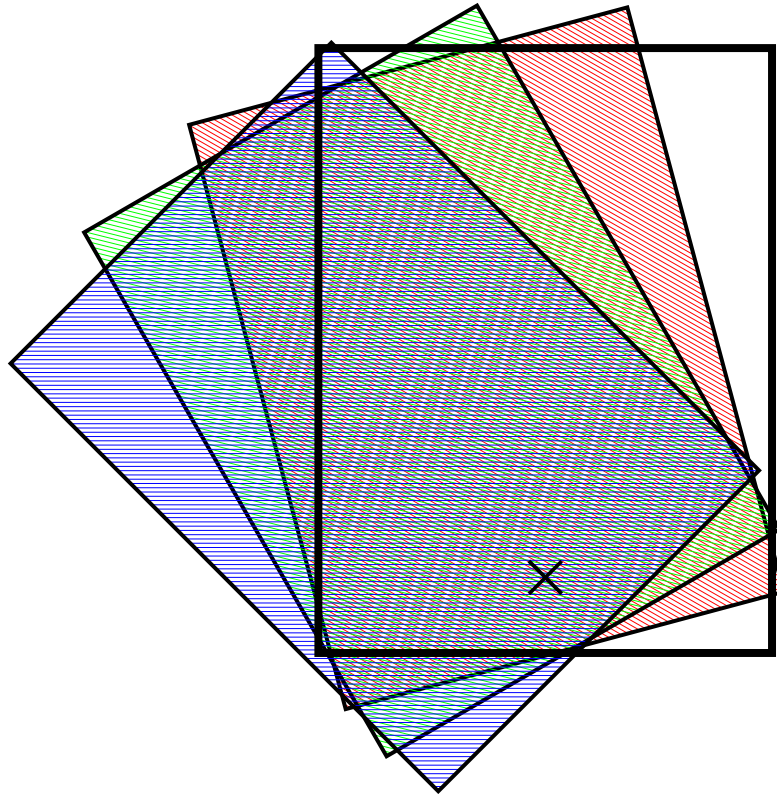
`psricks-add` supports real transparency and a simulated one with hatch lines:

```

1 \def\defineTColor{\@ifnextchar[{\defineTColor@i}{\defineTColor@i[]}}
2 \def\defineTColor@i[#1]#2#3{% transparency "Colors"
3   \newsstyle{#2}{%
4     fillstyle=vlines,hatchwidth=0.1\pslinewidth,
5     hatchsep=1\pslinewidth,hatchcolor=#3,#1%
6   }%
7 }
8 \defineTColor{TRed}{red}
9 \defineTColor{TGreen}{green}
10 \defineTColor{TBlue}{blue}

```

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



```

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

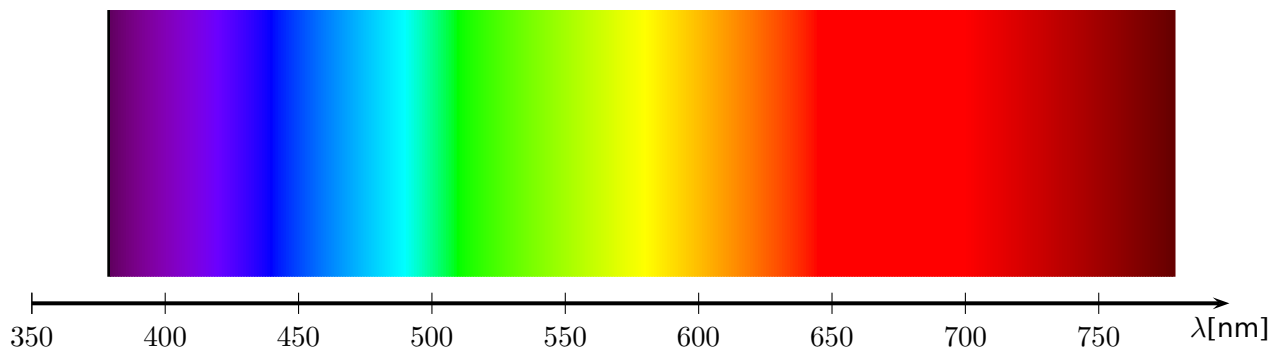
```

28. Calculated colors

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

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

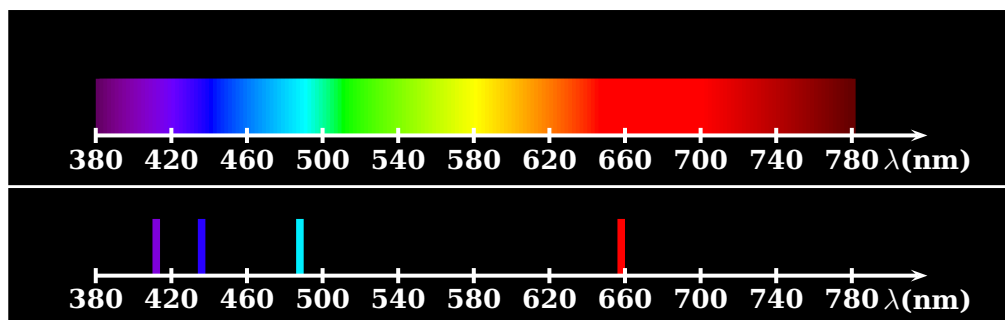
model can be one of the color models, which PostScript will understand, e.g. rgb. With this definition the color is calculated on the PostScript side.



```

1 \definecolor[ps]{bl}{rgb}{tx@addDict begin Red Green Blue end}%
2 \psset{unit=1bp}
3 \begin{pspicture}(0,-30)(400,100)
4 \multido{\iLAMBDA=0+1}{400}{%
5 \pstVerb{
6 \iLAMBDA\space 379 add dup /lambda exch def
7 tx@addDict begin wavelengthToRGB end
8 }%
9 \psline[linecolor=bl](\iLAMBDA,0)(\iLAMBDA,100)%
10 }
11 \psaxes[yAxis=false,0x=350,dx=50bp,Dx=50]{->}{-29,-10}(420,100)
12 \uput[-90](420,-10){$\lambda$[nm]}
13 \end{pspicture}

```



Spectrum of hydrogen emission (Manuel Luque)

```

1 \newcommand{\Touch}{%
2 \psframe[linestyle=none,fillstyle=solid,fillcolor=bl,dimen=middle](0.1,0.75)}
3 \definecolor[ps]{bl}{rgb}{tx@addDict begin Red Green Blue end}%
4 % Echelle 1cm <-> 40 nm
5 % 1 nm <-> 0.025 cm
6 \psframebox[fillstyle=solid,fillcolor=black]{%
7 \begin{pspicture}(-1,-0.5)(12,1.5)
8 \multido{\iLAMBDA=380+2}{200}{%
9 \pstVerb{
10 /lambda \iLAMBDA\space def
11 lambda
12 tx@addDict begin wavelengthToRGB end
13 }%
14 \rput(! lambda 0.025 mul 9.5 sub 0){\Touch}
15 }

```

```

16 \multido{\n=0+1,\iDiv=380+40}{11}{%
17   \psline[linecolor=white](\n,0.1)(\n,-0.1)
18   \uput[270](\n,0){\textbf{\white\iDiv}}
19   \psline[linecolor=white]{->}(11,0)
20   \uput[270](11,0){\textbf{\white$\lambda$(nm)}}
21 \end{pspicture}}
22
23 \psframebox[fillstyle=solid,fillcolor=black]{%
24 \begin{pspicture}(-1,-0.5)(12,1)
25   \pstVerb{
26     /lambda 656 def
27     lambda
28     tx@addDict begin wavelengthToRGB end
29   }%
30   \rput(! 656 0.025 mul 9.5 sub 0){\Touch}
31   \pstVerb{
32     /lambda 486 def
33     lambda
34     tx@addDict begin wavelengthToRGB end
35   }%
36   \rput(! 486 0.025 mul 9.5 sub 0){\Touch}
37   \pstVerb{
38     /lambda 434 def
39     lambda
40     tx@addDict begin wavelengthToRGB end
41   }%
42   \rput(! 434 0.025 mul 9.5 sub 0){\Touch}
43   \pstVerb{
44     /lambda 410 def
45     lambda
46     tx@addDict begin wavelengthToRGB end
47   }%
48   \rput(! 410 0.025 mul 9.5 sub 0){\Touch}
49 \multido{\n=0+1,\iDiv=380+40}{11}{%
50   \psline[linecolor=white](\n,0.1)(\n,-0.1)
51   \uput[270](\n,0){\textbf{\white\iDiv}}
52   \psline[linecolor=white]{->}(11,0)
53   \uput[270](11,0){\textbf{\white$\lambda$(nm)}}
54 \end{pspicture}}
55
56 Spectrum of hydrogen emission (Manuel Luque)

```

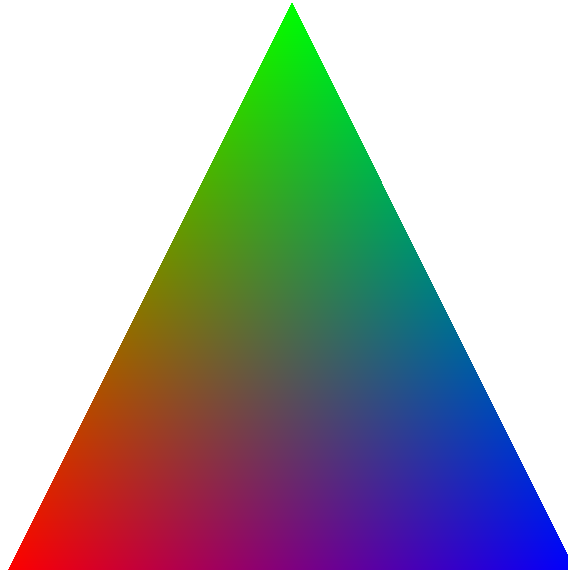
29. Gouraud shading

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

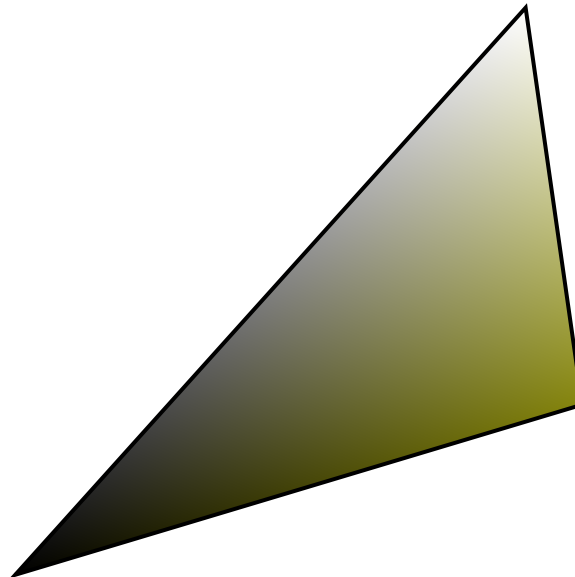
<http://www.wikipedia.org>

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

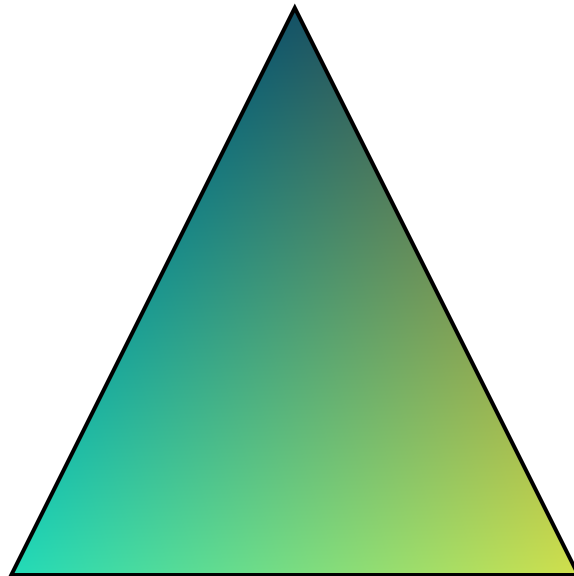
```
\psGTriangle(x1,y1)(x2,y2)(x3,y3){color1}{color2}{color3}
```



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



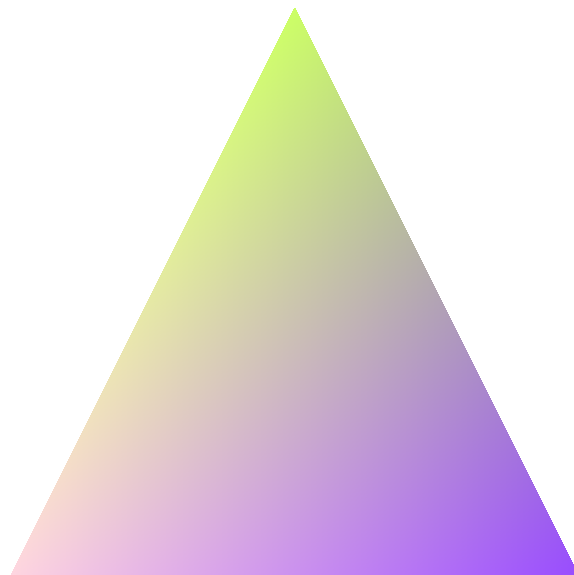
```
1 \begin{pspicture}(0,-.25)(10,10)
2   \psGTriangle*(0,0)(9,10)(10,3){black}{white!50}{red!50!green!95}
3 \end{pspicture}
```



```

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

```



```

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

```

A. \resetOptions

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

B. PostScript

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

Table 4: Some primitive PostScript macros

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

C. List of all optional arguments for pstricks-add

Key	Type	Default
CMYK	boolean	true
intSeparator	ordinary	[none]
braceWidth	ordinary	[none]
bracePos	ordinary	[none]
braceWidthInner	ordinary	[none]
braceWidthOuter	ordinary	[none]
veearrowlength	ordinary	3mm
veearrowangle	ordinary	30
veearrowlinewidth	ordinary	0.35mm
filledveearrowlength	ordinary	3mm
filledveearrowangle	ordinary	15
filledveearrowlinewidth	ordinary	0.35mm
tickarrowlength	ordinary	1.5mm
tickarrowlinewidth	ordinary	0.35mm
hooklength	ordinary	3mm
hookwidth	ordinary	1mm
ArrowFill	boolean	true
nArrowsA	ordinary	2
nArrowsB	ordinary	2
nArrows	ordinary	2
ArrowInside	ordinary	[none]
ArrowInsidePos	ordinary	0.5
ArrowInsideNo	ordinary	1
ArrowInsideOffset	ordinary	0
randomPoints	ordinary	1000
color	boolean	true
blName	command	
bcName	command	
brName	command	
clName	command	
ccName	command	
crName	command	
tlName	command	
tcName	command	
trName	command	
method	ordinary	[none]
whichabs	ordinary	[none]
whichord	ordinary	[none]
plotfuncx	ordinary	[none]
plotfuncy	ordinary	[none]
expression	ordinary	[none]

Continued on next page

Continued from previous page

Key	Type	Default
buildvector	boolean	true
varsteptol	ordinary	[none]
adamsorder	ordinary	[none]
StepType	ordinary	[none]
Derive	ordinary	[none]
Tnormal	boolean	true
GetFinalState	boolean	true
filename	ordinary	[none]
saveData	boolean	true
colorType	ordinary	0
chartStyle	ordinary	[none]
chartColor	ordinary	[none]
chartSep	ordinary	[none]
chartStack	ordinary	[none]
chartStackDepth	ordinary	[none]
chartStackWidth	ordinary	[none]
chartHeight	ordinary	[none]
userColor	ordinary	[none]
chartNodeI	ordinary	[none]
chartNode0	ordinary	[none]

References

- [1] Hendri Adriaens. xkeyval package. CTAN:/macros/latex/contrib/xkeyval, 2004.
- [2] Denis Girou. Présentation de PSTricks. *Cahier GUTenberg*, 16:21–70, April 1994.
- [3] Michel Goosens, Frank Mittelbach, Sebastian Rahtz, Denis Roegel, and Herbert Voß. *The L^AT_EX Graphics Companion*. Addison-Wesley Publishing Company, Reading, Mass., 2007.
- [4] Alan Hoenig. *T_EX Unbound: L^AT_EX & T_EX Strategies, Fonts, Graphics, and More*. Oxford University Press, London, 1998.
- [5] Laura E. Jackson and Herbert Voß. Die plot-funktionen von pst-plot. *Die T_EXnische Komödie*, 2/02:27–34, June 2002.
- [6] Nikolai G. Kollock. *PostScript richtig eingesetzt: vom Konzept zum praktischen Einsatz*. IWT, Vaterstetten, 1989.
- [7] Frank Mittelbach and Michel Goosens et al. *The L^AT_EX Companion*. Addison-Wesley Publishing Company, Boston, second edition, 2004.
- [8] Frank Mittelbach and Michel Goosens et al. *Der L^AT_EX Begleiter*. Pearson Education, München, zweite edition, 2005.

-
- [9] Herbert Voß. *Chaos und Fraktale selbst programmieren: von Mandelbrotmengen über Farbmanipulationen zur perfekten Darstellung*. Franzis Verlag, Poing, 1994.
- [10] Herbert Voß. Die mathematischen Funktionen von PostScript. *Die T_EXnische Komödie*, 1/02, March 2002.
- [11] Herbert Voß. *PSTricks Grafik für T_EX und L^AT_EX*. DANTE – Lob.media, Heidelberg/Hamburg, fifth edition, 2008.
- [12] Herbert Voß. *Mathematiksatz in L^AT_EX*. Lehmanns Media/DANTE, Berlin/Heidelberg, first edition, 2009.
- [13] Timothy Van Zandt. *PSTricks - PostScript macros for generic T_EX*. <http://www.tug.org/application/PSTricks>, 1993.
- [14] Timothy Van Zandt. *multido.tex - a loop macro, that supports fixed-point addition*. CTAN:/graphics/pstricks/generic/multido.tex, 1997.
- [15] Timothy Van Zandt. *pst-plot: Plotting two dimensional functions and data*. CTAN:/graphics/pstricks/generic/pst-plot.tex, 1999.
- [16] Timothy Van Zandt and Denis Girou. Inside PSTricks. *TUGboat*, 15:239–246, September 1994.

Index

Symbols

`\hAmacroB`, 5

`(-)`, 85

`)-(`, 85

`*`, 97

`**-**`, 85

`*-)`, 85

`*-*`, 85

`-`, 85

`->`, 85, 88

`-dCompatibilityLevel`, 97

`-<<`, 86

`->>`, 86

`<-`, 88

`<->`, 85

`<<->>`, 85

`>-<`, 85

`[->`, 85

`[-]`, 85

`]-[`, 85

`<<-`, 86

`<<-<<`, 86

`<<->>`, 86

`>>-`, 86

`>>-<<`, 85, 86

`>>->>`, 86

A

`abs`, 103

`add`, 103

`algebraic`, 33, 37

`algebraic=true`, 40, 42, 57, 65

`aload`, 103

`angle`, 24

`array`, 103

`ArrowFill`, 88

`ArrowInside`, 88

`ArrowInsideNo`, 88

`ArrowInsideOffset`, 88

`ArrowInsidePos`, 88

`arrowLW`, 97

`arrows`, 85

`arrowscale`, 97

`astore`, 103

`atan`, 103

B

`B`, 15

`b`, 15

`\BeginSaveFinalState`, 76

`brace`, 16

`bracePos`, 15

`braceWidth`, 15

`braceWidthInner`, 15

`braceWidthOuter`, 15

`buildvector`, 57

C

`C`, 15

`c`, 15

`\cancel`, 31

`cardioid`, 39

`ChangeOrder`, 77

`chartColor`, 11

`chartFillColor?`, 11

`chartNodeI`, 11

`chartNode0`, 11

`chartSep`, 11

`color`, 20

`colorType`, 80

`cos`, 103

`curvature`, 36

D

`dash`, 83

`default`, 37, 44

`Derive`, 37–39, 42

`div`, 103

`dotmatrix`, 77

`dup`, 103

E

`\EndSaveFinalState`, 76

`Environment`

– `pspicture`, 24

F

- false, 88
- File
 - pstricks-add-doc.dat, 80
- fillcolor, 15
- filledveearrowangle, 96
- filledveearrowlength, 96
- filledveearrowlinewidth, 96
- fillstyle
 - eofill, 2
- Fresnel, 69
- G**
- GetFinalState, 76
- Gouraud, 100
- H**
- H, 87
- hooklength, 87
- hookwidth, 87
- hydrogen, 99
- I**
- i, 33
- infimum, 33
- InsideArrow, 2
- intSeparator, 23
- K**
- Keyvalue
 - B, 15
 - b, 15
 - C, 15
 - c, 15
 - default, 44
 - i, 33
 - infimum, 33
 - l, 15
 - R, 33
 - r, 15
 - Riemann, 33
 - s, 33
 - supremum, 33
 - t, 15
 - u, 33
 - upper, 33
- Keyword
 - algebraic, 33, 37
 - algebraic=true, 40, 42, 57, 65
 - angle, 24
 - ArrowFill, 88
 - ArrowInside, 88
 - ArrowInsideNo, 88
 - ArrowInsideOffset, 88
 - ArrowInsidePos, 88
 - arrowLW, 97
 - arrows, 85
 - arrowscale, 97
 - bracePos, 15
 - braceWidth, 15
 - braceWidthInner, 15
 - braceWidthOuter, 15
 - buildvector, 57
 - ChangeOrder, 77
 - chartColor, 11
 - chartFillColor?, 11
 - chartNodeI, 11
 - chartNodeO, 11
 - chartSep, 11
 - color, 20
 - colorType, 80
 - curvature, 36
 - dash, 83
 - Derive, 37–39
 - fillcolor, 15
 - filledveearrowangle, 96
 - filledveearrowlength, 96
 - filledveearrowlinewidth, 96
 - GetFinalState, 76
 - hooklength, 87
 - hookwidth, 87
 - InsideArrow, 2
 - intSeparator, 23
 - labelFontSize, 2
 - lower, 33
 - method, 57, 58
 - nodesepA, 15
 - nodesepB, 15
 - opacity, 31
 - plotfuncx, 57
 - plotfuncy, 57
 - plotpoints, 44
 - randomPoints, 20

- ref, 15
- rot, 15
- StepType, 33
- strokeopacity, 31
- tickarrowlength, 96
- tickarrowlinewidth, 96
- ticksize, 2
- tickstyle, 2
- Tnormal, 36, 39
- trueAngle, 24, 26
- unit, 22
- userColor, 11
- varrkiv, 58
- VarStep, 44
- varStep, 47
- VarStepEpsilon, 6, 44
- varsteptol, 6, 58
- veearrowangle, 96
- veearrowlength, 96
- veearrowlinewidth, 96
- whichabs, 57
- whichord, 57
- xunit, 24
- xyLabel, 2
- yunit, 24

L

- l, 15, 33
- labelFontSize, 2
- Lissajou curve, 40
- ln, 103
- log, 103
- Lotka-Volterra, 70
- lower, 33

M

- Macro
 - $\backslash hAmacroB$, 5
 - $\backslash BeginSaveFinalState$, 76
 - $\backslash cancel$, 31
 - $\backslash EndSaveFinalState$, 76
 - $\backslash multirput$, 7
 - $\backslash ncline$, 10
 - $\backslash parametricplot$, 49
 - $\backslash psbezier$, 2
 - $\backslash psbrace*$, 15

- $\backslash psCancel*$, 31
- $\backslash psChart$, 11
- $\backslash psComment*$, 10
- $\backslash pscurve$, 36
- $\backslash psDefPSPNodes$, 24
- $\backslash psdice$, 22
- $\backslash psFormatInt$, 23
- $\backslash psGetDistance$, 5
- $\backslash psGetSlope$, 5
- $\backslash psHomothetie$, 14
- $\backslash pslineByHand$, 6
- $\backslash pslinewidth$, 15
- $\backslash psMatrixPlot$, 77
- $\backslash psplot$, 44
- $\backslash psplotDiffEqn$, 57
- $\backslash psplotTangent*$, 37
- $\backslash psplotTangent$, 36
- $\backslash psplotTangentLine$, 36
- $\backslash psRandom$, 20
- $\backslash psRelLine$, 25
- $\backslash psRelNode$, 24, 25
- $\backslash psrotate$, 8
- $\backslash psset$, 85, 87
- $\backslash psStep$, 33
- $\backslash psTangentLine$, 36
- $\backslash rmultiput*$, 7
- $\backslash rmultiput$, 7
- $\backslash rput$, 8, 11, 22
- $\backslash specialCoor$, 16
- matrix, 77
- method, 57, 58
- mul, 103
- $\backslash multirput$, 7

N

- $\backslash ncline$, 10
- neg, 103
- nodesepA, 15
- nodesepB, 15

O

- o, 97
- o-o, 85
- oo-oo, 85
- opacity, 31
- oscillation, 74

P

Package

- pst-func, 2
- pst-node, 103
- pst-plot, 2, 103
- pst-xkey, 2
- pstricks, 15, 103
- pstricks-add, 2, 23, 37, 85, 97
- xkeyval, 2

Package option

- -dCompatibilityLevel, 97

\parametricplot, 49

plotfuncx, 57

plotfuncy, 57

plotpoints, 44

pop, 103

PostScript, 103

- abs, 103
- add, 103
- aload, 103
- array, 103
- astore, 103
- atan, 103
- cos, 103
- Derive, 42
- div, 103
- dotmatrix, 77
- dup, 103
- ln, 103
- log, 103
- mul, 103
- neg, 103
- pop, 103
- sin, 103
- sqrt, 103
- sub, 103
- tan, 103

Program

- ps2pdf, 97

ps2pdf, 97

\psbezier, 2

\psbrace*, 15

\psCancel*, 31

\psChart, 11

\psComment*, 10

\pscurve, 36

\psDefPSPNodes, 24

\psdice, 22

\psFormatInt, 23

\psGetDistance, 5

\psGetSlope, 5

\psHomothetie, 14

\pslineByHand, 6

\pslinewidth, 15

\psMatrixPlot, 77

pspicture, 24

\psplot, 44

\psplotDiffEqn, 57

\psplotTangent, 36

\psplotTangent*, 37

\psplotTangentLine, 36

\psRandom, 20

\psRelLine, 25

\psRelNode, 24, 25

\psrotate, 8

\psset, 85, 87

\psStep, 33

pst-func, 2

pst-node, 103

pst-plot, 2, 103

pst-xkey, 2

\psTangentLine, 36

pstricks, 15, 103

pstricks-add, 2, 23, 37, 85, 97

pstricks-add-doc.dat, 80

R

R, 33

r, 15

randomPoints, 20

ref, 15

Riemann, 33

Riemann, 33

\rmultiput, 7

\rmultiput*, 7

rot, 15

\rput, 8, 11, 22

Runge-Kutta, 58

S

s, 33

sin, 103
\specialCoor, 16
Spectrum, 99
sqrt, 103
StepType, 33
strokeopacity, 31
sub, 103
supremum, 33
Syntax
- (-), 85
-)-(, 85
- *, 97
- **-**, 85
- *-), 85
- *-*, 85
- -, 85
- ->, 85, 88
- -<<, 86
- ->>, 86
- <-, 88
- <->, 85
- <<->>, 85
- >-<, 85
- [->, 85
- [-], 85
-]-[, 85
- <<- , 86
- <<-<<, 86
- <<->>, 86
- >>- , 86
- >>-<<, 85, 86
- >>->>, 86
- H, 87
- o, 97
- o-o, 85
- oo-oo, 85

T

t, 15
tan, 103
tickarrowlength, 96
tickarrowlinewidth, 96
ticksize, 2
tickstyle, 2
Tnormal, 36, 39
true, 33

trueAngle, 24, 26

U

u, 33
unit, 22
upper, 33
userColor, 11

V

Value
- default, 37
- false, 88
- l, 33
- lower, 33
- true, 33
varrkiv, 58
VarStep, 44
varStep, 47
VarStepEpsilon, 6, 44
varsteptol, 6, 58
veearrowangle, 96
veearrowlength, 96
veearrowlinewidth, 96

W

whichabs, 57
whichord, 57

X

xkeyval, 2
xunit, 24
xyLabel, 2

Y

yunit, 24