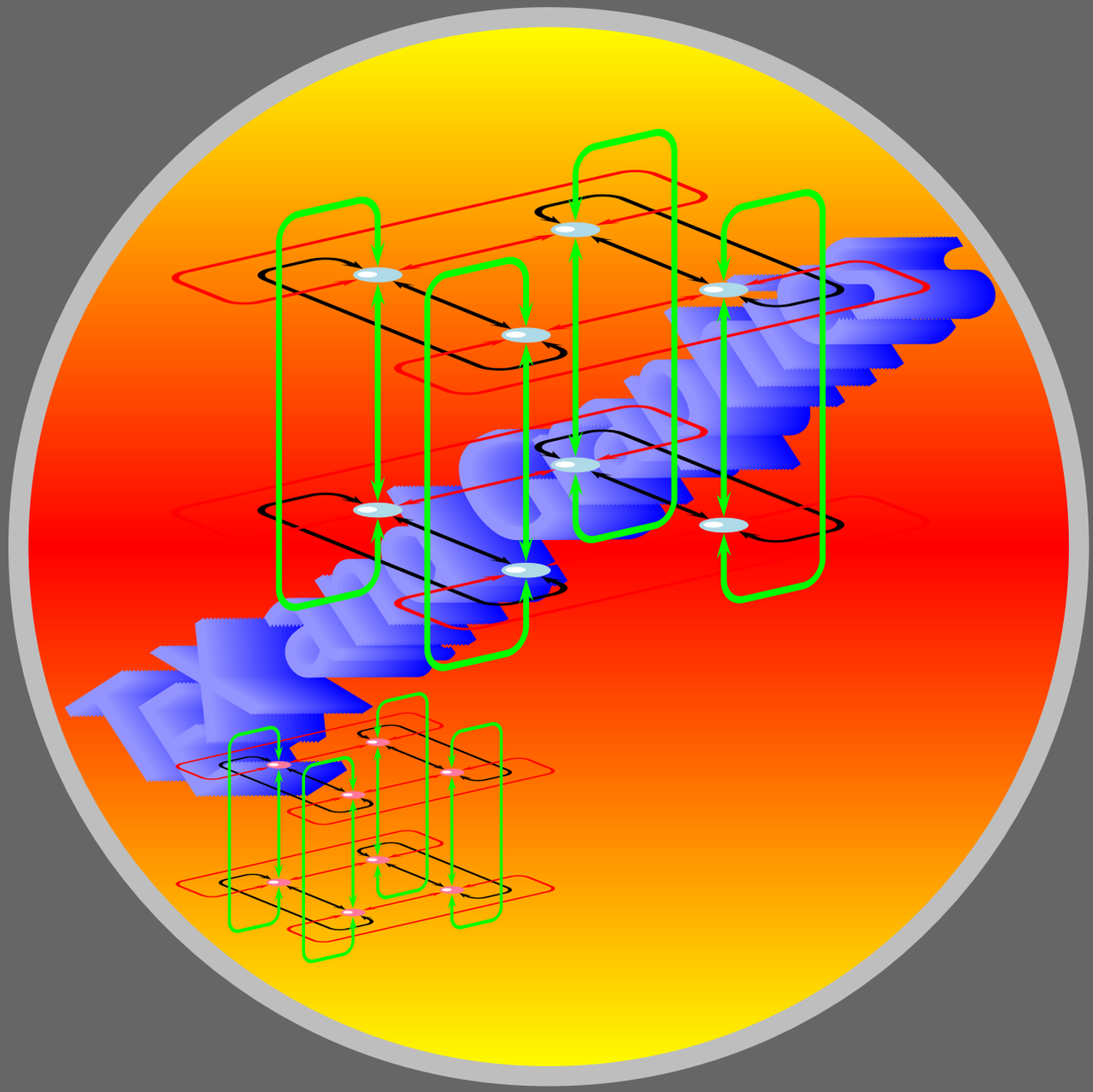


Graphics with PSTricks

Online LaTeX Tutorial

Part II – Graphics

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

http://www.tug.org.in

1. Graphics with PSTricks

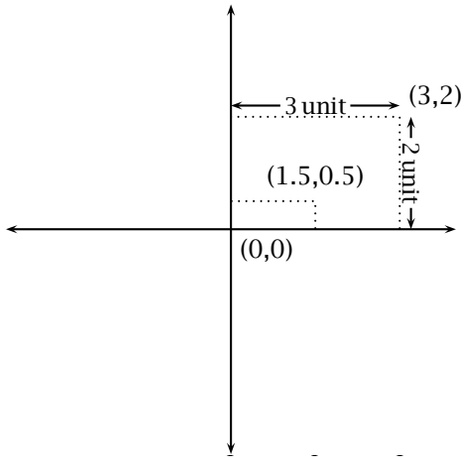
LaTeX has only limited drawing capabilities, while PostScript is a page description language which has a rich set of drawing commands; and there are programs (such as dvips) which translate the dvi output to PostScript. So, the natural question is whether one can include PostScript code in a TeX source file itself for programs such as dvips to process after the TeX compilation? This is the idea behind the PSTricks package of Timothy van Zandt. The beauty of it is one need not know PostScript to use it—the necessary PostScript code can be generated by TeX macros defined in the package.

Online LaTeX Tutorial
Part II – Graphics

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

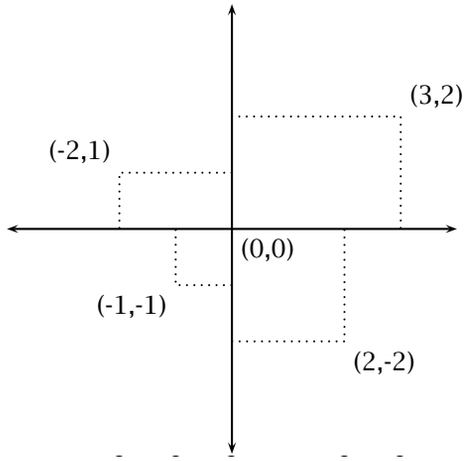
1.1. Getting the points

Any picture is drawn by stringing together appropriate points. How do we specify the points we need? We've a method of specifying each point in a plane using a pair of numbers, thanks to the 17th century French mathematicians Pierre de Fermat and René Descartes. The method is to fix a pair of perpendicular lines (called *axes*) and label each point with the numbers representing its distance from these two points (called *coordinates*) as shown in the figure below:

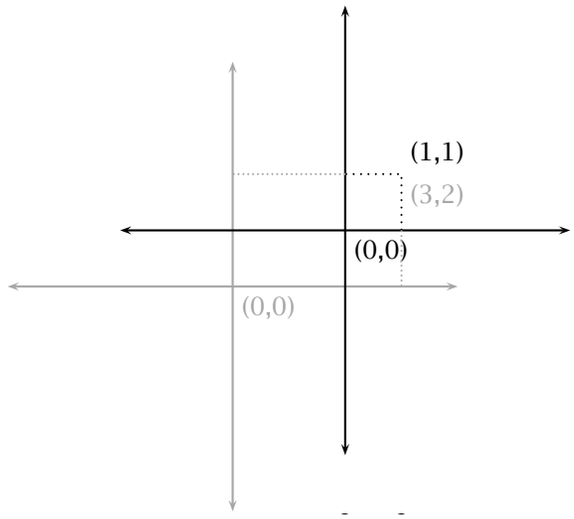


Note that the meeting point of the axes (called the *origin*) has coordinates (0,0). In order to associate each pair of numbers with a *unique* point, we make the convention that horizontal distances to the left of the origin and vertical distances below the origin are *negative* as illustrated below:

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



Another fact to note is that the coordinates of points depend on the position of the axes chosen, so that the same point has different pairs of coordinates with respect to different set of axes. This is illustrated in the figure below, where the point which originally had coordinates (3,2) with respect to the axes shown in gray has new coordinates (1,1) with respect to new axes shown in black.



The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

The PSTricks package uses coordinates to specify points to plot and then various other commands to join them.

Online LaTeX Tutorial
Part II – Graphics

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

1.2. Drawing Dots

Now let's see how to draw pictures with PSTricks. The basic package to use is pstricks and so we assume in all the codes given below that this package has been loaded with the command `\usepackage{pstricks}` in the document preamble.

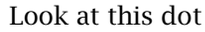
Let's start with the simplest of graphical objects—a single dot. Type in the code below in your document:

```
Look at this dot \psdots(1,0)
```

and TEX compile the document. To produce the PostScript, you'll have to use the dvips program or any other dvi to PostScript translator available in your system. With dvips, this done by the command

```
dvips filename -o
```

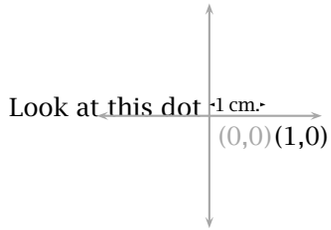
where *filename* is the name of your file *without any extension* (or with the extension `.dvi`). This creates a PostScript file of the same name but with the extension `.ps` which you can view using a PostScript previewer, such as ghostview. It looks like this:



Some explanations are in order. Evidently the command to draw a dot is `\psdots` followed by the coordinates of the point where the dot is to be placed. But we know that the assignment of coordinates to points (and *vice versa*) makes sense only after fixing the axes. So when we specify coordinates such as `(0,1)` as above, what are the axes used? By default, PSTricks uses the current point in TEX as the origin and horizontal and vertical lines through this point as the axes. Again, the default unit is 1 cm. Thus in the above example, a point is drawn 1 cm. away from the letter `t` in `dot`. This is

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

illustrated in the figure below, where the (invisible) axes are shown in gray.



A single `\psdots` command can be used to plot any number of points. For example, the input

```
Look at these dots \psdots(0,0)(2,0)(1,1)
```

produce the (PostScript) output

Look at these dots

Now suppose we try

```
Look at these dots \psdots(0,0)(2,0)(1,1) forming the vertices
(corners)  of a triangle.
```

the output produced is

Look at these dotsforming the vertexes (corners) of a triangle.

What happened? Why were the dots overwritten? What happened actually is that TEX did not reserve any space for the picture (recall that the

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

picture is drawn *after* the TEX compilation) and so the dots were drawn over the text. (if you look closely, you can see that the dots are over the letters). This brings up an important point to be kept in mind: *most of the* PSTricks *commands produce 0-dimensional boxes in* TEX. So, we must ensure that TEX leaves enough space for the pictures to be drawn, by enclosing the picture in a TEX box of suitable size. PSTricks itself provides a convenient method of doing this, in the form of the `pspicture` environment. See how we can modify the previous example:

```
\begin{pspicture}(-0.5,0)(2.5,1)
  \psdots(0,0)(2,0)(1,1)
\end{pspicture}
```

This gives the output

Here the pairs (-0.5,0) and (2.5,1) are the coordinates of the bottom-left and top-right corners of a box which encloses the picture as shown in the figure below:

Look at these dots

forming the vertexes of a triangle.

In fact, the first pair of coordinates is optional and defaults to (0,0). Thus for example,

...

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(1,2)...\end{pspicture}
```

is equivalent to

```
\begin{pspicture}(0,0)(1,2) ... \end{pspicture}
```

We can also 'display' the picture by

```
\begin{pspicture}(-0.5,-0.5)(2.5,1.5)
  \psdots(0,0)(2,0)(1,1)
\end{pspicture}
```

This produces

Can you see why the *second* coordinate of the 'box' is changed to -0.5 and 1.5 from its values 0 and 1 in the previous example?

The dots we've been drawing so far are all circular and black. How about square and white dots? Change the input of the previous example as follows:

```
Look at these dots
 \begin{center}
   \begin{pspicture}(-0.5,-0.5)(2.5,1.5)
     \psdots[dotstyle=square](0,0)(2,0)(1,1)
   \end{pspicture}
 \end{center}
forming the vertices of a triangle.
```


Online LaTeX Tutorial
Part II – Graphics

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

We then get the output shown below:
Look at these dots

forming the vertexes of a triangle.

Thus the shape of the dots is controlled by the parameter `dotstyle` and it's to be specified within square brackets after the `\psdots` command. The various possible values of this parameter and the corresponding shape of the dots is shown in the table below:

STYLE	EXAMPLE	STYLE	EXAMPLE
*		o	
+	+ + +	x	× × ×
oplus	⊕ ⊕ ⊕	otimes	⊗ ⊗ ⊗
asterisk	* * *	|	| | |
triangle		triangle*	
square		square*	
diamond	♦ ♦ ♦	diamond*	◊ ◊ ◊
pentagon		pentagon*	

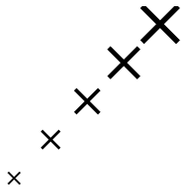
Also, dots can be scaled using the parameter `dotscale` and rotated using the parameter `dotangle`. For example

Online LaTeX Tutorial
Part II – Graphics

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(-0.5,-0.5)(2.5,2.5)
 \psdots[dotstyle=+,dotangle=45](0,0)
 \psdots[dotstyle=+,dotscale=1.5,
                dotangle=45](0.5,0.5)
 \psdots[dotstyle=+,dotscale=2,
                dotangle=45](1,1)
 \psdots[dotstyle=+,dotscale=2.5,
                dotangle=45](1.5,1.5)
 \psdots[dotstyle=+,dotscale=3,
               dotangle=45](2,2)
\end{pspicture}
```

gives



Instead of scaling, we can explicitly specify the size of dots. But this we'll discuss in the next section (with a reason, of course).

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

http://www.tug.org.in

1.3. Simple Lines

Let's see how we draw lines next. The command is `\psline` with the coordinates of the points to be joined. For example

```
     Look at the line segment below
     \begin{center}
       \begin{pspicture}(0,0)(3.5,2.5)
         \psline(2,1)(3,2)
       \end{pspicture}
     \end{center}
equally slanted to the horizontal and the vertical.
```

gives
Look at the line segment below

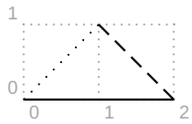
equally slanted to the horizontal and the vertical.

We can draw dashed or dotted lines using the `linestyle` parameter. Thus

```
\begin{pspicture}(0,0)(2,1)
    \psline(0,0)(2,0)
    \psline[linestyle=dashed](2,0)(1,1)
    \psline[linestyle=dotted](1,1)(0,0)
  \end{pspicture}
```

gives

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

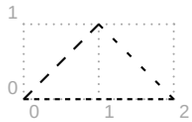


In this and many of the pictures below, we include a “coordinate grid” for convenience of reference. It is not produced by the code given alongside.

In the `dashed` style, the length of the black and white segments is controlled by the parameter `dash` Thus `dash=3pt 2pt` produces dashed line with black segments of length 3 pt. and white segments of length 2 pt. Thus

```
\begin{center}
 \begin{pspicture}(-0.5,-0.5)(2.5,1.5)
  \psline[linestyle=dashed,dash=2pt 2pt]
          (0,0)(2,0)
  \psline[linestyle=dashed,dash=2pt 5pt]
          (2,0)(1,1)
  \psline[linestyle=dashed,dash=5pt 5pt]
          (1,1)(0,0)
 \end{pspicture}
\end{center}
```

gives



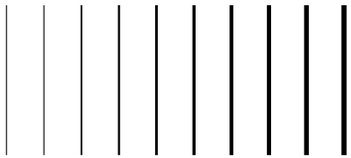
The default value of `dash` is 5 pt 3 pt. Again, in the `dotted` style, the distance between dots is controlled by the parameter `dotsep` whose default value is 3 pt.

We can also alter the thickness of the lines by changing the value of the parameter `linewidth` which has a default value of 0.8 pt. Look at the example below:

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{center}
  \begin{pspicture}(0,-0.5)(2.5,4.5)
    \psline[linewidth=0.2pt](0,0)(0,2)
    \psline[linewidth=0.4pt](0.5,0)(0.5,2)
    \psline[linewidth=0.6pt](1,0)(1,2)
    \psline[linewidth=0.8pt](1.5,0)(1.5,2)
    \psline[linewidth=1pt](2,0)(2,2)
    \psline[linewidth=1.2pt](2.5,0)(2.5,2)
    \psline[linewidth=1.4pt](3,0)(3,2)
    \psline[linewidth=1.6pt](3.5,0)(3.5,2)
    \psline[linewidth=1.8pt](4,0)(4,2)
    \psline[linewidth=2pt](4.5,0)(4.5,2)
  \end{pspicture}
\end{center}
```

produces



Online LaTeX Tutorial
Part II – Graphics

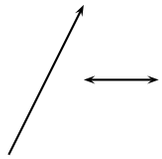
The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

1.4. Ends of Lines

Lines can be provided with arrowheads. This is done by the `arrows` parameter

```
\begin{center}
  \begin{pspicture}(0,-0.5)(2,2.5)
    \psline[arrows=->](0,0)(1,2)
    \psline[arrows=<->](1,1)(2,1)
  \end{pspicture}
 \end{center}
```

produces



Instead of arrowheads, lines can be made to terminate with circles, T-bars and so on, using the parameter `arrows`. The available values of this parameter and the corresponding line terminators are given in the Table 1.1. We can mix and match these terminators as values for the `arrows` parameter such as `*->` or `|-<<`.

Certain terminators are clearly seen only for thick lines. For example

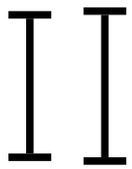
```
\begin{pspicture}(-0.5,-0.5)(2.5,2.5)
  \psline[linewidth=0.1cm,arrows=|-|]
          (1,0)(1,2)
  \psline[linewidth=0.1cm,arrows=|*-|*]
          (2,0)(2,2)
\end{pspicture}
```

gives

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

VALUE	EXAMPLE	NAME
-		none
<->		arrowheads
>-<		reverse arrowheads
<<->>		double arrowheads
>>-<<		double reverse arrowheads
|-|		T-bars, flush with end points
|*-|*		T-bars, centered on end points
[-]		square brackets
(-)		rounded brackets
o-o		circles, centered on end points
-		disks, centered on end points
-		disks, flush to endpoints
c-c		extended, rounded ends
cc-cc		flush round ends
C-C		extended, square ends

Table 1.1: Line terminators



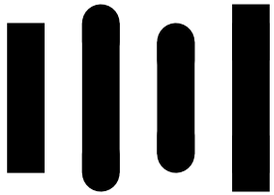
To see some other terminators clearly, thicker lines are needed. Thus

Online LaTeX Tutorial
Part II – Graphics

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(-0.5,-0.5)(3.5,2.5)
\psline[linewidth=0.5cm](0,0)(0,2)
\psline[linewidth=0.5cm,arrows=c-c]
        (1,0)(1,2)
\psline[linewidth=0.5cm,arrows=cc-cc]
        (2,0)(2,2)
\psline[linewidth=0.5cm,arrows=C-C]
        (3,0)(3,2)
\end{pspicture}
```

gives



The `arrows` parameter can also be specified as an optional argument within *braces* after the other options (in square brackets). Thus instead of

```
\psline[linestyle=dotted,arrows=<->](0,0)(2,0)
```

we can also write

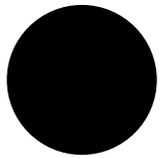
```
\psline[linestyle=dotted]{<->}(0,0)(2,0)
```

Now is the time to talk of (no, not cabbages and kings) the size of dots. The diameter of a circular dot is 2.5 times the current linewidth plus .5 pt. This can be changed by the parameter `dotsize`. Thus for example

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{center}
\begin{pspicture}(0,0)(2,2)
  \psdot[linewidth=0.1cm,dotsize=1cm 10](1,1)
\end{pspicture}
\end{center}
```

gives



which is a circular disk of diameters $10 \times 0.1 + 1 = 2$ centimeters. (We'll soon see better method of drawing such disks). The polygonal dots are sized to have the same area as circles. The `dotsize` is made to depend on `linewidth` since dots are often used in conjunction with lines as in `arrows` (and `showpoints` which we will discuss later). Note that the dotsize can be set to any absolute value independent of the linewidth by setting the second number of the `dotsize` parameter to 0.

There are parameters determining the dimensions of the other types of line terminators also, which are given in Table 1.2. In this, *width* refers to a dimension perpendicular to the line and length refers to a dimension in the direction of the line.

The example below illustrates the use of some of these parameters

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

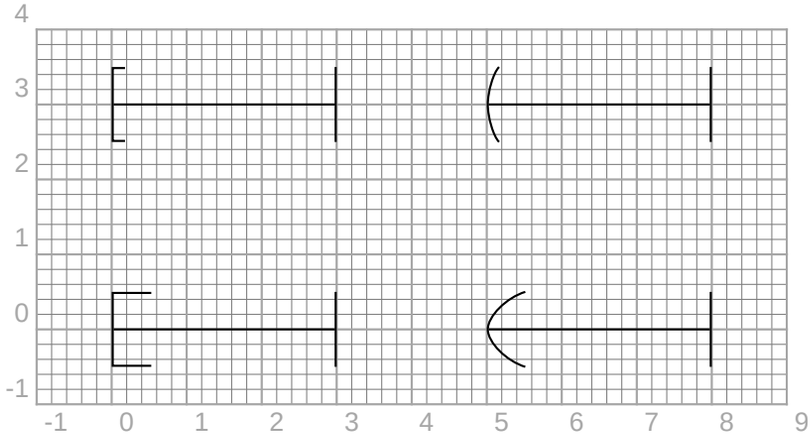
PARAMETER	VALUE	DESCRIPTION	DEFAULT VALUE
`dotsize` = *dim num*	*num* × `linewidth` + *dim*	the diameter of a circle or disc	0.5 pt 5
`tbarsize` = *dim num*	*num* × `linewidth` + *dim*	the *width* of a T-bar, square bracket or round bracket	2 pt 5
`bracketlength` = *num*	*number* × *width*	the *length* of a square bracket	0.15
`rbracketlength` = *num*	*number* × *width*	the *length* of a round bracket	0.15

Table 1.2: Parameters for line terminators

```
\begin{center}
\begin{pspicture}(-1,-1)(9,4)
\psline[tbarsize=1cm 0,bracketlength=0.5]{[-|}(0,0)(3,0)
\psline[tbarsize=1cm 0]{[-|}(0,3)(3,3)
\psline[tbarsize=1cm 0,rbracketlength=0.5]{(-|}(5,0)(8,0)
\psline[tbarsize=1cm 0]{(-|}(5,3)(8,3)
\end{pspicture}
\end{center}
```

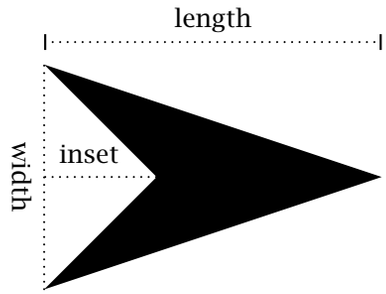
which produces the output below.

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



Note that the coordinate grid in the picture above is not produced by the given code.

The shape of arrowheads is determined by its *length, width* and *inset* and the parameters controlling them are `arrowsize, arrowlength` and `arrowinset` as shown in the figure below:



$$\texttt{arrowsize} = \textit{dim num}$$
$$\text{width} = \textit{num} \times \texttt{linewidth} + \textit{dim}$$
$$\text{length} = \texttt{arrowlength} \times \text{width}$$
$$\text{inset} = \texttt{arrowinset} \times \text{length}$$

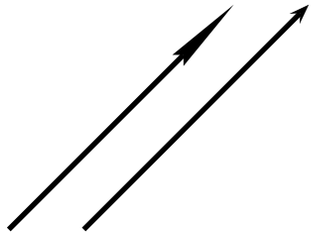
The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

The default values of the parameters are

`arrowsize` = 2 pt 3 `arrowlength` = 1.4 `arrowinset` = 0.4

The example below illustrates the effect of changing these parameters.

```
\begin{center}
\begin{pspicture}(0.5,0.5)(5.5,4.5)
\psline[linewidth=2pt,
        arrowsize=2pt 2,
        arrowlength=5,
        arrowinset=0.1]
        {->}(1,1)(4,4)
\psline[linewidth=2pt]
       {->}(2,1)(5,4)
\end{pspicture}
\end{center}
```



We can also draw “double lines” by setting the parameter `doubleline` to `true` (by default, it’s `false`). For example

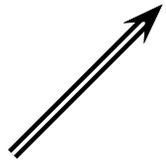
```
\begin{center}
\begin{pspicture}(-0.5,-0.5)(2.5,2.5)
\psline[linewidth=0.06,
        doubleline=true,
        doublesep=0.05,
        (0,0)(2,2)
\end{pspicture}
\end{center}
```


Online LaTeX Tutorial

Part II – Graphics

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

gives



Online LaTeX Tutorial
Part II – Graphics

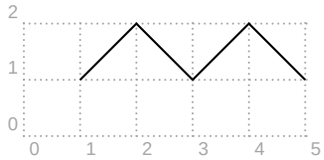
The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

1.5. Bent Lines and Polygons

As in the case of `\psdots` we can draw multiple lines with a single `\psline` command. For example,

```
\begin{center}
\begin{pspicture}(0,0)(5,2)
\psline(1,1)(2,2)(3,1)(4,2)(5,1)
\end{pspicture}
\end{center}
```

gives



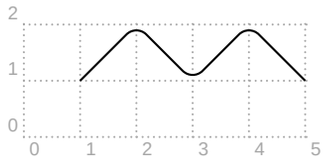
Note that the coordinate grid is not produced by the code given alongside.

The corners in the above picture can be rounded by giving a positive value to the `linearc` parameter which has default value 0 pt. It is actually the radius of the arc drawn at the corners. Thus

```
\begin{center}
\begin{pspicture}(0,0)(5,2)
\psline[linearc=0.25]%
       (1,1)(2,2)(3,1)(4,2)(5,1)
\end{pspicture}
\end{center}
```

gives

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

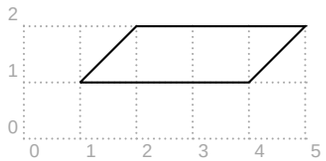


Now change the value of `linearc` to 0.5 in the above code and see what happens.

Polygons can be drawn with `\psline` by taking the first and the last points same. For example

```
\begin{center}
\begin{pspicture}(0,0)(5,3)
\psline(1,1)(2,2)(5,2)(4,1)(1,1)
\end{pspicture}
\end{center}
```

gives



We can also use the command `\pspolygon` to draw polygons. Here, we need not repeat the starting point as in `\psline`. Thus in the last example above, the parallelogram could also be drawn by the command

```
\pspolygon(1,1)(2,2)(5,2)(4,1)
```

instead of the command

```
\psline(1,1)(2,2)(5,2)(4,1)(1,1)
```

The `\pspolygon` command also has a "starred" version which draws a

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

"filled up" polygon. For example

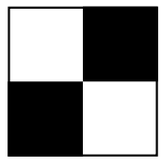
```
\begin{center}
\begin{pspicture}(0,0)(5,3)
\pspolygon*(1,1)(2,2)(5,2)(4,1)
\end{pspicture}
\end{center}
```

gives

For drawing rectangles, there's a simpler command `\psframe` in which we need only specify the bottom-left and top-right coordinates. There's also a `\psframe*` command for a filled-up version. For example,

```
\begin{center}
\begin{pspicture}(0,0)(6,4)
\psframe(1,1)(3,3)
\psframe*(1,1)(2,2)
\psframe*(2,2)(3,3)
\end{pspicture}
\end{center}
```

gives



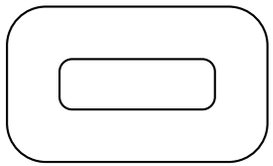
The corners of a frame can also be rounded. The parameter to set is `framearc`. If we set `framearc=`*number*, then the radius of the rounded corners is half the *number* times the width or height of the frame, whichever

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

is less. Thus

```
\begin{center}
\begin{pspicture}(-0.5,0.5)(5.5,3.5)
\psframe[framearc=0.5](0,0)(5,3)
\psframe[framearc=0.5](1,1)(4,2)
\end{pspicture}
\end{center}
```

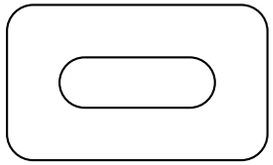
gives



Note that the corners of the larger rectangle are more rounded, as should be obvious from the definition of the `framearc` parameter. The radius of the corners can be made he same by setting the parameter `cornersize` to `absolute` (its default setting is `relative`) and then setting the radius using the `linearc` parameter as in the example below:

```
\begin{center}
\begin{pspicture}(-0.5,-0.5)(5.5,3.5)
\psframe[cornersize=absolute,%
         linearc=0.5](0,0)(5,3)
\psframe[cornersize=absolute,%
         linearc=0.5](1,1)(4,2)
\end{pspicture}
\end{center}
```


The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



There are also commands to draw isoceles triangles (that is, triangles in which two sides are equal) and rhombuses (diamonds). The command

```
\pstriangle((x,y)(b,h))
```

draws an isoceles triangle with its base horizontal, the mid-point of its base at (x, y), length of base b and height h while the command

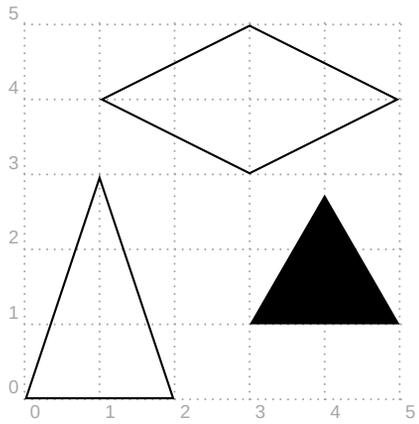
```
\psdiamond((x,y)(d_1,d_2))
```

draws a rhombus with its diagonals along the horizontal and the vertical, which meet (x, y) and have lengths $2d_1$ and $2d_2$. Thus

```
\begin{center}
\begin{pspicture}(0,0)(5,5)
\pstriangle(1,0)(2,3)
\pstriangle*(4,1)(2,1.732)
\psdiamond(3,4)(2,1)
\end{pspicture}
\end{center}
```

gives

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



So far we've been drawing only straight lines (except for smoothing some corners). We'll discuss curves in the next few sections.

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

1.6. Simple Curves

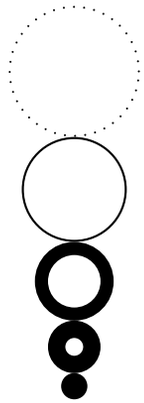
Circles, ellipses, circular arcs and so on can be easily drawn in PSTricks, using preset commands. Let's start with circles. The command is `\pscircle` (what else?) and we've to specify the coordinates of the center and the length of the radius. Recall that the default unit is centimeter, so that to produce a circle of radius 0.5 cm centered at (2,1), we write

```
\pscircle(2,1){0.5}
```

Since a circle is only a curved "line", various line parameters discussed earlier can also be used. There is also a starred version `\pscircle*` which gives a "solid" circle. See the example below:

```
\begin{center}
 \begin{pspicture}(-1,-1)(3,8)
  \pscircle*(1,0.25){0.25}
  \pscircle[linewidth=0.33]%
           (1,1){0.5}
  \pscircle[linewidth=0.25]%
           (1,2.25){0.75}
  \pscircle(1,4){1}
  \pscircle[linestyle=dotted]%
           (1,6.25){1.25}
 \end{pspicture}
\end{center}
```

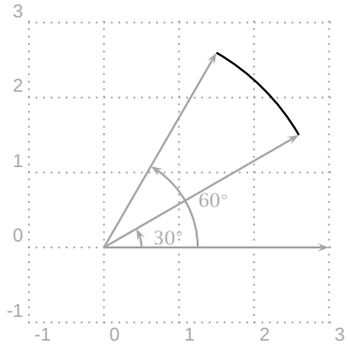

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



Pieces of circles can also be easily drawn. For example, the command `\psarc` draws a circular arc of specified center and radius from a given *angle* to another going *counterclockwise*. Note that the angles are measured from the horizontal. In the example below, we show the radii and the angles in gray along with the grid. (note that these are not produced by the given code).

```
\begin{center}
 \begin{pspicture}(-1,-1)(3,3)
  \psarc(0,0){3}{30}{60}
 \end{pspicture}
\end{center}
```

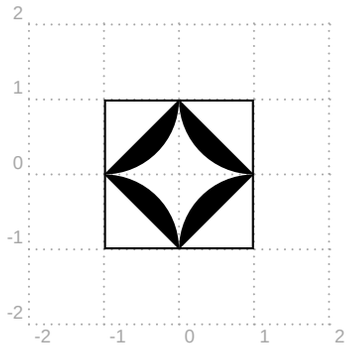

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



There's also a starred version \psarc* which draws a solid "segment" of a circle. For example,

```
\begin{center}
 \begin{pspicture}(-2,-2)(2,2)
  \psframe(-1,-1)(1,1)
  \psarc*(-1,-1){1}{0}{90}
  \psarc*(1,-1){1}{90}{180}
  \psarc*(1,1){1}{180}{270}
  \psarc*(-1,1){1}{270}{360}
 \end{pspicture}
\end{center}
```

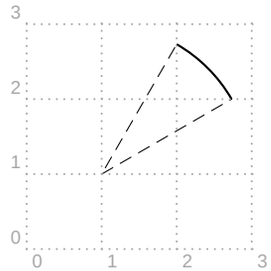
gives



The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

While making a picture containing circular arcs, it may sometimes be convenient to "see" the center and radii. If the parameter `showpoints` is set to `true` (its default value is `false`), then the command `\psarc` (or `\psarc`) draws dashed lines from the center to the extremities of the arc. (This setting can be used with other commands also, where it will draw appropriate control points or lines). See the example below:

```
\begin{center}
 \begin{pspicture}(0,0)(3,3)
  \psarc[showpoints=true]%
       (1,1){2}{30}{60}
  \end{pspicture}
\end{center}
```

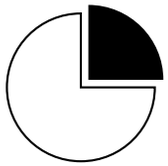


If we want to draw an arc with its bounding radii, we can use the `\pswedge` command. The starred version `\pswedge*` draws a solid sector as shown in the example below:

```
\begin{center}
 \begin{pspicture}(-1.5,-1.5)(1.5,1.5)
  \pswedge(0,0){1}{90}{360}
  \pswedge*(0.1,0.1){1}{0}{90}
 \end{pspicture}
\end{center}
```

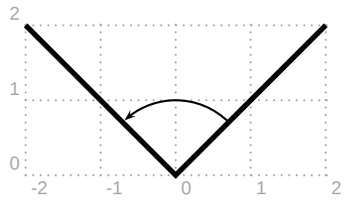

Online LaTeX Tutorial
Part II – Graphics

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



The line terminators discussed earlier can be used with arcs also. If we want to show the angle between two (thick) intersecting lines using an arc with an arrowhead, we'd like to have the tip of the arrow would *just* touch the line. For this, we can use the parameters `arcsepA` and `arcsepB`. If we set `arcsepA`=*dim*, then the first angle in the `\psarc` command would be adjusted so that the arc would just touch a line of width *dim* from the center of the arc in the direction of this angle. The parameter `arcsepB` makes a similar adjustment in the second angle. The parameter `arcsep` adjusts both the angles. The example below illustrates this.

```
\begin{center}
 \begin{pspicture}(-2,0)(2,2)
  \psline[linewidth=2pt]%
        (2,2)(0,0)(-2,2)
  \psarc[arcsepB=2pt]{->}%
        (0,0){1}{45}{135}
 \end{pspicture}
\end{center}
```



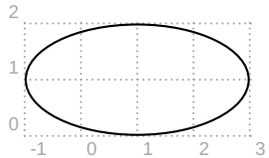
To see the difference, try the same code without the setting of `arcsepB`. An ellipse is a sort of a stretched circle and can be drawn much the

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

same way as s circle. The command is `\psellipse` and we have to specify the center and half the width and height (technically, the "semi-major" and "semi-minor" axes). Thus to draw an ellipse centered at (1,1) with width width 4 cm and height 2 cm, we type

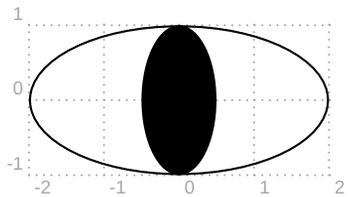
```
\begin{center}
 \begin{pspicture}(-1,0)(3,2)
  \psellipse(1,1)(2,1)
 \end{pspicture}
\end{center}
```

which gives



There's also a `\psellipse*` which, as you've probably guessed, draws a solid black ellipse.

```
\begin{center}
 \begin{pspicture}(-2,-1)(2,1)
  \psellipse(0,0)(2,1)
  \psellipse*(0,0)(0.5,1)
 \end{pspicture}
\end{center}
```



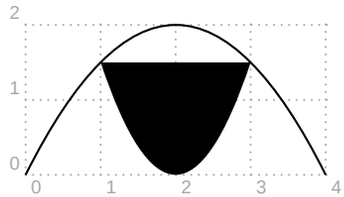
Online LaTeX Tutorial
Part II – Graphics

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

Another curve for which a preset command is available is a parabola (the path of a stone thrown at an angle, for example). It is drawn by the command `\parabola` (surprise!). We must specify the starting point and the maximum or minimum. As usual, we have a `\parabola*` also. Thus

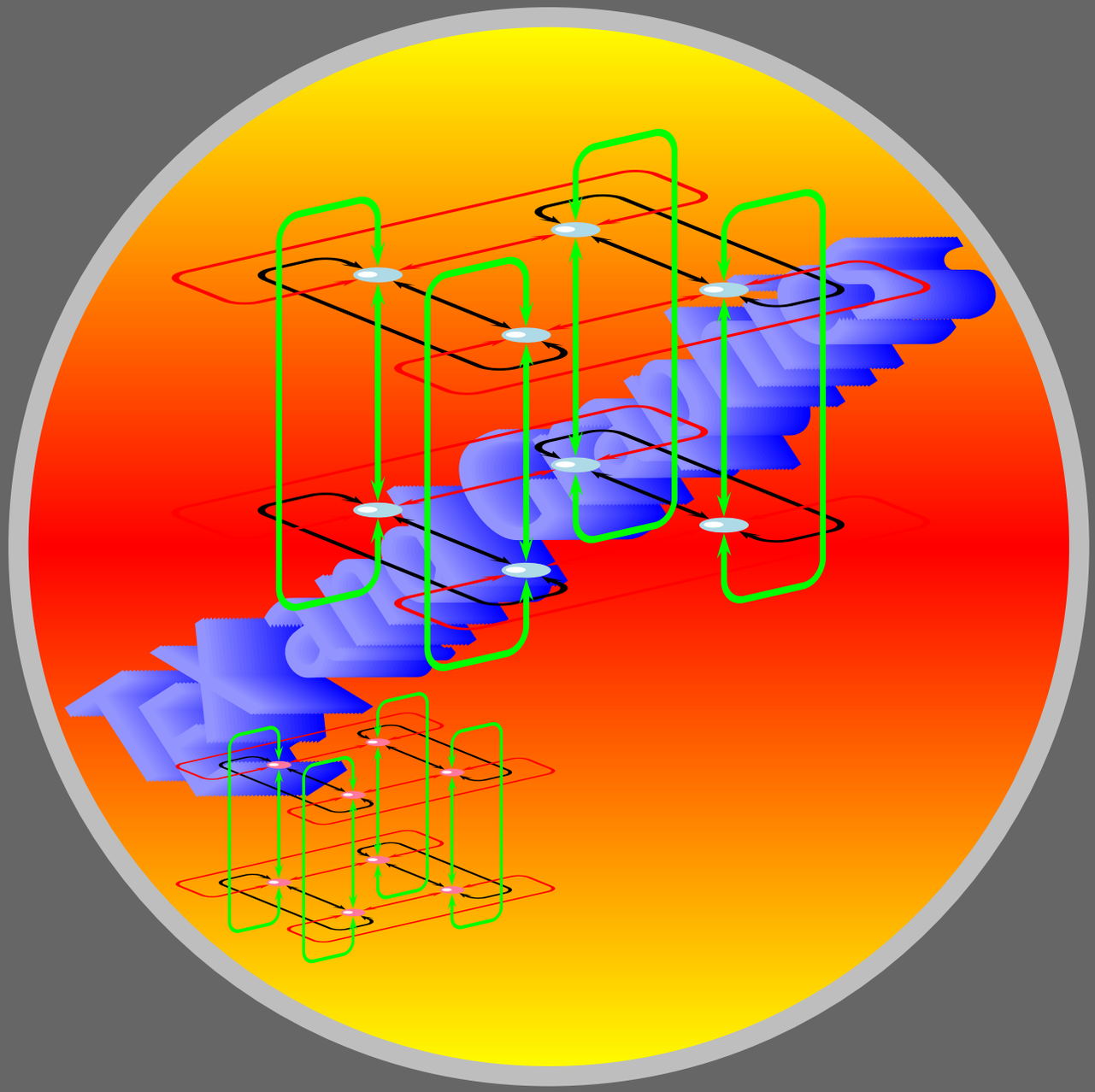
```
\begin{center}
 \begin{pspicture}(0,0)(4,2)
  \parabola(0,0)(2,2)
  \parabola*(1,1.5)(2,0)
 \end{pspicture}
\end{center}
```

gives



Online LaTeX Tutorial
Part II – Graphics

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



Colorful Tricks

Online LaTeX Tutorial

Part II – Graphics

PSTricks

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

2. Colorful Tricks

Seeing the (ps)tricks so far, at least some of you may be wishing for a bit of color in the graphics. Here's good news for such people: you can have your wish! PSTricks comes with a set of macros that provide a basic set of colors and lets you define your own colors. However, it has some incompatibility with the LaTeX package color. However, David Carlisle has written a package pstcol which modifies the PSTricks color interface to work with LaTeX colors. All of our examples in this chapter assumes that this package is loaded, using the command `\usepackage{pstcol}` in the preamble. Note that this loads the pstricks package also, so that it need not be separately loaded.

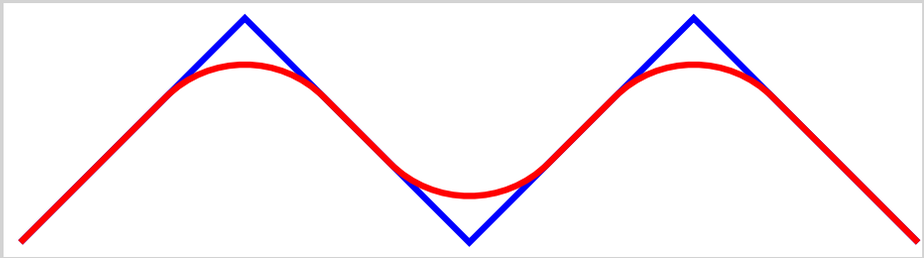
The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

2.1. Ordinary colors

The colors `red`, `green`, `blue`, `cyan`, `magenta`, `yellow`, `black`, `white` are predefined in pstcol and various parts of a picture can be colored with these by assigning these values to the various "color" parameters.

Lines are colored by setting the parameter `linecolor`. Thus we can colorfully distinguish the effect of `linearc` (do you remember this parameter?) as in the example below:

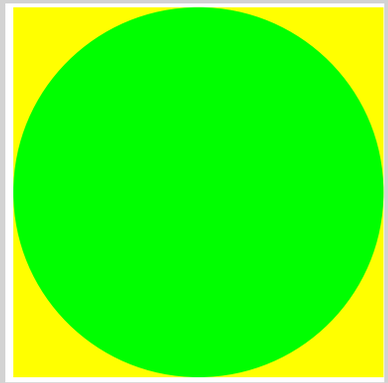
```
\begin{pspicture}(0,0)(5,2)
\psline[linecolor=blue](1,1)(2,2)(3,1)(4,2)(5,1)
\psline[linearc=0.5,linecolor=red](1,1)(2,2)(3,1)(4,2)(5,1)
\end{pspicture}
```



The same parameter `linecolor` can also be used to color "solid" objects made with "starred" commands as in the next example:

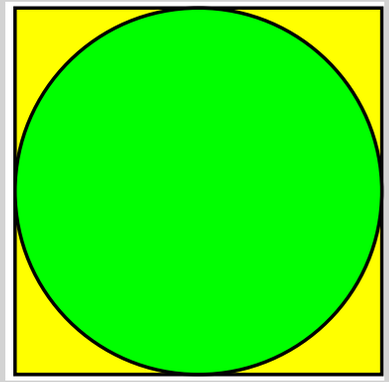
```
\begin{pspicture}(0,0)(3,3)
\psframe*[linecolor=yellow](0,0)(3,3)
\pscircle*[linecolor=green](1.5,1.5){1.5}
\end{pspicture}
```


The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



Another way of coloring closed regions is to use the `fillstyle` and `fillcolor` parameters. For example

```
\begin{pspicture}(0,0)(3,3)
  \psframe[fillstyle=solid, fillcolor=yellow](0,0)(3,3)
  \pscircle[fillstyle=solid, fillcolor=green](1.5,1.5){1.5}
\end{pspicture}
```



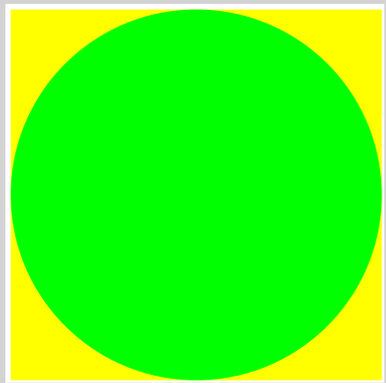
Do you see any difference? Yes, the black outlines. Note that with a "solid" object made with the starred commands and `linecolor`, you're sort of painting the *entire* object—and this includes the boundary—*line by line*,

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

while in the case of a “closed” region and `fillcolor`, you’re painting only the region *enclosed* by the boundary *after* drawing the boundary in the default `linecolor`, which is black.

We can get rid of the “boundaries” in this example by setting the `linestyle` parameter to `none`. (Do you remember other possible values of this parameter?)

```
\begin{pspicture}(0,0)(3,3)
  \psframe[linestyle=none,fillstyle=solid,fillcolor=yellow](0,0)(3,3)
  \pscircle[linestyle=none,fillstyle=solid,fillcolor=green](1.5,1.5){1.5}
\end{pspicture}
```

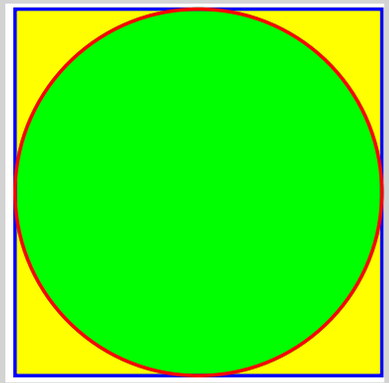


which is exactly the same output of the second example. (In fact what the starred versions of the commands do is to set `linewidth` to 0, `linestyle` to `none`, `fillcolor` to `linecolor` and `fillstyle` to `solid`.)

On the other hand, to put a boundary around a “solid” object colored with “linecolor”, just redraw the boundary, and you can do this with any color:

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(0,0)(3,3)
   \psframe*[linecolor=yellow](0,0)(3,3)
   \pscircle*[linecolor=green](1.5,1.5){1.5}
   \psframe[linecolor=blue](0,0)(3,3)
   \pscircle[linecolor=red](1.5,1.5){1.5}
 \end{pspicture}
```



Online LaTeX Tutorial

Part II – Graphics

PSTricks

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

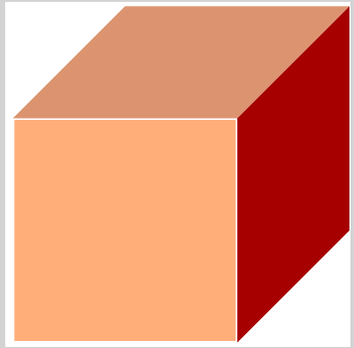
2.2. More colors

Some dvi drivers support a named color model, which means in practical terms that you can use the names of a certain set of predefined colors. For example, the dvips offers 64 colors as listed in the Figure 2.1. To use these colors, load the package pstcol with the option usenames as

```
\usepackage[usenames]{pstcol}
```

Then for example, with the code given below, you can produce the picture shown alongside:

```
\begin{pspicture}(0,0)(3,3)
 \psframe[linestyle=none, fillstyle=solid,fillcolor=Apricot](0,0)(2,2)
 \pspolygon[linestyle=none,fillstyle=solid,fillcolor=Tan](0,2)(2,2)(3,3)(1,3)
 \pspolygon[linestyle=none,fillstyle=solid,fillcolor=Mahogany]
                        (2,0)(3,1)(3,3)(2,2)
\end{pspicture}
```



The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

NAME	CMYK	COLOR	NAME	CMYK	COLOR
GreenYellow	0.15,0,0.69,0		RoyalPurple	0.75,0.90,0,0	
Yellow	0,0,1,0		BlueViolet	0.86,0.91,0,0.04	
Goldenrod	0,0.10,0.84,0		Periwinkle	0.57,0.55,0,0	
Dandelion	0,0.29,0.84,0		CadetBlue	0.62,0.57,0.23,0	
Apricot	0,0.32,0.52,0		CornflowerBlue	0.65,0.13,0,0	
Peach	0,0.50,0.70,0		MidnightBlue	0.98,0.13,0,0.43	
Melon	0,0.46,0.50		NavyBlue	0.94,0.54,0,0	
YellowOrange	0,0.42,1,0		RoyalBlue	1,0.50,0,0	
Orange	0,0.61,0.87,0		Blue	1,1,0,0	
BurntOrange	0,0.51,1,0		Cerulean	0.94,0.11,0,0	
Bittersweet	0,0.75,1,0.24		Cyan	1,0,0,0	
RedOrange	0,0.77,0.87,0		ProcessBlue	0.96,0,0,0	
Mahogany	0,0.85,0.87,0.35		SkyBlue	0.62,0,0.12,0	
Maroon	0,0.87,0.68,0.32		Turquoise	0.85,0,0.20,0	
BrickRed	0,0.89,0.94,0.28		TealBlue	0.86,0,0.34,0.02	
Red	0,1,1,0		Aquamarine	0.82,0,0.30,0	
OrangeRed	0,1,0.50,0		BlueGreen	0.85,0,0.33,0	
RubineRed	0,1,0.13,0		Emerald	1,0,0.50,0	
WildStrawberry	0,0.96,0.39,0		JungleGreen	0.99,0,0.52,0	
Salmon	0,0.53,0.38,0		SeaGreen	0.69,0,0.50,0	
CarnationPink	0,0.63,0,0		Green	1,0,1,0	
Magenta	0,1,0,0		ForestGreen	0.91,0,0.88,0.12	
VioletRed	0,0.81,0,0		PineGreen	0.92,0,0.59,0.25	
Rhodamine	0,0.82,0,0		LimeGreen	0.50,0,1,0	
Mulberry	0.34,0.90,0,0.02		YellowGreen	0.44,0,0.74,0	
RedViolet	0.07,0.90,0,0.34		SpringGreen	0.26,0,0.76,0	
Fuchsia	0.47,0.91,0,0.08		OliveGreen	0.64,0,0.95,0.40	
Lavender	0,0.48,0,0		RawSienna	0,0.72,1,0.45	
Thistle	0.12,0.59,0,0		Sepia	0,0.83,1,0.70	
Orchid	0.32,0.64,0,0		Brown	0,0.81,1,0.60	
DarkOrchid	0.40,0.80,0.20,0		Tan	0.14,0.42,0.56,0	
Purple	0.45,0.86,0,0		Gray	0,0,0,0.50	
Plum	0.50,1,0,0		Black	0,0,0,1	
Violet	0.79,0.88,0,0		White	0,0,0,0	

Figure 2.1: Named colors in dvips

Online LaTeX Tutorial

Part II – Graphics

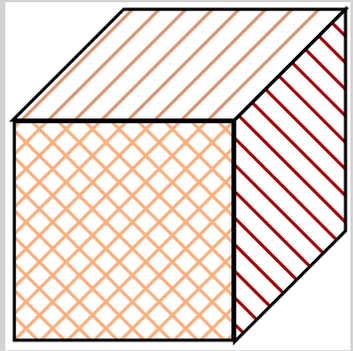
PSTricks

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

2.3. Fill—in style

We've often used the setting `fillstyle=solid` in the examples above. There are various other ways of filling up closed regions, by assigning different values to the parameter `fillstyle`. The values `vlines`, `hlines` and `crosshatch` fill the region with vertical lines, horizontal lines and criss-cross lines, as shown in the example below:

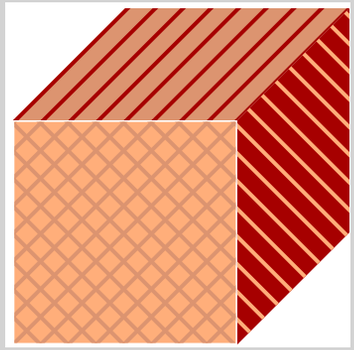
```
\begin{pspicture}(0,0)(3,3)
  \psframe[fillstyle=crosshatch,hatchcolor=Apricot](0,0)(2,2)
  \pspolygon[fillstyle=hlines,hatchcolor=Tan](0,2)(2,2)(3,3)(1,3)
  \pspolygon[fillstyle=vlines,hatchcolor=Mahogany](2,0)(3,1)(3,3)(2,2)
 \end{pspicture}
```



As can be seen from this example, the color of the lines making up the fill-pattern is set by the parameter `hatchcolor`. We can also set the background color using the parameter `fillcolor`, if we use the starred form of the *values* for the `fillstyle`. The example below illustrates this. Note also the use of the parameter `hatchwidth` which controls the width of the lines making up the pattern. Its default value is 0.8pt.

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(0,0)(4,4)
  \psframe[linestyle=none,fillstyle=crosshatch*,hatchcolor=Tan,%
           hatchwidth=1pt,fillcolor=Apricot](0,0)(2,2)
  \pspolygon[linestyle=none,fillstyle=hlines*,hatchcolor=Mahogany,%
             hatchwidth=1pt,fillcolor=Tan](0,2)(2,2)(3,3)(1,3)
 \pspolygon[linestyle=none,fillstyle=vlines*,hatchcolor=Apricot,%
            hatchwidth=1pt,fillcolor=Mahogany](2,0)(3,1)(3,3)(2,2)
\end{pspicture}
```

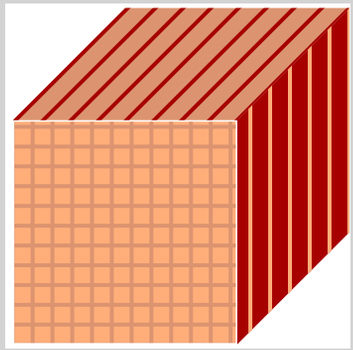


The slant of the lines in the pattern is controlled by the `hatchangle` parameter and its default value is 45 (degrees). The next example shows the effect of changing it.

```
\begin{pspicture}(0,0)(3,3)
  \psframe[linestyle=none,%
           fillstyle=crosshatch*,%
           hatchcolor=Tan,%
           hatchwidth=1pt,%
           hatchangle=90,%
           fillcolor=Apricot]%
          (0,0)(2,2)
  \pspolygon[linestyle=none,%
             fillstyle=hlines*,%
```


The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
              hatchcolor=Mahogany,%
              hatchwidth=1pt,%
              fillcolor=Tan]%
             (0,2)(2,2)(3,3)(1,3)
  \pspolygon[linestyle=none,%
              fillstyle=vlines*,%
              hatchcolor=Apricot,%
              hatchwidth=1pt,%
              hatchangle=180,%
              fillcolor=Mahogany]%
             (2,0)(3,1)(3,3)(2,2)
\end{pspicture}
```



Online LaTeX Tutorial
Part II – Graphics
PSTricks

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

2.4. Custom colors

If you are not satisfied with any of the sixty four named colors, you can define your own colors using the `\definecolor` command. The syntax for this command is

```
\definecolor{name}{model}{spec}
```

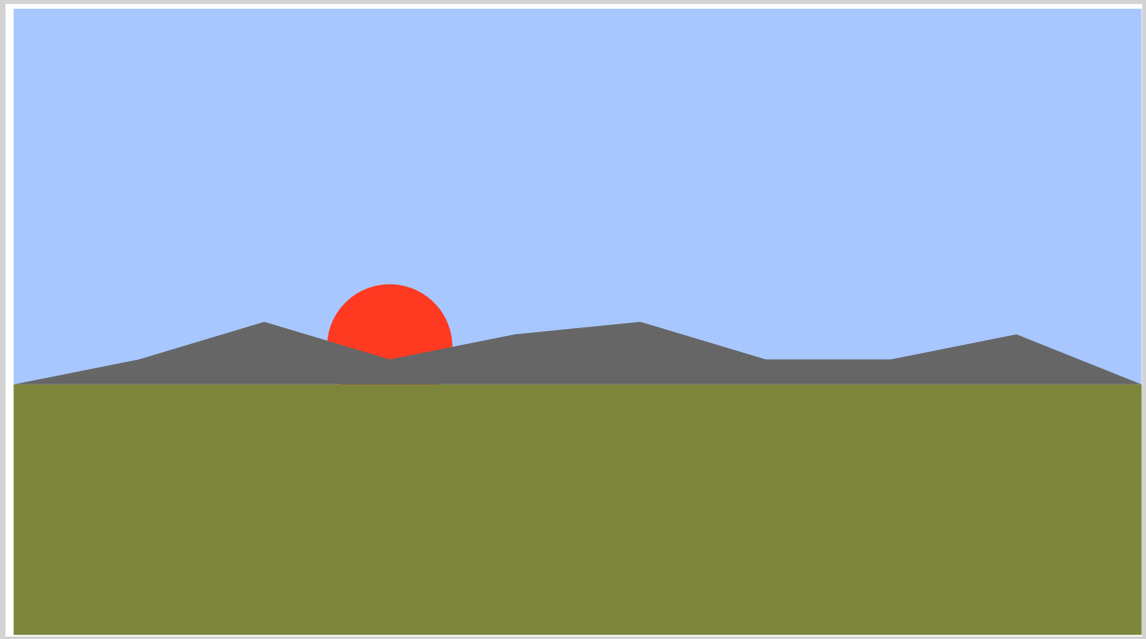
where *name* is the name of the color you want to create, *model* is the scheme of specifying the color such as `rgb`, `cmyk`, `gray` or `named`. For example, see how the colors `myblue`, `mygreen` and `mygray` are used in the code below.

Note especially the definition of `mygray`: different shades of gray from white to black can be created by using the `gray` model and specifying a number between 0 and 1; the larger the number, the lighter the shade with 0 giving black and 1, white.

```
\definecolor{myblue}{rgb}{0.66,0.78,1.00}
\definecolor{mygreen}{rgb}{0.49,0.52,0.23}
\definecolor{mygray}{gray}{0.4}
\begin{pspicture}(0,0)(9,5)
  \psframe[fillstyle=solid,%
           fillcolor=myblue]%
          (0,2)(9,5)
  \pscircle[fillstyle=solid,%
            fillcolor=RedOrange]%
           (3,2.3){0.5}
  \pspolygon[fillstyle=solid,%
             fillcolor=mygray]%
            (0,2)(1,2.2)(2,2.5)%
            (3,2.2)(4,2.4)(5,2.5)%
            (6,2.2)(7,2.2)(8,2.4)(9,2)
```


The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
    \psframe[fillstyle=solid,%
             fillcolor=mygreen]%
            (0,0)(9,2)
  \end{pspicture}
```



Online LaTeX Tutorial
Part II – Graphics
PSTricks

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

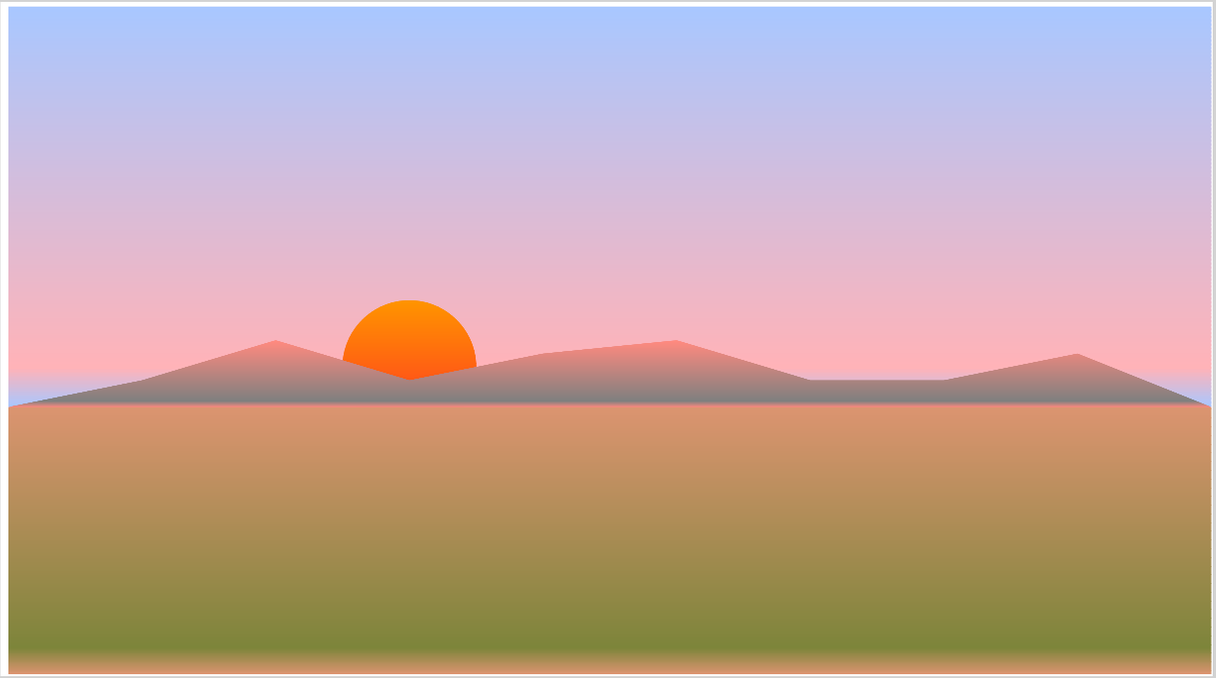
2.5. From one color to another

There's yet another `fillstyle` which is available, if we use the package `pst-grad`. This style is called `gradient` and it allows us to fill a closed region using *two* colors, the color gradually shifting from one to the other. We do this by setting color names to the parameters `gradbegin` and `gradend`. The example below shows how we can add more "effects" to the landscape we'd drawn earlier:

```
\definecolor{myblue}{rgb}{0.66,0.78,1.00}
\definecolor{mypink}{rgb}{1.00,0.70,0.72}
\definecolor{mygreen}{rgb}{0.49,0.52,0.23}
\begin{pspicture}(0,0)(9,5)
  \psframe[linestyle=none,%
           linewidth=0pt,%
           fillstyle=gradient,%
           gradbegin=myblue,%
           gradend=mypink]%
          (0,2)(9,5)
 \pscircle[linestyle=none,%
           linewidth=0pt,%
           fillstyle=gradient,%
           gradbegin=YellowOrange,%
           gradend=RedOrange]%
          (3,2.3){0.5}
\pspolygon[linestyle=none,%
           linewidth=0pt,%
           fillstyle=gradient,%
           gradbegin=Melon,%
           gradend=Gray]%
          (0,2)(1,2.2)(2,2.5)(3,2.2)(4,2.4)%
          (5,2.5)(6,2.2)(7,2.2)(8,2.4)(9,2)
\psframe[linestyle=none,
         linewidth=0pt,
         fillstyle=gradient,
```


The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
            gradbegin=Tan,
            gradend=mygreen]
           (0,0)(9,2)
\end{pspicture}
```



By default, this style of filling starts with the `gradbegin` color from the top, gets to the `gradend` color *near* the bottom and again starts with the `gradbegin` color. (If you look at the picture above closely, you can see that the sky goes from blue to pink and there's a small strip of blue again after the pink. The same thing can be seen in the grass also.) Just where the `gradend` color appears is controlled by the `gradmidpoint` parameter, which can take a number between 0 and 1 as its value. The default value is 0.9. See the effect of setting this to 1 in the picture above:

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{center}
  \definecolor{myblue}{rgb}{0.66,0.78,1.00}
  \definecolor{mypink}{rgb}{1.00,0.70,0.72}
  \definecolor{mygreen}{rgb}{0.49,0.52,0.23}
  \begin{pspicture}(0,0)(9,5)
      \psframe[linestyle=none,linewidth=0pt,%
                fillstyle=gradient,gradbegin=myblue,%
              gradend=mypink,gradmidpoint=1]%
             (0,2)(9,5)
    \pscircle[linestyle=none,%
              linewidth=0pt,%
              fillstyle=gradient,%
              gradangle=0,%
              gradbegin=YellowOrange,%
              gradend=RedOrange]%
             (3,2.3){0.5}
    \pspolygon[linestyle=none,%
               linewidth=0pt,%
               fillstyle=gradient,%
               gradbegin=Melon,%
               gradend=Gray,%
               gradmidpoint=1]%
              (0,2)(1,2.2)(2,2.5)(3,2.2)(4,2.4)%
              (5,2.5)(6,2.2)(7,2.2)(8,2.4)(9,2)
    \psframe[linestyle=none,%
             linewidth=0pt,%
             fillstyle=gradient,%
             gradbegin=Tan,%
             gradend=mygreen,%
             gradmidpoint=1]%
            (0,0)(9,2)
    \psline[linecolor=Tan](0,2)(9,2)
  \end{pspicture}
\end{center}
```

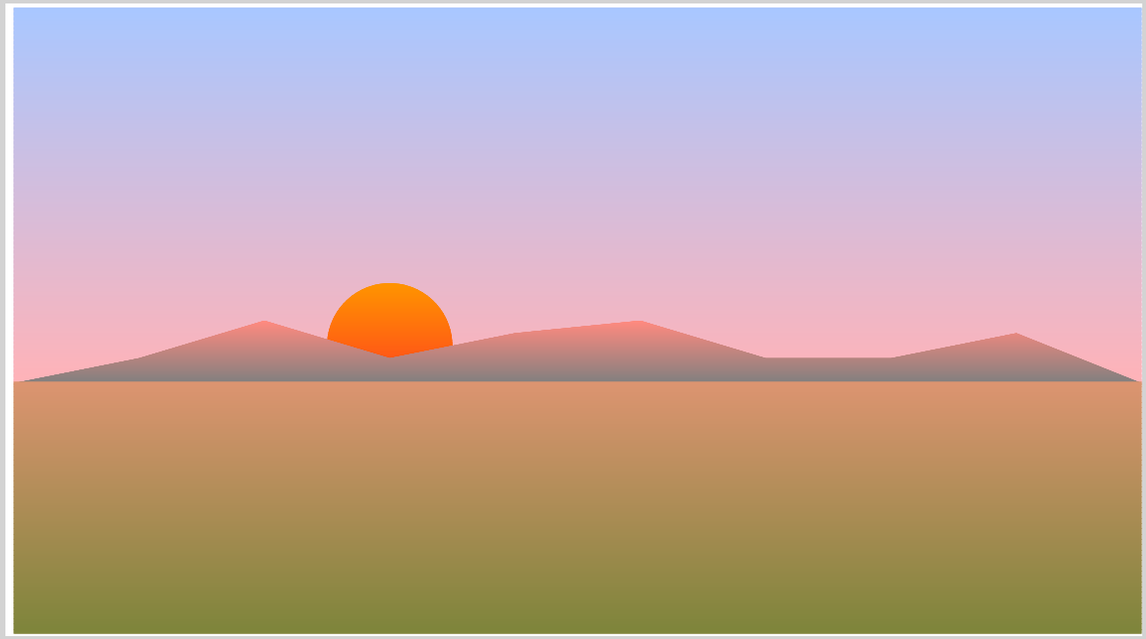
Colorful Tricks

Online LaTeX Tutorial

Part II – Graphics

PSTricks

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



The angle of color transition is set by the parameter `gradengle` with default value 0. The example below shows our landscape with different values for this parameter:

```
\begin{center}
  \definecolor{myblue}{rgb}{0.66,0.78,1.00}
  \definecolor{mypink}{rgb}{1.00,0.70,0.72}
  \definecolor{mygreen}{rgb}{0.49,0.52,0.23}
  \begin{pspicture}(0,0)(9,5)
    \psframe[linestyle=none,%
             linewidth=0pt,%
             fillstyle=gradient,%
             gradangle=350,%
             gradbegin=myblue,%
             gradend=mypink,%
             gradmidpoint=1]%
           (0,2)(9,5)
```


The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
  \pscircle[linestyle=none,%
            linewidth=0pt,%
            fillstyle=gradient,%
            gradangle=0,%
            gradbegin=YellowOrange,%
            gradend=RedOrange]%
            (3,2.3){0.5}
  \pspolygon[linestyle=none,%
             linewidth=0pt,%
             fillstyle=gradient,%
             gradangle=90,%
             gradbegin=Melon,%
             gradend=Gray,%
             gradmidpoint=1]%
             (0,2)(1,2.2)(2,2.5)(3,2.2)(4,2.4)%
             (5,2.5)(6,2.2)(7,2.2)(8,2.4)(9,2)
   \psframe[linestyle=none,%
            linewidth=0pt,%
            fillstyle=gradient,%
            gradangle=10,%
            gradbegin=Tan,%
            gradend=mygreen,%
            gradmidpoint=1]%
            (0,0)(9,2)
  \psline[linecolor=Tan](0,2)(9,2)
\end{pspicture}
\end{center}
```


Online LaTeX Tutorial

Part II – Graphics

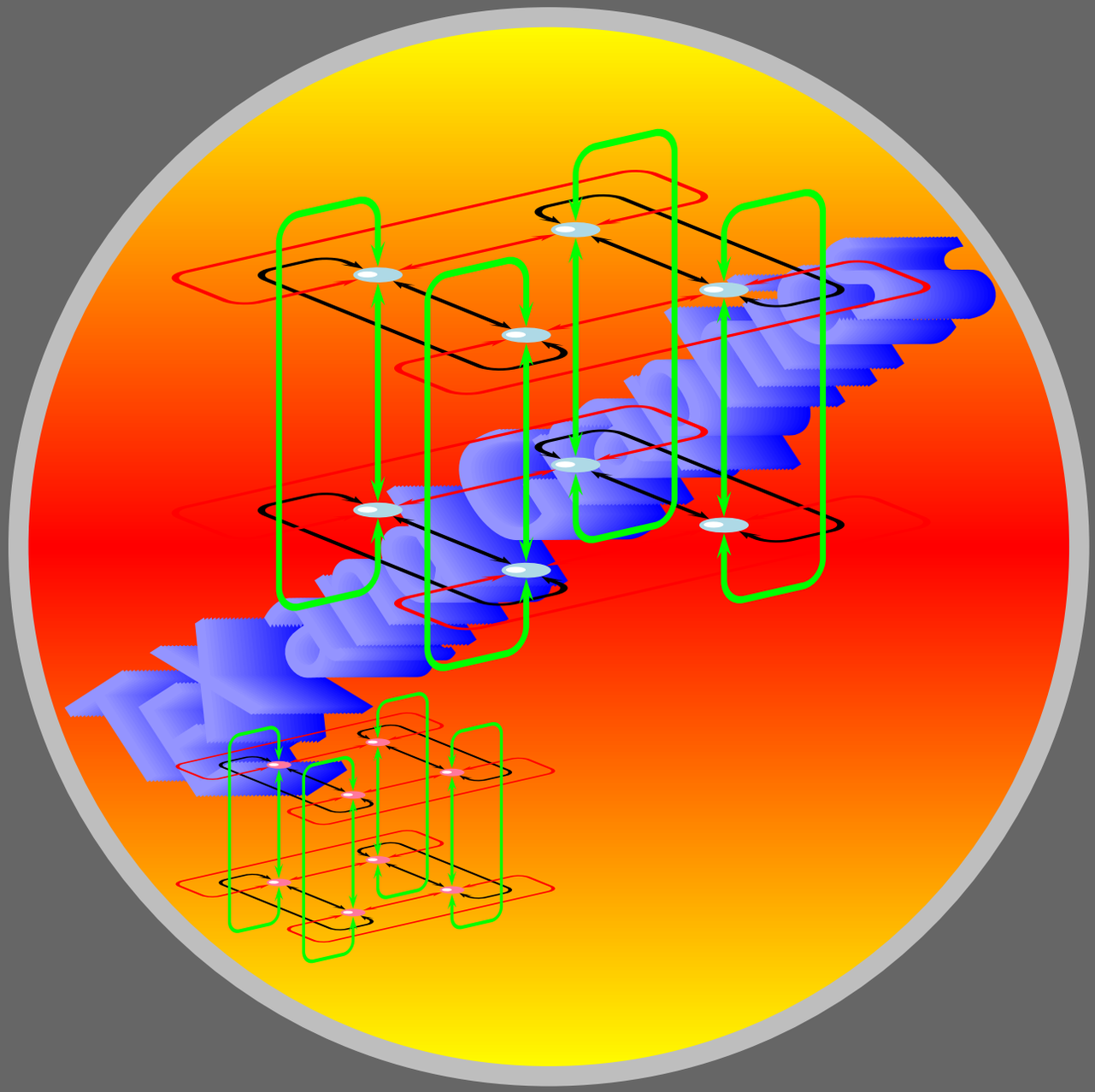
PSTricks

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

With this, we close our discussion on colors. But the general discussion on PSTricks is far from over.

Online LaTeX Tutorial
Part II – Graphics
PSTricks

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



Borderline Tricks

Double boundary
Inside, outside or in the middle?
Borders—visible or invisible
Shadows

Online LaTeX Tutorial

Part II – Graphics

PSTricks

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

3. Borderline Tricks

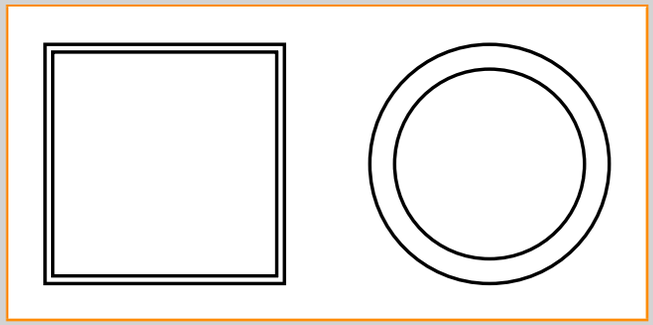
In the first chapter we've seen how we can draw various graphic objects with PSTricks and in the next, we saw how we can add a bit of color to the proceedings. In all these, we've been mostly interested in the *interior* of these objects. In this chapter, we'll see how we can decorate the *boundary*.

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

3.1. Double boundary

In the first chapter, we saw that "double lines" could be drawn by setting the parameter `doubleline` to `true`. This setting also draws the boundary of other graphic objects in double. For example

```
\begin{pspicture}(0,0)(2,2)
  \psframe[doubleline=true]%
          (0,0)(2,2)
\end{pspicture}
\hspace{0.5cm}
\begin{pspicture}(0,0)(2,2)
  \pscircle[doubleline=true,%
            doublesep=5pt]%
           (1,1){1}
\end{pspicture}
```

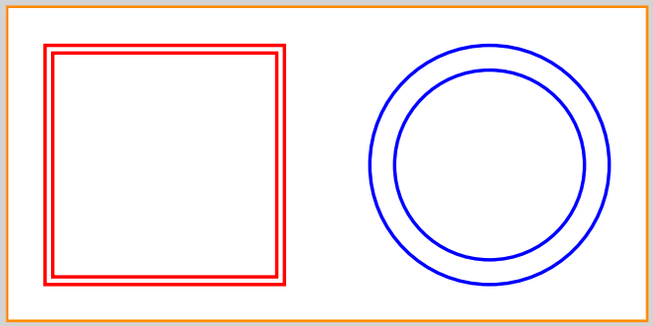


Note that the parameter `doublesep` is used to set the distance between the two lines. Its default value is $1.25 \times$ `linewidth` (remember the parameter `linewidth`?)

The double line can be colored using the `linecolor` parameter as in the example below:

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(0,0)(2,2)
  \psframe[doubleline=true,%
           linecolor=Red]%
          (0,0)(2,2)
\end{pspicture}
\hspace{0.5cm}
\begin{pspicture}(0,0)(2,2)
  \pscircle[doubleline=true,%
            doublesep=5pt,%
            linecolor=Blue]%
           (1,1){1}
\end{pspicture}
```



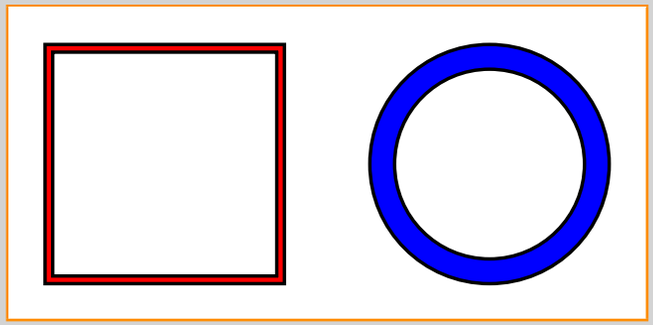
The gap between the two lines of the boundary can be filled with color using the parameter `doublecolor` as in the next example:

```
\begin{pspicture}(0,0)(2,2)
  \psframe[doubleline=true,%
           doublecolor=Red]%
          (0,0)(2,2)
 \end{pspicture}
 \hspace{0.5cm}
 \begin{pspicture}(0,0)(2,2)
   \pscircle[doubleline=true,%
```


Online LaTeX Tutorial
Part II – Graphics
PSTricks

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
            doublesep=5pt,%
            doublecolor=Blue]%
           (1,1){1}
\end{pspicture}
```

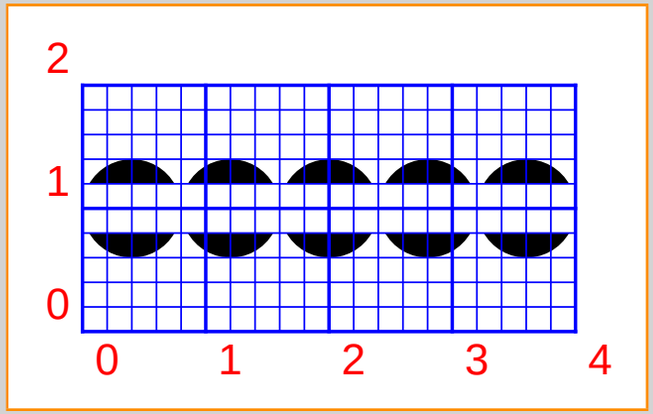


Now something funny happens, if you combine `doubleline=true` with `linestyle=dotted`. Look at the example below:

```
\begin{pspicture}(0,0)(4,0)
  \psline[linestyle=dotted,%
          linewidth=2pt,%
          doubleline=true]%
         (0,0)(4,0)
\end{pspicture}
```

If you look closely, you can see that, instead of two lines of dots as we would expect, we get one line of large dots split down the middle. To understand what really happened, let's consider a larger version of this picture, with a grid beneath for easy measurement:

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



Here, the line is drawn by the command:

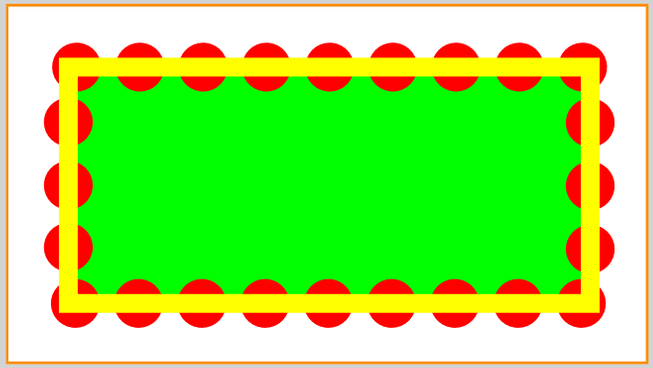
```
\psline[linestyle=dotted,%
          linewidth=2mm,%
          doubleline=true,%
          doublesep=4mm]%
          (0,1)(4,1)
```

and the grid is made up of 2 mm squares (we'll talk about such grids later).. Now we can see that each of the circular segments making up the two lines is 2 mm high (the `linewidth`) and the gap separating them is 4 mm (the `doublesep`). Thus in this case, PSTricks creates a row of dots, each of diameter 8 mm (2 + 4 + 2) and splits them down the middle by a cut 4 mm wide. (Now try to work out the diameter of the dots—before they were split—in our first picture, remembering the default `doublesep` is 1.25 × `\linewidth`.)

We can now use this feature to produce some pretty pictures like

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(0,0)(4,2)
  \psframe[fillstyle=solid,%
           fillcolor=Green,%
           linestyle=dotted,%
           linewidth=3pt,%
           linecolor=Red,%
           doubleline=true,%
           doublecolor=Yellow]%
          (0,0)(4,2)
\end{pspicture}
```



The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

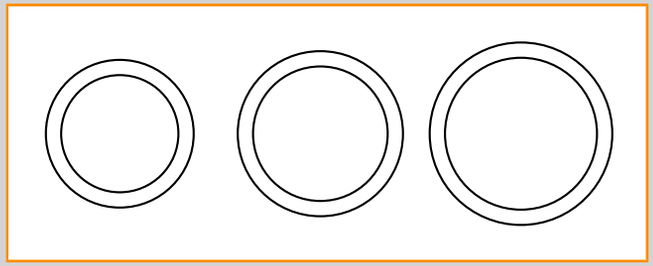
3.2. Inside, outside or in the middle?

When we draw a double boundary for an object, one natural question is whether the dimensions of the object are with reference to the outer or inner boundary. For example, if we specify the radius of a circle as 1 cm and give it a double border, is it the inner circle or the outer circle that has radius 1 cm? By default, it's the outer circle, but it can be changed with the help of the `dimen` parameter. Its possible values are `inner`, `middle` and `outer` and the default value is outer. The example below illustrates this:

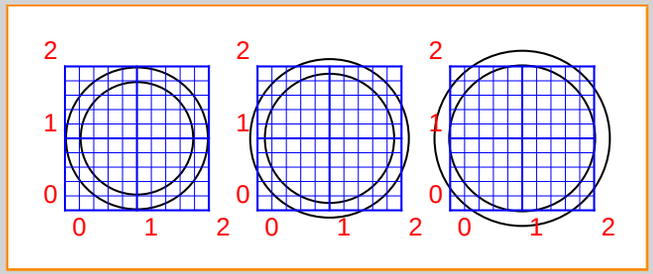
```
\begin{pspicture}(0,0)(2,2)
  \pscircle[doubleline=true,%
            doublesep=5pt,%
            dimen=outer]%
           (1,1){1}
  \end{pspicture}
  \hspace{.5cm}
  \begin{pspicture}(0,0)(2,2)
    \pscircle[doubleline=true,%
              doublesep=5pt,%
              dimen=middle]%
              (1,1){1}
  \end{pspicture}
  \hspace{.5cm}
  \begin{pspicture}(0,0)(2,2)
    \pscircle[doubleline=true,%
              doublesep=5pt,%
              dimen=inner]%
              (1,1){1}
  \end{pspicture}
```

gives

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



(The value `dimen=outer` for the first circle is actually redundant, since by default, the parameter `dimen` is set to `outer`). Perhaps the difference will be better seen if each figure is provided with a coordinate grid underneath as shown below:

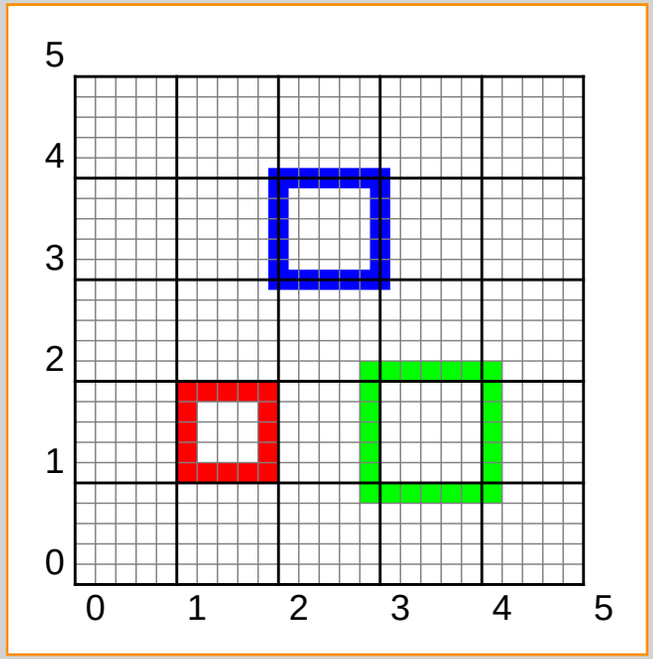


The `dimen` parameter can be applied to such closed graphic objects as `\psframe`, `\pscircle`, `\psellipse` and `\pswedge`, even when `doublelines` is *not* in effect. It then determines whether the measurements refer to the outside, inside or the middle of the boundary. The difference however is noticeable, only for large `linewidth`. The example below illustrates this.

```
\begin{pspicture}(0,0)(5,5)
  \psframe[linewidth=2mm,%
           linecolor=Red,%
           dimen=outer]%
          (1,1)(2,2)
  \psframe[linewidth=2mm,%
           linecolor=Blue,%
```


The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
            dimen=middle]%
          (2,3)(3,4)
  \psframe[linewidth=2mm,%
           linecolor=Green,%
           dimen=inner]%
          (3,1)(4,2)
\end{pspicture}
```



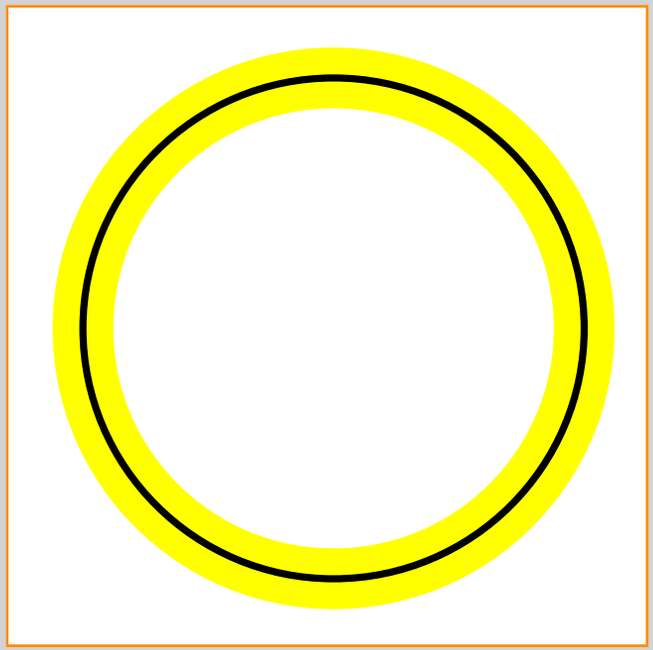
Online LaTeX Tutorial
Part II – Graphics
PSTricks

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

3.3. Borders—visible or invisible

We can put a border around the edge of an object by setting the `border` parameter (default value 0pt) to a positive *length*. The color of the border is set by the parameter `bordercolor`, whose default value is white. For example,

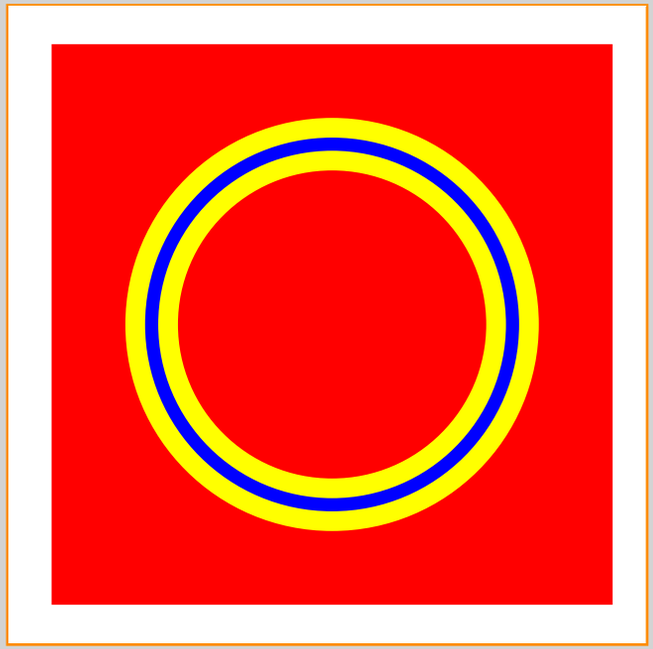
```
\begin{pspicture}(2,0)(3,2)
  \pscircle[border=3pt,%
            bordercolor=Yellow]%
            (2,1){1}
\end{pspicture}
```



Perhaps the edges of a border will be seen better, if its set in a dark background as in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

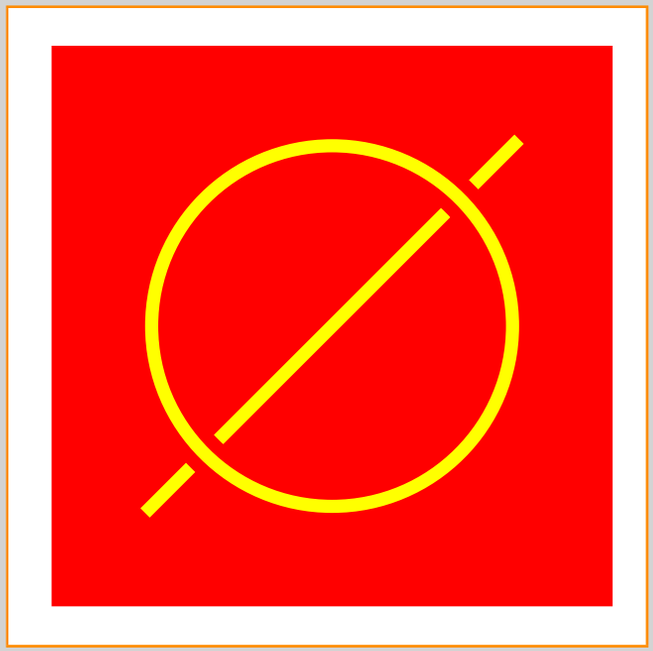
```
\begin{pspicture}(0,0)(3,3)
    \psframe*[linecolor=Red]%
             (0,0)(3,3)
    \pscircle[linewidth=2pt,%
              linecolor=Blue,%
              border=3pt,%
              bordercolor=Yellow]
             (1.5,1.5){1}
 \end{pspicture}
```



An interesting possibility is to make the border color the same as the background color, which makes the border invisible to us, but "seen" by the graphic objects drawn *before* it. This can be used to create the effect of a line *passing over* another, for example. This is illustrated below:

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

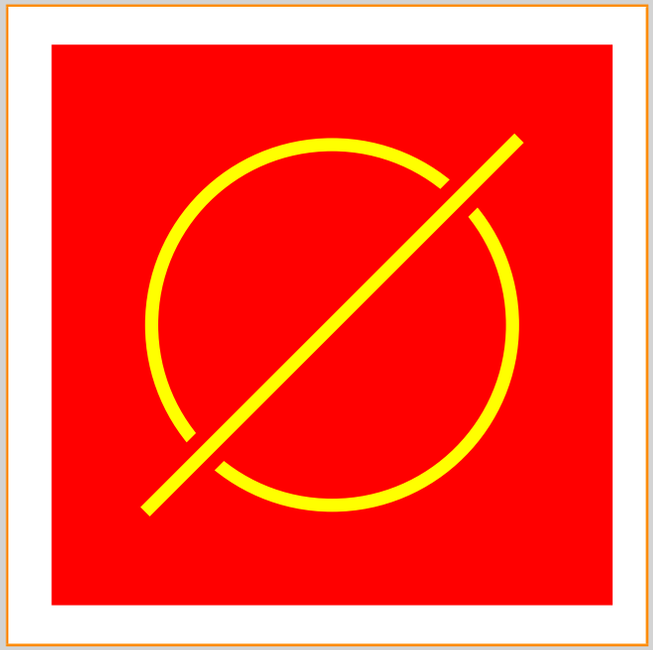
```
\begin{pspicture}(0,0)(3,3)
    \psframe*[linecolor=Red]%
             (0,0)(3,3)
    \psline[linecolor=Yellow,%
            linewidth=2pt]%
           (0.5,0.5)(2.5,2.5)
    \pscircle[linewidth=2pt,%
              linecolor=Yellow,%
              border=2pt,%
              bordercolor=Red]
             (1.5,1.5){1}
\end{pspicture}
```



Note that the circle with the border is placed over the line and the red border blots out pieces of the line. We can reverse this effect by first drawing the circle *without border* and then the line *with border*

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(0,0)(3,3)
  \psframe*[linecolor=Red]%
           (0,0)(3,3)
  \pscircle[linewidth=2pt,%
            linecolor=Yellow]
          (1.5,1.5){1}
  \psline[linecolor=Yellow,%
         linewidth=2pt,%
         border=2pt,%
         bordercolor=Red]%
        (0.5,0.5)(2.5,2.5)
\end{pspicture}
```



Online LaTeX Tutorial

Part II – Graphics

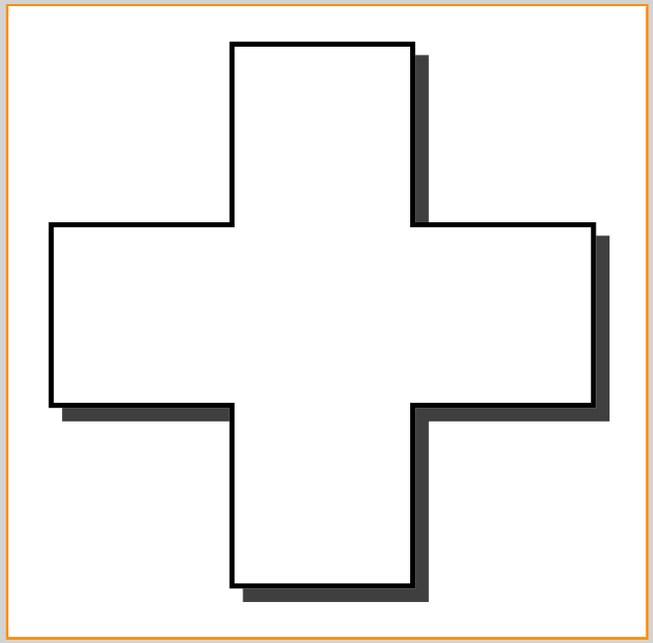
PSTricks

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

3.4. Shadows

An object can be given a shadow, by setting the `shadow` parameter to `true`. (Its default value is `false`.) Look at the example below:

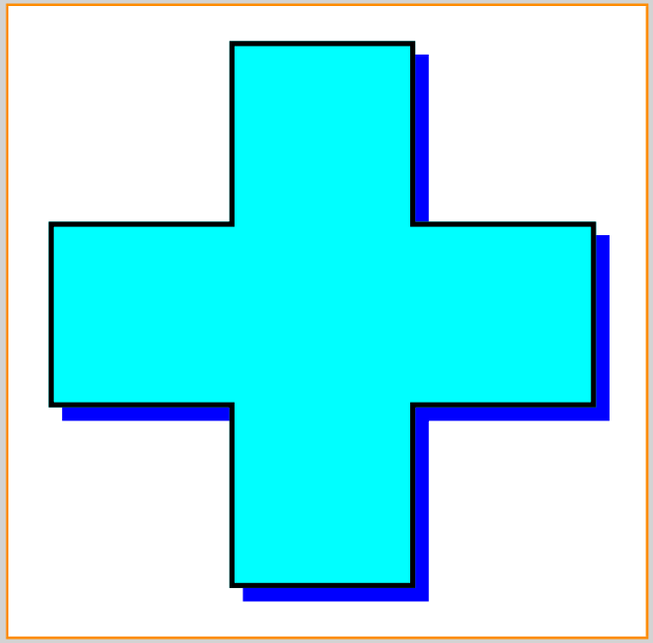
```
\begin{pspicture}(0,0)(3,3)
  \pspolygon[shadow=true]%
            (1,1)(1,0)(2,0)(2,1)
            (3,1)(3,2)(2,2)(2,3)
            (1,3)(1,2)(0,2)(0,1)
\end{pspicture}
```



The color of the shadow is set by the parameter `shadowcolor`, whose default value is `darkgray`.

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

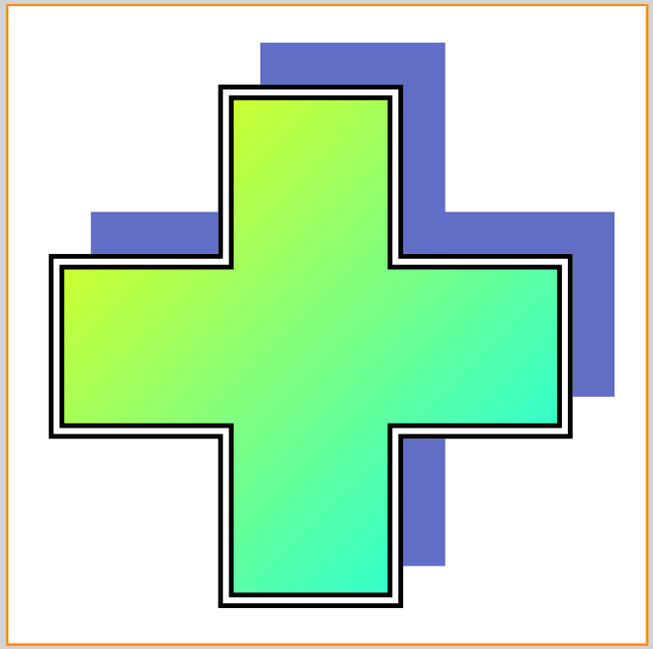
```
\begin{pspicture}(0,0)(3,3)
  \pspolygon[fillstyle=solid,%
             fillcolor=Cyan,%
             shadow=true,%
             shadowcolor=Blue]%
            (1,1)(1,0)(2,0)(2,1)
            (3,1)(3,2)(2,2)(2,3)
            (1,3)(1,2)(0,2)(0,1)
 \end{pspicture}
```



The *size* of the shadow is specified by `shadowsize` (with default value 3 pt). Also, the *position* of the shadow is determined by `shadowangle` which is to be specified as an *angle*. (The default value is -45). These are illustrated in the example below (where we have embellished the original object also with gradient colors and double borders).

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(0,0)(3.5,3.5)
  \pspolygon[fillstyle=gradient,%
             gradbegin=Yellow,%
             gradend=Cyan,%
             gradangle=45,%
             gradmidpoint=1,%
             shadow=true,%
             shadowsize=10pt,%
             shadowangle=45,%
             shadowcolor=CadetBlue,%
             doubleline=true]%
            (1,1)(1,0)(2,0)(2,1)
            (3,1)(3,2)(2,2)(2,3)
            (1,3)(1,2)(0,2)(0,1)
 \end{pspicture}
```



Borderline Tricks

Double boundary
Inside, outside or in the middle?
Borders—visible or invisible
Shadows

Online LaTeX Tutorial

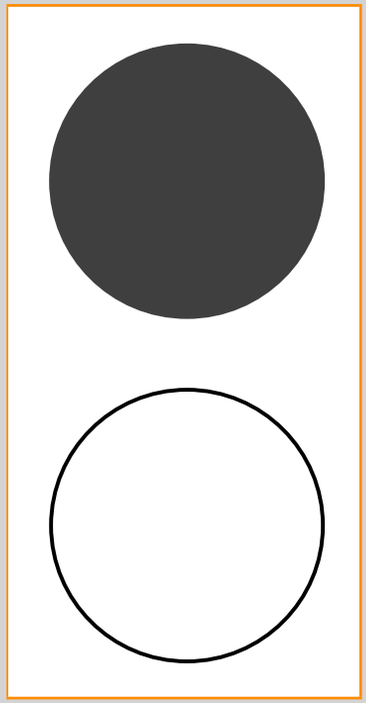
Part II – Graphics

PSTricks

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

By defining the `shadowsize` suitably large, we can detach the shadow from the object, as in the example below:

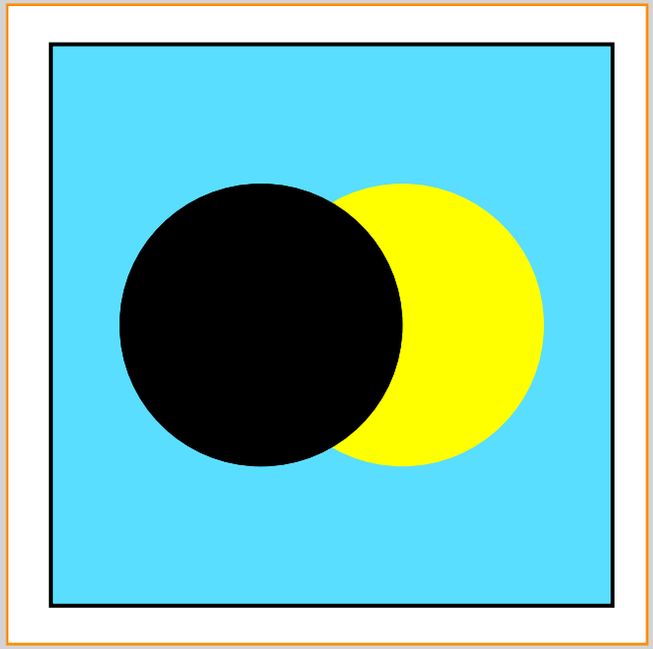
```
\begin{pspicture}(0,0)(2,5)
  \pscircle[shadow=true,%
            shadowsize=2.5cm,%
            shadowangle=90]%
           (1,1){1}
\end{pspicture}
```



Note that the “shadow” never overdraws the original object. But we can create an “eclipse” effect by suitably coloring the object and the shadow as in the example below:

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(0,0)(5,4)
  \psframe[fillstyle=solid,%
           fillcolor=CornflowerBlue]%
          (0,0)(4,4)
  \pscircle[fillstyle=solid,%
            fillcolor=black,%
            shadow=true,%
            shadowsize=1cm,%
            shadowangle=0,%
            shadowcolor=Yellow]
           (1.5,2){1}
\end{pspicture}
```



Borderline Tricks

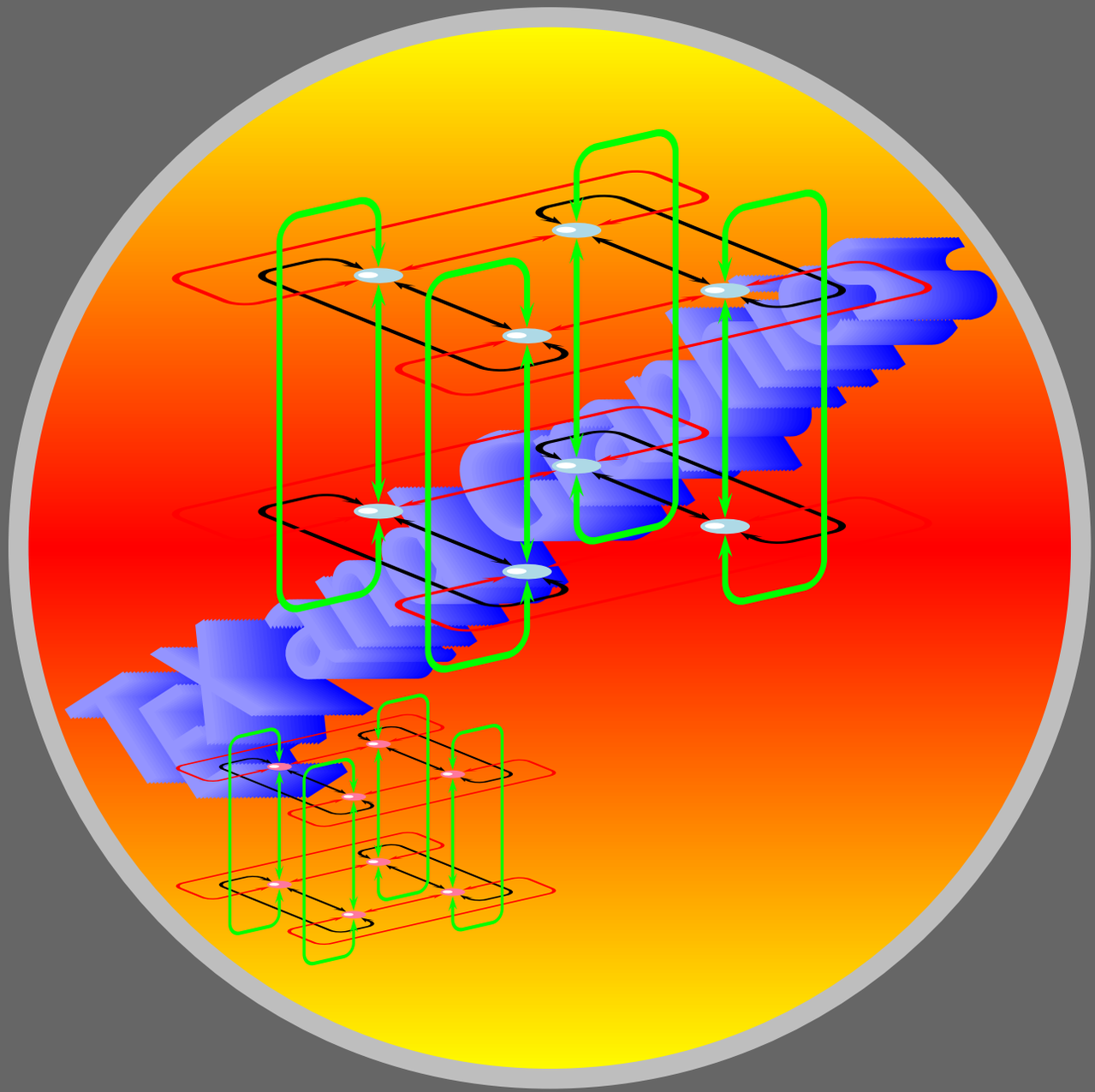
Double boundary
Inside, outside or in the middle?
Borders—visible or invisible
Shadows

Online LaTeX Tutorial

Part II – Graphics

PSTricks

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



Curvy Tricks

Open and closed curves
Invisible ends
Curve tweaking
A new curve

Online LaTeX Tutorial

Part II – Graphics

PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team.
Comments and suggestions may be mailed to
tutorialteam@tug.org.in

This document is generated by PDFTeX with hyperref, pstricks, pdftricks and pdfscreen packages in an intel PC running GNU/LINUX and is released under LPPL

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

4. Curvy Tricks

We've seen how the `\psline` command can be used to form paths and how the `\pspolygon` path can be used to form closed paths joining specified points. Both these commands produce paths made up of pieces of *straight lines*. We can also produce *curves* joining specified points. This chapter explains how.

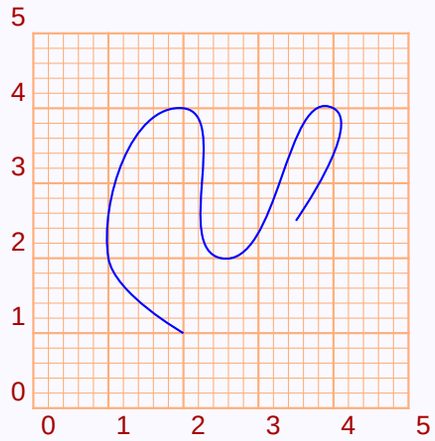
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

4.1. Open and closed curves

To produce an "open" curve joining specified points (analogous to `\psline`), we use the `\pscurve` command. Look at the example below:

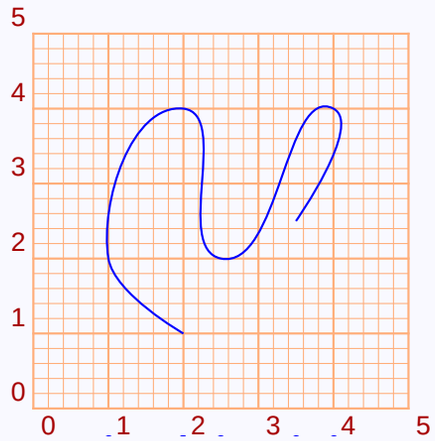
```
\begin{pspicture}(0,0)(5,5)
  \pscurve[linecolor=Blue]%
          (2,1)(1,2)(2,4)%
          (2.5,2)(4,4)(3.5,2.5)
\end{pspicture}
```



As in the earlier examples, the `grid` is *not* drawn by the code shown here.

The points specified can be shown in the picture by setting the parameter `showpoints` to true as shown below:

```
\begin{pspicture}(0,0)(5,5)
  \pscurve[linecolor=Blue,%
           showpoints=true]%
          (2,1)(1,2)(2,4)%
          (2.5,2)(4,4)(3.5,2.5)
\end{pspicture}
```

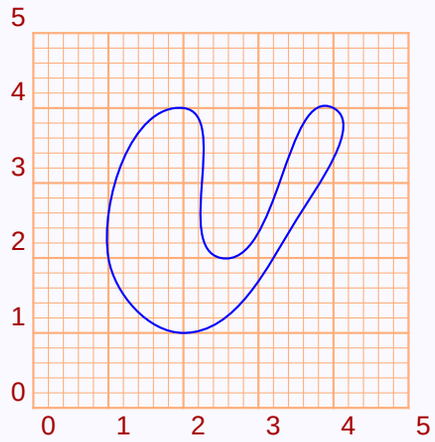


E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

A "closed" curve joining specified points is produced by the \psccurve. (Note the extra c in the middle. It stands for "closed"). The same points in the above example are used to form a closed curve in the next example:

```
\begin{pspicture}(0,0)(5,5)
  \psccurve[linecolor=Blue]%
           (2,1)(1,2)(2,4)%
           (2.5,2)(4,4)(3.5,2.5)
\end{pspicture}
```

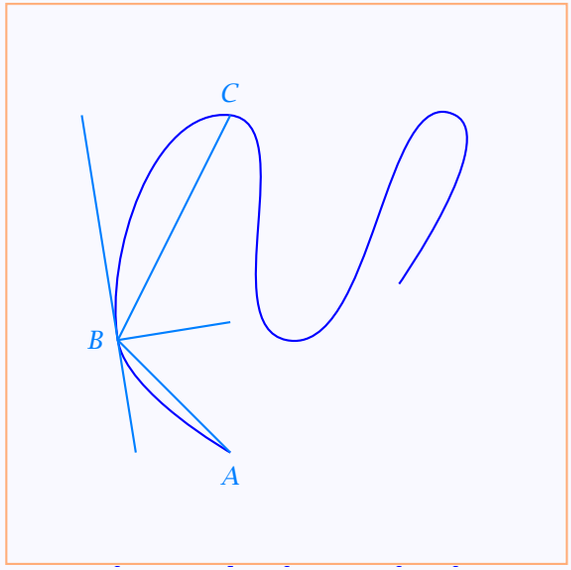


As we know, we can draw infinitely many curves through a set of specified points. So, what's the peculiarity of the curve that \pscurve (or \psccurve) produces? Well, it's like this: if A, B, C are three *consecutive* points of the specified set, then the curve is drawn such that at B (the middle point), the curve (or more precisely, the tangent to the curve) is perpendicular to the bisector of angle ABC. Perhaps this is better described by a picture. The picture is a magnified version of the open curve we've drawn above with the first three points marked as A, B, C and showing the bisector of angle ABC and its perpendicular at B (which you can see is the tangent to the curve at C).

Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



Curvy Tricks

Online LaTeX Tutorial
Part II – Graphics
PSTricks

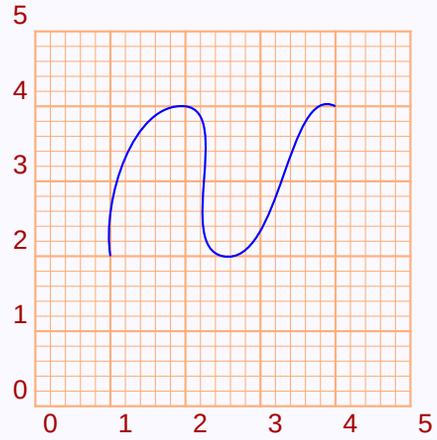
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team.
Comments and suggestions may be mailed to
tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

4.2. Invisible ends

there's a third command to form a curve, namely, `\psecurve`, which draws a curve through the specified points *excluding the first and the last* (The `e` in the middle suggests "endpoints" . Thus the same set of points we used above joined using this command gives the picture below:

```
\begin{pspicture}(0,0)(5,5)
  \psecurve[linecolor=Blue]%
           (2,1)(1,2)(2,4)%
           (2.5,2)(4,4)(3.5,2.5)
\end{pspicture}
```

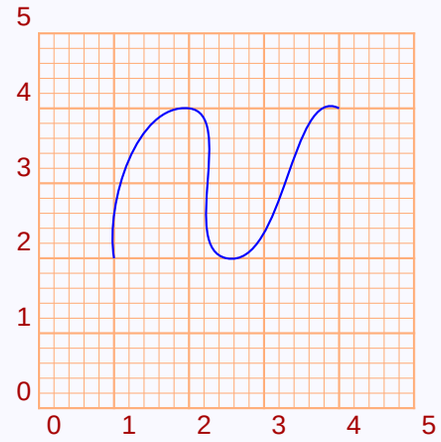


Now what's the use of specifying some points, if you are not using them in the picture? The fact is, though `\psecurve` does not *draw* the curve through the endpoints, it does *see* them. The invisible end points determine the *shape* of the curve at the visible extremities. Look at the example below:

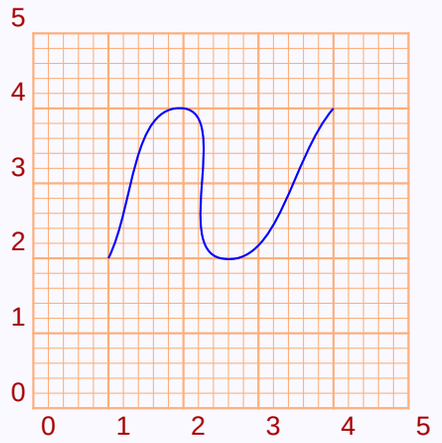
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(0,0)(5,5)
  \psecurve[linecolor=Blue]%
           (2,1)(1,2)(2,4)%
           (2.5,2)(4,4)(3.5,2.5)
\end{pspicture}
```




```
\begin{pspicture}(0,0)(5,5)
  \psecurve[linecolor=Blue]%
           (0.5,1)(1,2)(2,4)%
           (2.5,2)(4,4)(4.5,4.5)
\end{pspicture}
```

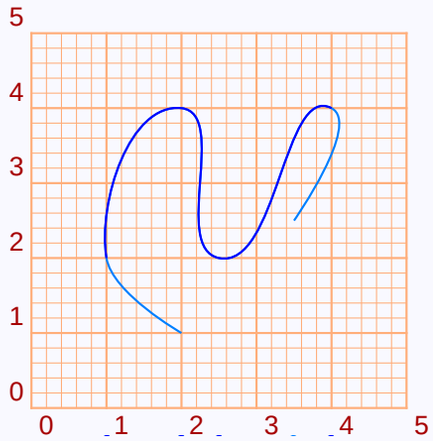


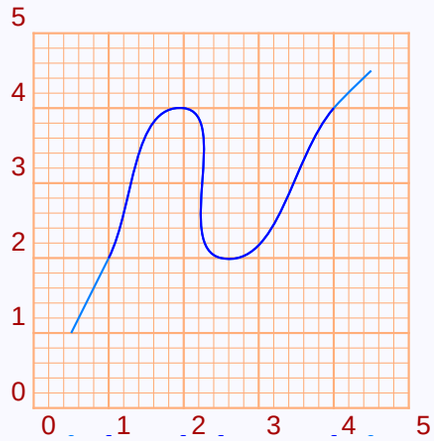
Perhaps the influence of the invisible endpoints is better illustrated, if we draw both the \pscurve and \psecurve together. The next picture does this for the two sets of points in the above example, where the the portion of the \pscurve not drawn by \psecurve is shown in a lighter color.

Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in





Curvy Tricks

Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

4.3. Curve tweaking

The \psecurve is one way of changing the default shape of the curve that \pscurve produces. There are also ways to change the "tautness" of the curve and also its "slope". This is done by changing the values of the curvature parameter. Its value is a triplet of numbers, with the "useful" values of each ranging from −1 to 2. We specify the values by setting

curvature=*number1 number2 number3*

The default set of values for these three numbers are 1, 0.1 and 0. We'll look at each of these values separately to see what each signifies.

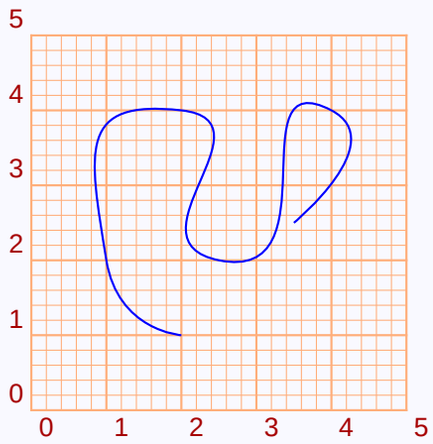
The first number of curvature determines the tautness of the curve. As mentioned above, its default value is 1. Lower values make the curve tighter and higher values makes it looser. The pictures below, with the values of the curvature given underneath, illustrates this. The first picture for example is produced by

```
\pscurve[linecolor=Blue,%
         showpoints=true,%
         curvature=2 0.1 0]%
        (2,1)(1,2)(2,4)%
        (2.5,2)(4,4)(3.5,2.5)
```

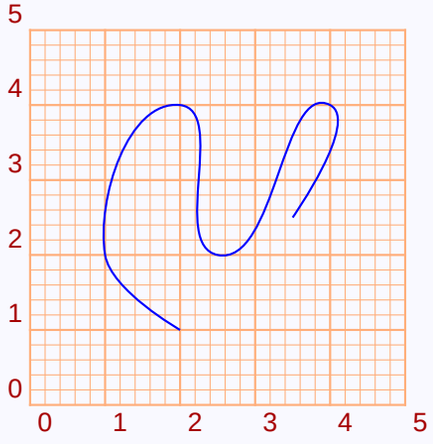
Note that *only* the first number in the curvature is changed in these four curves, the others retaining their default values. The value 0 for this number makes \pscurve join every pair of points with straight lines (so that it becomes \psline) and further tightening through higher values bents it out of shape.

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

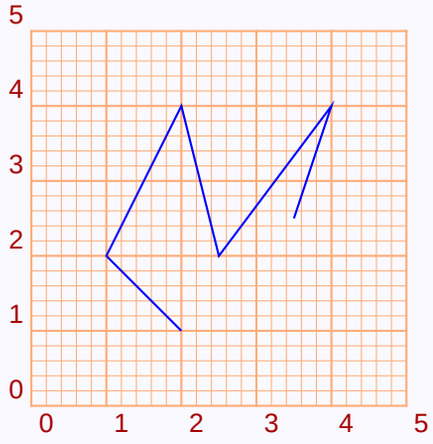
The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



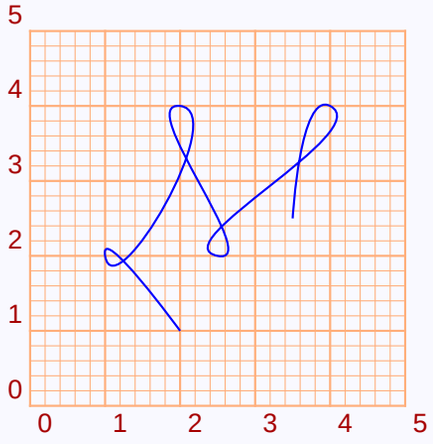
curvature = 2 0.1 0



curvature = 1 0.1 0 (Default)



curvature = 0 0.1 0



curvature = −1 0.1 0

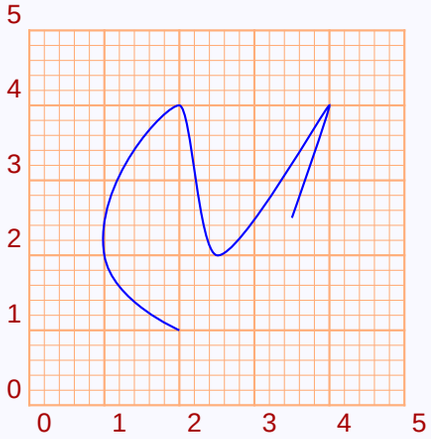
The second number of `curvature` is again for tightening or loosening the curve, but only the portions between certain pairs of points. Suppose A, B, C are three consecutive points specified. If angle ABC is less than 45 degrees, then lower values for this second number makes the curve tighter *around* B, and higher values makes it looser around B. If the angle is greater than 45 degrees, it does the reverse. The next four pictures illustrate this behavior for the set of points we have been using in all our examples. Note that for these points the angle described above is greater than 45 degrees

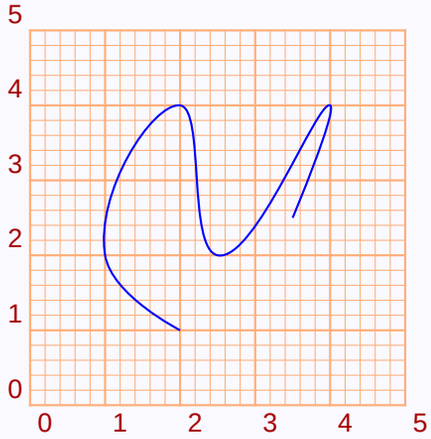
Online LaTeX Tutorial
Part II – Graphics
PSTricks

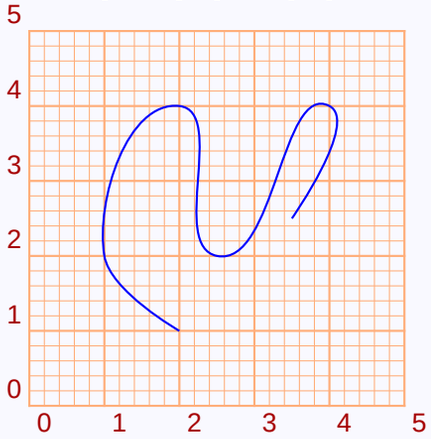
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

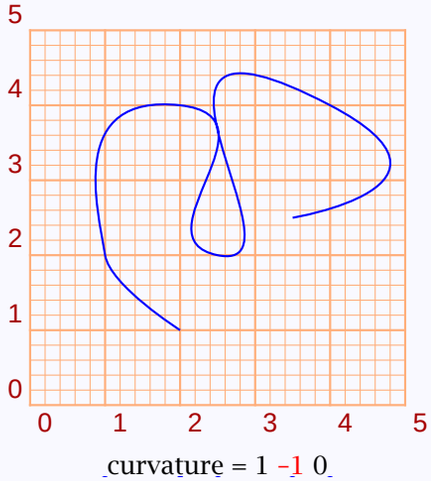
The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

only at the second point.









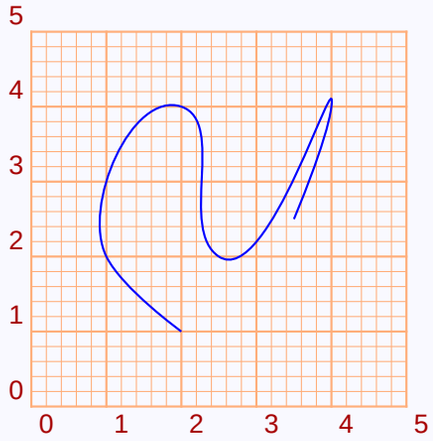
Note how the curve becomes increasingly taut at the second point and increasingly slack elsewhere as the second number decreases.

The third value of `curvature` controls the "slope" of the curve at each point—that is, the angle the tangent at each point makes with the horizontal. We had mentioned earlier that for the curve produced by `\pscurve`, if A, B, C are three consecutive points, then the tangent at B is perpendicular to the bisector of angle ABC. Actually, this is true only for the default

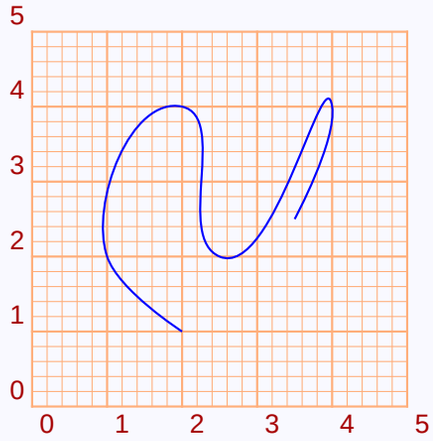
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

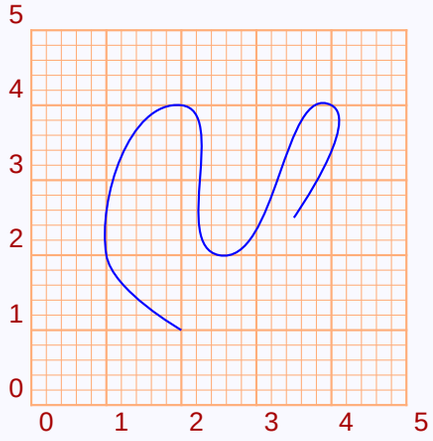
value of 0 for the third number in `curvature`. The value −1, for example, makes the tangent at B parallel to the line AC. The pictures below show the changes in the curve produced by varying this number.



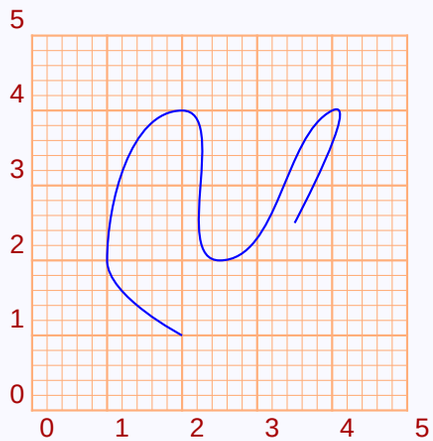
curvature = 1 0.1 2



curvature = 1 0.1 1



curvature = 1 0.1 0 (Default)

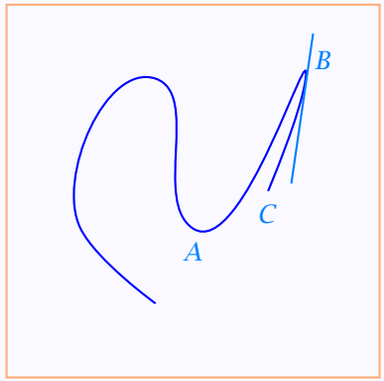


curvature = 1 0.1 −1

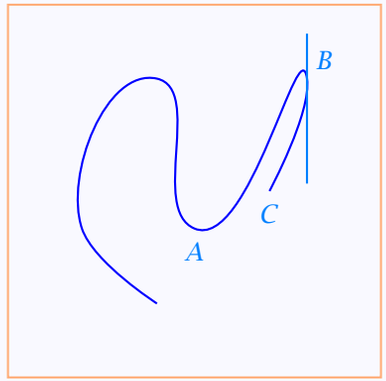
Perhaps, the effect will be better seen, if we draw the tangents also at some point. We do this in the next four pictures, taking the last three points as a sample:

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to `tutorialteam@tug.org.in`

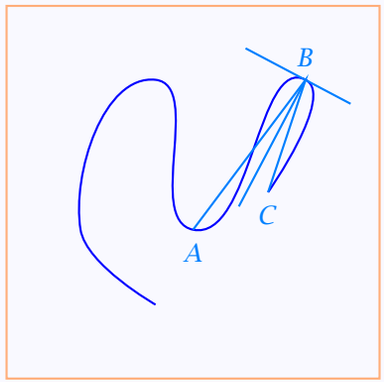
The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
`http://www.tug.org.in`



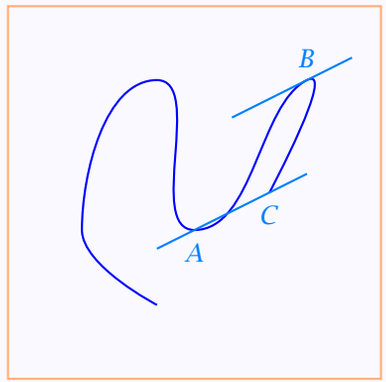
curvature = 1 0.1 2



curvature = 1 0.1 1



curvature = 1 0.1 0 (Default)



curvature = 1 0.1 -1

Note how decreasing the third number rotates the tangent at *B* counter-clockwise.

Online LaTeX Tutorial
Part II – Graphics
PSTricks

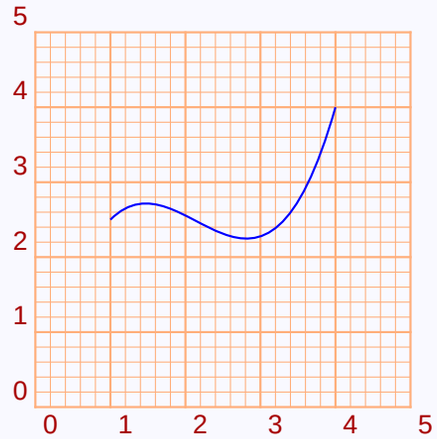
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

4.4. A new curve

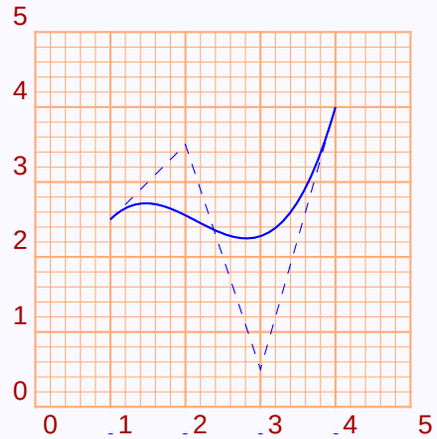
Another type of curve we can draw with PSTricks is the so called Bézier curve (named after Pierre Bézier, who realized their importance in computer-aided design). This is done by the command `\psbezier` command and we must specify four "control" points. Look at the example below:

```
\begin{pspicture}(0,-1)(5,6)
  \psbezier[linecolor=Blue]%
           (1,2.5)(2,3.5)%
           (3,0.5)(4,4)
\end{pspicture}
```



Let's try `showpoints=true` on this.

```
\begin{pspicture}(0,-1)(5,6)
  \psbezier[linecolor=Blue,%
            showpoints=true]%
           (1,2.5)(2,3.5)%
           (3,0.5)(4,4)
\end{pspicture}
```



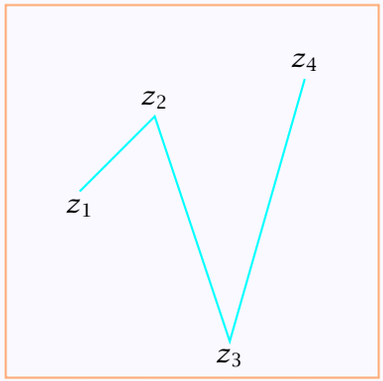
Online LaTeX Tutorial
Part II – Graphics
PSTricks

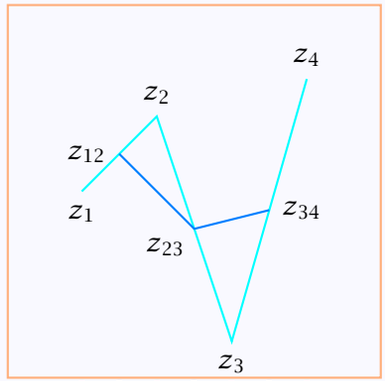
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

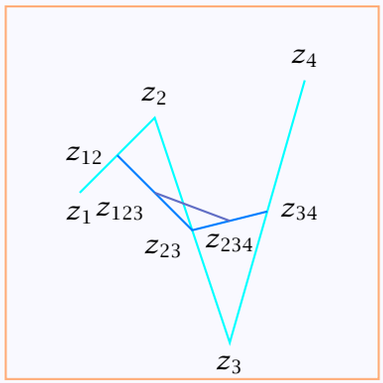
You can see that the curve starts at the first point specified, tangent to the line joining this point to the second and ends at the last point, tangent to the the line joining this point and the third. The second and third points, in addition to determining the tangency at the endpoints, also exert a "pull" on the curve. Let's see in detail how the curve is actually drawn.

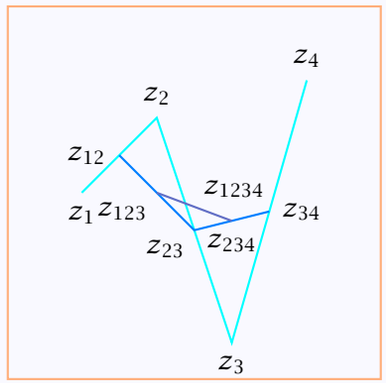
Suppose the control points are z_1, z_2, z_3, z_4. First they are joined by straight lines and the midpoints of each segment is found. Let's call them z_{12}, z_{23} and z_{34}





These three points are again joined by straight lines and then the midpoints of the two line segments are found. Let them be z_{123} and z_{234}. These two points are joined by a straight line and the mid point of this line segment is found.

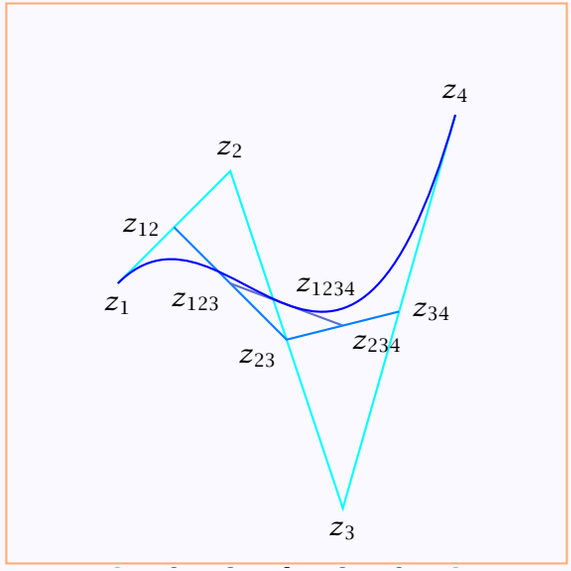




E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

This point, which we denote by z_{1234} is a point on the curve as can be seen in the next (enlarged) picture.



How about the other points of the curve? Well, the process we've described above is repeated with the new control points $z_1, z_{12}, z_{123}, z_{1234}$ and $z_{1234}, z_{234}, z_{34}, z_4$ to get another point and so on *ad infinitum*. This process converges rather quickly and the scaffolding is finally removed.[1]

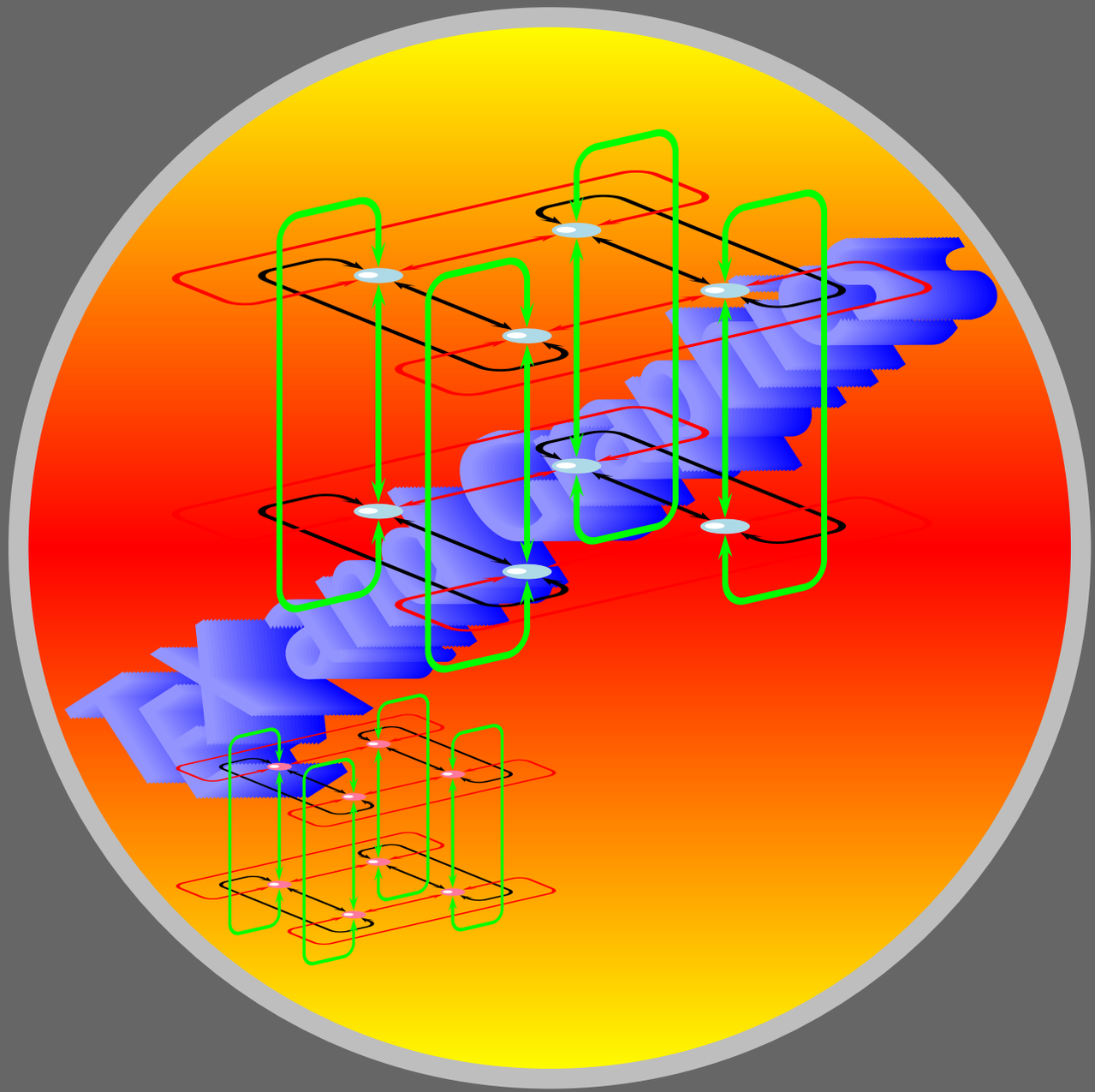
[1]For the mathematically inclined, it may be interesting to note that the curve can be algebraically described by the equation

$$z(t) = (1-t)^3 z_1 + 3(1-t)^2 t z_2 + 3(1-t)t^2 z_3 + t^3 z_4$$

a *Bernshteĭn polynomial* of degree 3.

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



More on Coordinates

Online LaTeX Tutorial

Part II – Graphics

PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

This document is generated by PDFTeX with hyperref, pstricks, pdftricks and pdfscreen packages in an intel PC running GNU/LINUX and is released under LPPL

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

5. More on Coordinates

We have seen that in PSTricks, everything is done with coordinates. We now take a closer look at coordinates and see how we can track and manipulate them. It maybe a good idea to glance back at the first chapter, where we've discussed coordinates in some detail.

Online LaTeX Tutorial

Part II – Graphics

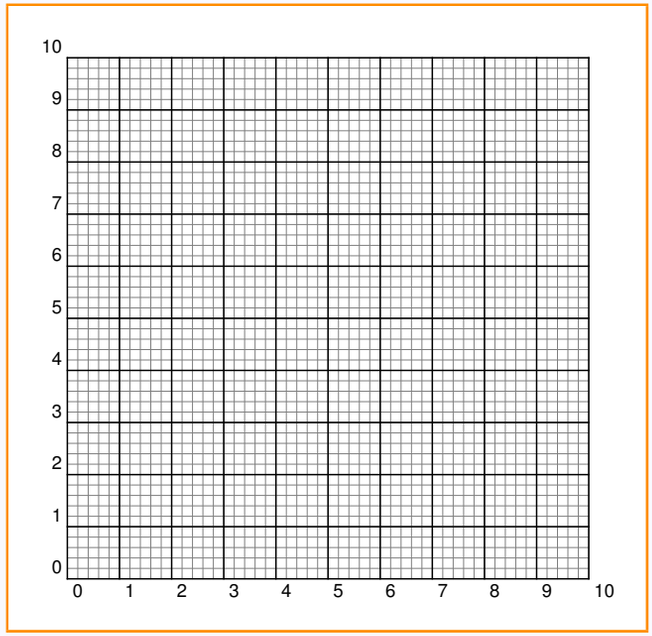
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

5.1. Coordinate grids

To position objects where we want in a picture, we must specify the coordinates. Thus we must imagine an invisible "coordinate grid" (that is, a "graph paper") underlying our picture. But it'd be nice to have the coordinate grid visible, when we first draw a picture. The command `\psgrid` draws such a grid for us; by default, this command draws a 10×10 grid as shown below:



The dimensions of the grid and the positioning of the numbers denoting the intervals can be controlled by specifying coordinates: thus

$$\texttt{\textbackslash psgrid}(x_0, y_0)(x_1, y_1)(x_2, y_2)$$

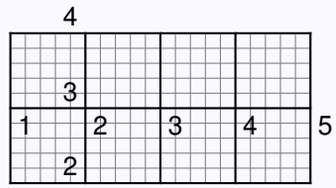
produces a grid with (x_1, y_1) and (x_2, y_2) as opposing corners, and the numbers denoting the x-coordinates running along the line with y-coordinate y_0 and the numbers denoting the y-coordinates running along the line with x-coordinate x_0. Maybe the idea is better understood by an

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

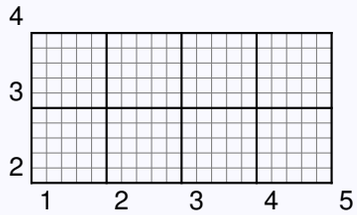
example:

```
\psgrid(2,3)(1,2)(5,4)
```



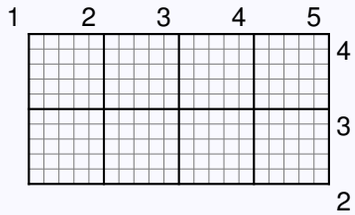
If we specify only *two* pairs of coordinates in a `\psgrid` command, then these are used for opposing corners of the grid and the first pair is used for positioning the numbers, as can be seen from the next example:

```
\psgrid(1,2)(5,4)
```



Note also that the position of the labels with respect to the reference lines (left/right, above/below) is determined by the *order* of specifying the corners. Compare the above example with the one below:

```
\psgrid(5,4)(1,2)
```

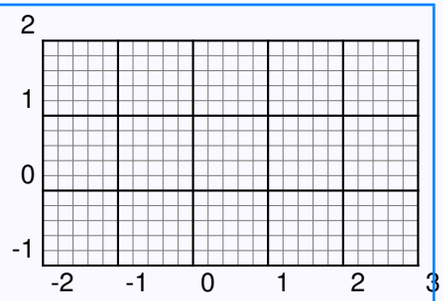


Within a `pspicture` environment, the command `\psgrid` without any coordinates specified, uses the coordinates of the `pspicture`, as shown below:

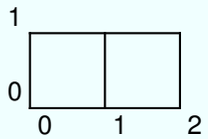
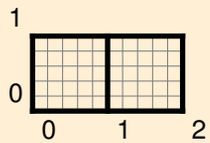
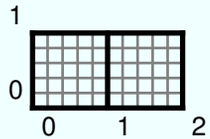
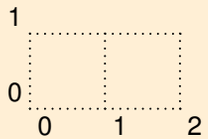
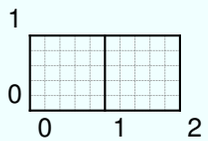
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(-2,-1)(3,2)
  \psgrid
\end{pspicture}
```



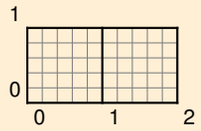
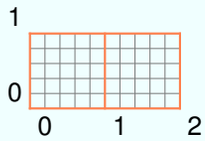
There are various parameters which control the look of the grid which can be tweaked to produce custom grids. See the table below:

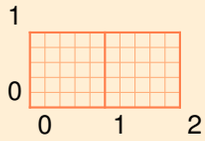
PARAMETER	MEANING	DEFAULT	EXAMPLE
`subgriddiv`	The number of subdivisions of the main grid	5	`\psgrid[subgriddiv=1]% (0,0)(2,1)` 
`gridwidth`	The width of lines in the main grid	0.8 pt	`\psgrid[gridwidth=2pt]% (0,0)(2,1)` 
`subgridwidth`	The width of lines in the subgrid	0.4 pt	`\psgrid[gridwidth=2pt,% subgridwidth=1pt]% (0,0)(2,1)` 
`griddots`	If this number is positive, then the main grid lines are dotted, with that many dots per division	0	`\psgrid[griddots=10,% subgriddiv=1]% (0,0)(2,1)` 
`subgriddots`	If this number is positive, then the subgrid lines are dotted, with that many dots per division	0	`\psgrid[subgriddots=10]% (0,0)(2,1)` 

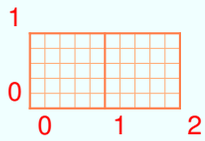
Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

PARAMETER	MEANING	DEFAULT	EXAMPLE
`gridlabels`	Font size of the numbers used to mark the grid	10 pt	`\psgrid[gridlabels=8pt]%` `(0,0)(2,1)` 
`gridcolor`	The color of the main grid lines	black	`\psgrid[gridcolor=Peach]%` `(0,0)(2,1)` 

`subgridcolor`	The color of the subgrid lines	black	`\psgrid[gridcolor=Peach,%`
			`subgridcolor=Apricot]%`
			`(0,0)(2,1)` 

`gridlabelcolor`	The color of the numbers used to mark the grid	black	`\psgrid[gridcolor=Peach,%`
			`subgridcolor=Apricot,%`
			`gridlabelcolor=Red]%`
			`(0,0)(2,1)` 

Another important parameter for `\psgrid` is `unit`. Since this parameter affects not only `psgrid`, but the entire picture, we'll consider it separately.

Online LaTeX Tutorial
Part II – Graphics
PSTricks

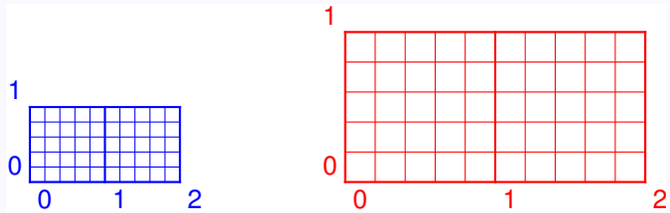
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to `tutorialteam@tug.org.in`

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
`http://www.tug.org.in`

5.2. Changing size

We've mentioned somewhere in the first chapter that the default unit in PSTricks is 1 cm, so that a point specified by $(2,3)$ is 2 centimeters away from the y-axis and 3 centimeters away from the x-axis. This can be changed by setting the `unit` parameter as in the example below:

```
\begin{pspicture}(0,0)(2,1)
  \psgrid[gridcolor=Blue,%
          subgridcolor=Blue,%
          gridlabelcolor=Blue]
\end{pspicture}
\hspace{2cm}
\begin{pspicture}(0,0)(2,1)
  \psgrid[unit=2cm,%
          gridcolor=Red,%
          subgridcolor=Red,%
          gridlabelcolor=Red]
\end{pspicture}
```



This can be used to "scale" a picture as illustrated in the next example:

```
\begin{pspicture}(0,0)(3,2)
  \pspolygon[linecolor=Blue]%
            (0,0)(2,0)(1,1)
\end{pspicture}
\hspace{2cm}
\begin{pspicture}(0,0)(3,2)
  \pspolygon[unit=1.5cm,%
            linecolor=Red]%
            (0,0)(2,0)(1,1)
\end{pspicture}
```

Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

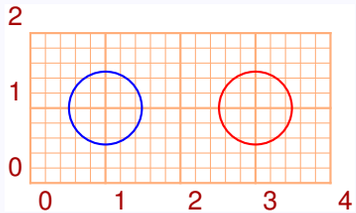
The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

Instead of scaling by the *same* amount horizontally and vertically, we can have unequal scaling by setting the `xunit` and `yunit` separately, as shown below:

```
\begin{pspicture}(0,0)(3,2)
  \pspolygon[linecolor=Blue]%
            (0,0)(2,0)(1,1)
\end{pspicture}
\hspace{2cm}
\begin{pspicture}(0,0)(3,2)
  \pspolygon[xunit=1.5cm,%
             yunit=0.5cm,%
             linecolor=Red]%
             (0,0)(2,0)(1,1)
\end{pspicture}
```

Note that the `xunit` and `yunit` settings do not affect the *radius* of circles (but they do affect the center) as illustrated below:

```
\begin{pspicture}(0,0)(4,2)
  \psgrid{gridcolor=Apricot,%
          gridlabelcolor=Mahogany,%
          subgridcolor=Apricot}
  \pscircle[linecolor=Blue](1,1){0.5}
  \pscircle[xunit=1.5cm,%
            yunit=0.5cm,%
            linecolor=Red]%
            (2,2){0.5}
 \end{pspicture}
```



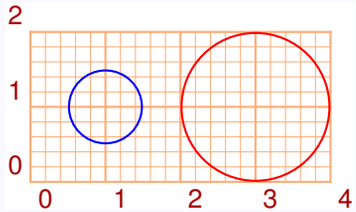
The radius can also be scaled by setting the “runit” parameter as in the next example:

Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(0,0)(4,2)
  \psgrid{gridcolor=Apricot,%
          gridlabelcolor=Mahogany,%
          subgridcolor=Apricot}
  \pscircle[linecolor=Blue](1,1){0.5}
  \pscircle[xunit=1.5cm,%
            yunit=0.5cm,%
            runit=2cm,%
            linecolor=Red]%
            (2,2){0.5}
 \end{pspicture}
```



Note that the parameter `unit` controls `xunit`, `yunit` *and* `runit`.

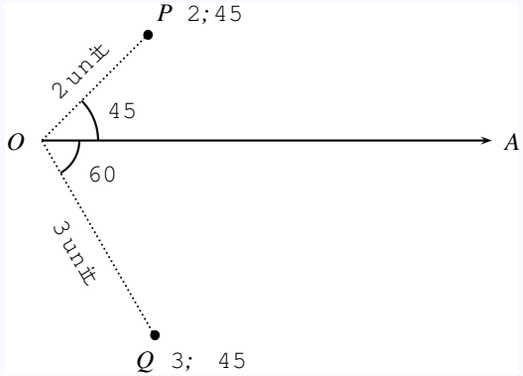
Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

5.3. Another type of coordinates

The Cartesian (or is it Fermatian?) method of using distances from two reference lines is not the only way of labeling points in a plane. Another device mathematicians use is to fix a a point O and a line OA through it and then label each point P by the distance OP and the angle AOP as shown below:



If the distance OP is equal to r and $\angle AOP$ is equal to θ, then r and θ are said to be the *polar coordinates* of P and P is labeled (r, θ). Thus in the picture above, P has polar coordinates $(2, 45)$ and Q has polar coordinates $(3, -60)$. Note that Q can also be represented as $(3, 300)$ (and P as $(2; 405)$, for that matter).

We can specify points using polar coordinates in PSTricks, by invoking the command

```
\SpecialCoor
```

Polar coordinates are specified as `(r;a)` where `r` is the distance and `a` is the angle. (Note that the separator is a *semicolon* and not a comma as in Cartesian coordinates.)

Polar coordinates are very convenient in certain contexts. Look at the example below:

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

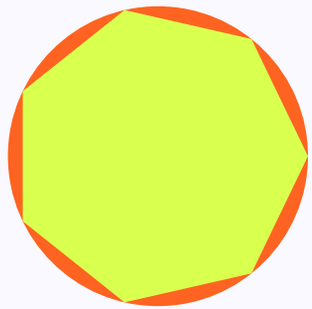
```
\begin{pspicture}(-2,-2)(2,2)
  \pscircle*[linecolor=Lavender](0,0){2}
  \SpecialCoor
  \pspolygon*[linecolor=CornflowerBlue]%
       (2;234)(2;90)(2;306)(2;162)(2;18)
\end{pspicture}
```

By default, angles in polar coordinates are to be specified in degrees; but this can be changed by the command

`\degrees[`*number*`]`

where *number* is the number of parts into which the circle is divided. Thus for example, a regular heptagon can be easily drawn (without calculating the actual angles), by specifying `\degrees[7]`, as in the example below:

```
\begin{pspicture}(-2,-2)(2,2)
  \pscircle*[linecolor=Orange](0,0){2}
  \SpecialCoor
  \degrees[7]
  \pspolygon*[linecolor=GreenYellow]%
   (2;1)(2;2)(2;3)(2;4)(2;5)(2;6)(2;7)
\end{pspicture}
```



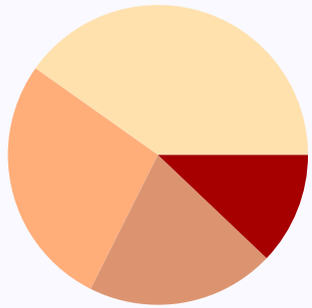
The command `\degrees` can be used even without invoking the `\SpecialCoor`. Thus `\degrees[100]` is a great help in drawing pie charts, where the data is given in percents, as in the example below:

Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

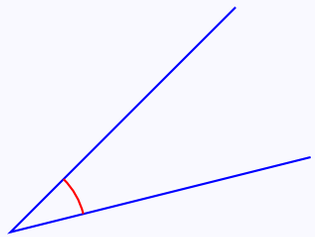
```
\definecolor{PaleApricot}{cmyk}
   {0,0.12,0.32,0}
\begin{pspicture}(-2,-2)(2,2)
  \degrees[100]
  \pswedge*[linecolor=PaleApricot]
    (0,0){2}{0}{40.2}
  \pswedge*[linecolor=Apricot]
    (0,0){2}{40.2}{67.6}
  \pswedge*[linecolor=Tan]
    (0,0){2}{67.6}{87.9}
  \pswedge*[linecolor=Mahogany]
    (0,0){2}{87.9}{100}
\end{pspicture}
```



Angles can be specified in radians by using the command `\radians`. It is equivalent to `\degrees[6.28319]`. (Remember that $\dot{A}$ radians = 180° and that $\dot{A}$ is approximately equal to 3.141592.)

Again in `\SpecialCoor`, angles can be specified in some other ways. We can specify a pair of coordinates indicating the *direction* of the angle as illustrated in the example below. (Note in particular the braces { } surrounding the coordinate pair.)

```
\begin{pspicture}(0,0)(4,3)
  \psline[linecolor=Blue](4,1)(0,0)(3,3)
  \SpecialCoor
  \psarc[linecolor=Red](0,0){1}{(4,1)}{(3,3)}
\end{pspicture}
```



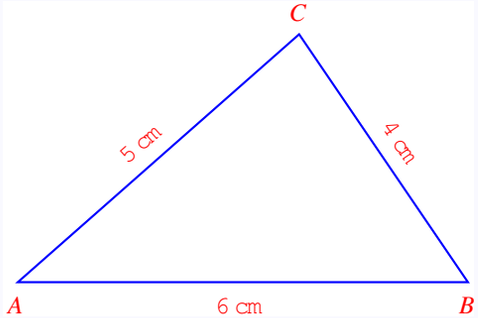
Another way of specifying an angle is to use raw PostScript code which evaluates a number. The code should be preceded by `!`. For example, sup-

Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

pose we want to draw a triangle with sides 2 cm, 3 cm and 4 cm as shown below:



We can specify A as $(0,0)$ and B as $(6,0)$, but what about C? If $\angle A = \theta$, then C has polar coordinates $(5,\theta)$. Now from elementary trigonometry, we have

$$A = 2\tan^{-1}\sqrt{\frac{(s-b)(s-c)}{s(s-a)}}$$

in any triangle ABC with $BC = a$, $CA = b$ and $AB = c$, where $s = \frac{1}{2}(a+b+c)$. For our triangle above, this works out to be

$$A = 2\tan^{-1}\sqrt{\frac{2.5\times 1.5}{7.5\times 3.5}}$$

This computation can be done by PostScript and in the syntax of this language, it is written

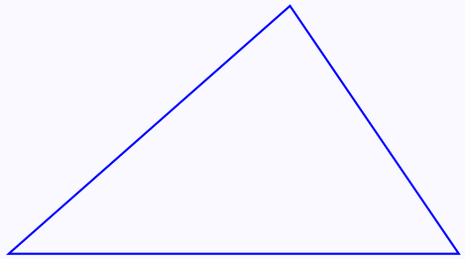
```
2.5 1.5 mul sqrt 7.5 3.5 mul sqrt atan 2 mul
```

(We will explain this a bit in the appendix to this chapter.) Now by the device of including PostScript code in an angle specification using `!`, we can produce the above triangle by

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(0,-0.5)(6,3.5)
  \SpecialCoor
  \pspolygon[linecolor=Blue]%
    (0,0)(6,0)%
    (5;!2.5 1.5 mul sqrt
     7.5 3.5 mul sqrt atan 2 mul)
\end{pspicture}
```



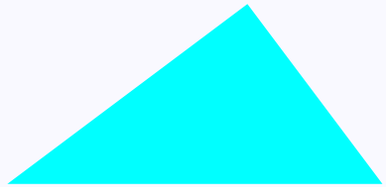
(What about the “labels” for the vertexes and the sides?, Well, that’s another story, better told in a separate chapter.)

Perhaps it is better to have a LaTeX macro to draw a triangle with specified sides. Here’s one:

```
\newcommand{\pstrilateral}[4][]{%
  \SpecialCoor
  \pspolygon[#1](0,0)(#4,0)%
   (#3;!#2 #3 add #4 sub #2 #3 sub #4 add mul sqrt
        #2 #3 add #4 add #3 #4 add #2 sub mul sqrt atan 2 mul)}
```

The command `\pstrilateral` can then be used to draw for example, a “solid” cyan colored triangle of sides 3 cm, 4 cm and 5 cm as shown below:

```
\begin{pspicture}(0,-0.5)(5,2.5)
      \pstrilateral[fillstyle=solid,%
                    fillcolor=Cyan,%
                    linestyle=none]%
                    {3}{4}{5}
    \end{pspicture}
```



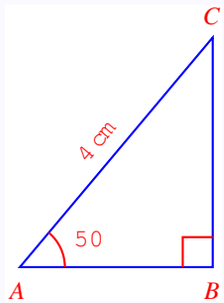
Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

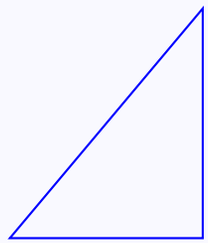
5.4. More special coordinates

Under `\SpecialCoor`, not only angles, but the entire pair of coordinates can be specified using raw PostScript code using the `!` signifier. For example, suppose we want to draw a right angled triangle of hypotenuse 4 cm and one angle equal to 50°, as shown below:



It can be easily seen that B has coordinates $(4\cos 50°, 0)$ and C has coordinates $(4\cos 50°, 4\sin 50°)$. The triangle (sans the labels) can be drawn by writing these coordinates in PostScript as below:

```
\begin{pspicture}(0,-0.5)(3,3.5)
  \SpecialCoor
  \pspolygon[linecolor=Blue]%
            (0,0)%
            (!50 cos 4 mul 0)%
            (!50 cos 4 mul 50 sin 4 mul)
  \end{pspicture}
```



Here, the top vertex (C in the first figure) can also be specified more simply in polar coordinates as `(4;50)`. There's a simpler way to specify B also. Note that the x-coordinate of B is the same as that of C and its y-coordinate is 0. Under `\SpecialCoor`, we can specify the coordinates of a point by referring to these coordinates (in any form) of two other points

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

such that the required point has x-coordinate equal to that of the first point and y-coordinate equal to the y-coordinate of the second point. The general syntax is

(*coordinates1*|*coordinates2*)

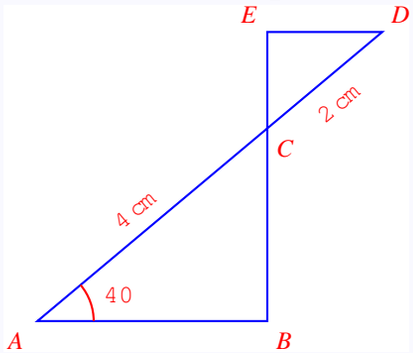
Thus in our example, the point B can be specified as

(4;50|0,0)

(Note that the coordinates of the two reference points are given without enclosing parentheses and a vertical bar | separates these coordinates.) Thus another way of drawing the above triangle is by

```
\begin{pspicture}(0,-1)(3,4)
  \SpecialCoor
  \pspolygon[linecolor=Blue](0,0)(4;50)(4;50|0,0)
\end{pspicture}
```

As an another illustration of this technique, consider the figure below:

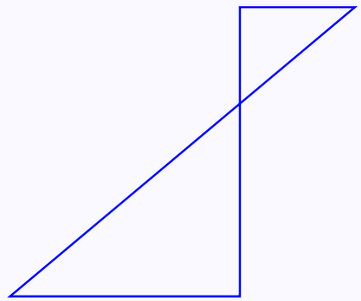


Taking A as $(0,0)$, we can specify C and D by polar coordinates as (4;40) and (6;40). Using the technique just described, B can be specified as (4;40|0,0) and E as (4;40|6;40). Thus this figure (without the labels, of course) can be produced as shown below:

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(0,-0.5)(5,4.5)
  \SpecialCoor
  \pspolygon[linecolor=Blue]%
            (0,0)(6;40)(4;40|6;40)(4;40|0,0)
\end{pspicture}
```



There are somewhere ways of specifying coordinates under \SpecialCoor, using “nodes” and these will be described in another chapter which deals with nodes and their connections using the pst-node package.

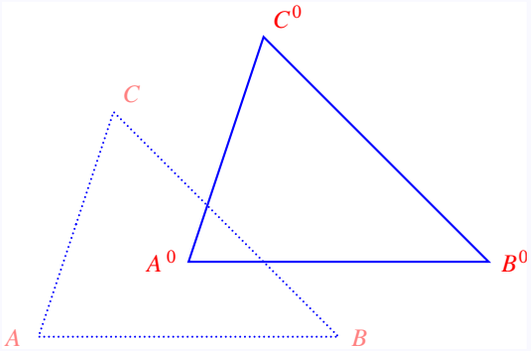
Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

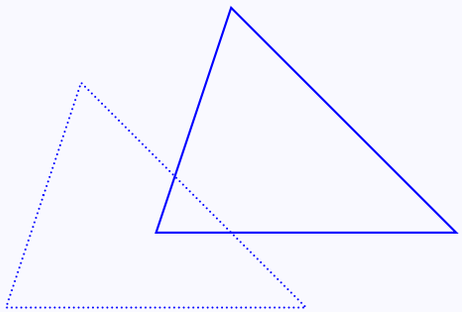
5.5. Changing the system

In drawing pictures, it is sometimes convenient to make some changes to the system of coordinates in the middle. For example, consider the picture below:



The bold triangle $A'B'C'$ is an exact replica of the dotted triangle ABC, only shifted to the right a little. Having drawn ABC, if we can shift the coordinate system to have the origin at A', then the same code could be used to draw $A'B'C'$ also. This can be done by (re)setting the parameter `origin` in the code for drawing $A'B'C'$. Thus the above picture (without the labels, as usual) can be drawn as shown below:

```
\begin{pspicture}(0,0)(6,4)
  \pspolygon[linestyle=dotted,%
             dotsep=1pt,%
             linecolor=Blue]%
            (0,0)(4,0)(1,3)
  \pspolygon[origin={-2,-1},%
             linecolor=Blue]%
            (0,0)(4,0)(1,3)
\end{pspicture}
```

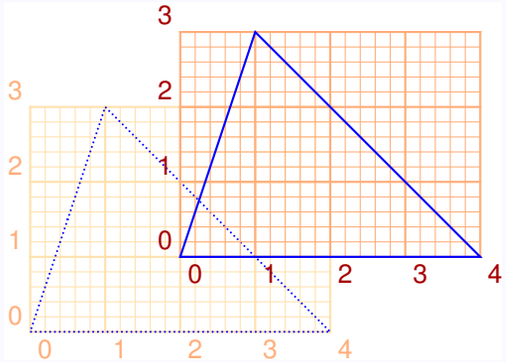


In the second `\pspolygon`, the setting `origin={-2,-1}` translates the co-

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

ordinate axes to a new position such that the original origin has coordinates $(-2,-1)$ with respect to this new system ¿This is better illustrated in the picture below, which shows the triangles together with the two coordinate systems.



Note that, the origin of the original system (shown by the pale grid) is $(-2,-1)$ with respect to the new system (shown by the darker grid). In general, the setting

```
origin={x,y}
```

translates the coordinate axes such that the origin of the original system is (x,y) with respect to the new system. In practical terms, this means, if we want the new origin to be at (x,y), set `origin={-x,-y}`. Note also the use of the curly braces `{ }` to enclose the coordinates, instead of the customary parentheses `( )` in this setting.

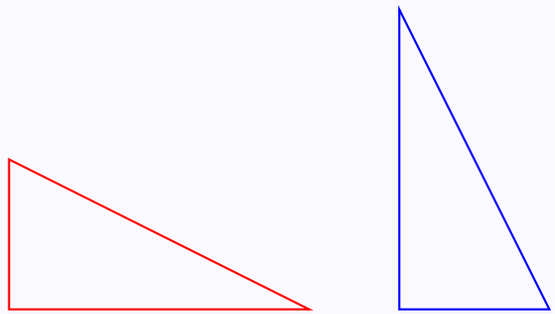
We can also interchange the x and y axes by setting the parameter `\swapaxes` to `true`. (Its default value is `false`.) This is helpful in changing the orientation of a picture. Look at the example below:

Online LaTeX Tutorial
Part II – Graphics
PSTricks

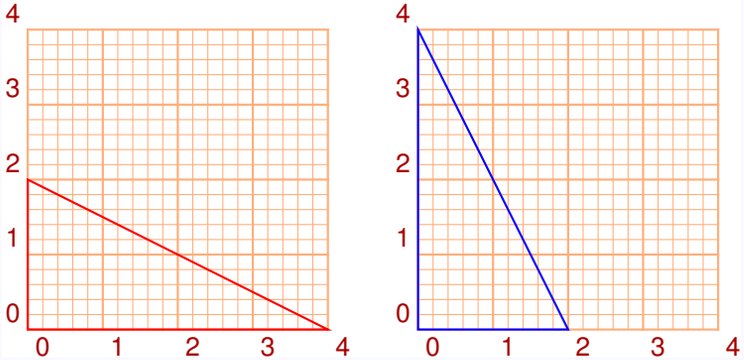
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(0,0)(4,4)
  \pspolygon[linecolor=Red]%
  (0,0)(4,0)(0,2)
\end{pspicture}
\hspace{1cm}
\begin{pspicture}(0,0)(4,4)
  \pspolygon[linecolor=Blue,%
             swapaxes=true]%
            (0,0)(4,0)(0,2)
\end{pspicture}
```



Note that with `\swapaxes=true` in effect, a point with coordinates specified as (a, b) is plotted with x-coordinate b and y-coordinate a. The figures above with the coordinate grids used to draw them makes this clear.



Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

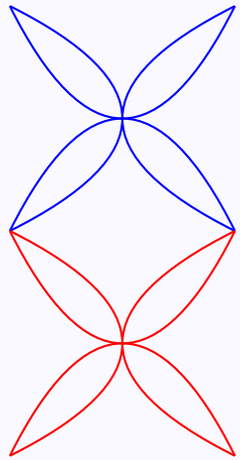
5.6. Setting parameters

Instead of setting the parameters `origin` and `\swapaxes` locally for each object for which we need such effects, we can set them globally with the `\psset` command. This is true for the other graphics parameters such as `linewidth`, `linecolor`, `linestyle` and so on, which we have discussed earlier. The general syntax is

`\psset{`*parameter1=value1*`,`*parameter2=value2*`,...}`

The example below illustrates this:

```
\psset{linecolor=Blue,unit=1.5}
\begin{pspicture}(-1,0)(1,4)
  \parabola(1,1)(0,0)
  \parabola(-1,-1)(0,0)
  \psset{swapaxes=true}
  \parabola(1,1)(0,0)
  \parabola(-1,-1)(0,0)
  \psset{origin={0,2},linecolor=Red}
  \parabola(1,1)(0,0)
  \parabola(-1,-1)(0,0)
  \psset{swapaxes=false}
  \parabola(1,1)(0,0)
  \parabola(-1,-1)(0,0)
\end{pspicture}
```



Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

Appendix—Math in PostScript

We've given a few examples of specifying coordinates using raw PostScript code in the section on "Special Coordinates". Here we give a list of mathematical operators available in this language and their syntax.

OPERATOR	MEANING	syntax	example	
			PostScript CODE	VALUE
`add`	sum of two numbers	*number1* *number2* `add`	`7 2 add`	9
`sub`	difference of two numbers	*number1* *number2* `sub`	`7 2 sub`	5
`mul`	product of two numbers	*number1* *number2* `mul`	`7 2 mul`	14
`div`	quotient of two numbers	*number1* *number2* `sub`	`7 2 div`	3.5
`exp`	power of a number	*number1* *number2* `exp`	`7 2 exp`	49
`idiv`	integral part of the quotient of two integers	*number1* *number2* `idiv`	`7 2 idiv`	3
`mod`	reminder obtained on dividing an integer by an integer	*number1* *number2* `mod`	`7 2 mod`	1
`sqrt`	square root of a number	*number* `sqrt`	`16 sqrt`	4
`neg`	negative of a number	*number* `neg`	`7 neg`	−7
`abs`	absolute value of a number	*number* `abs`	`-7 abs`	7
`ceiling`	smallest integer greater than or equal to a number	*number* `ceiling`	`7.6 ceiling`	8

Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to `tutorialteam@tug.org.in`

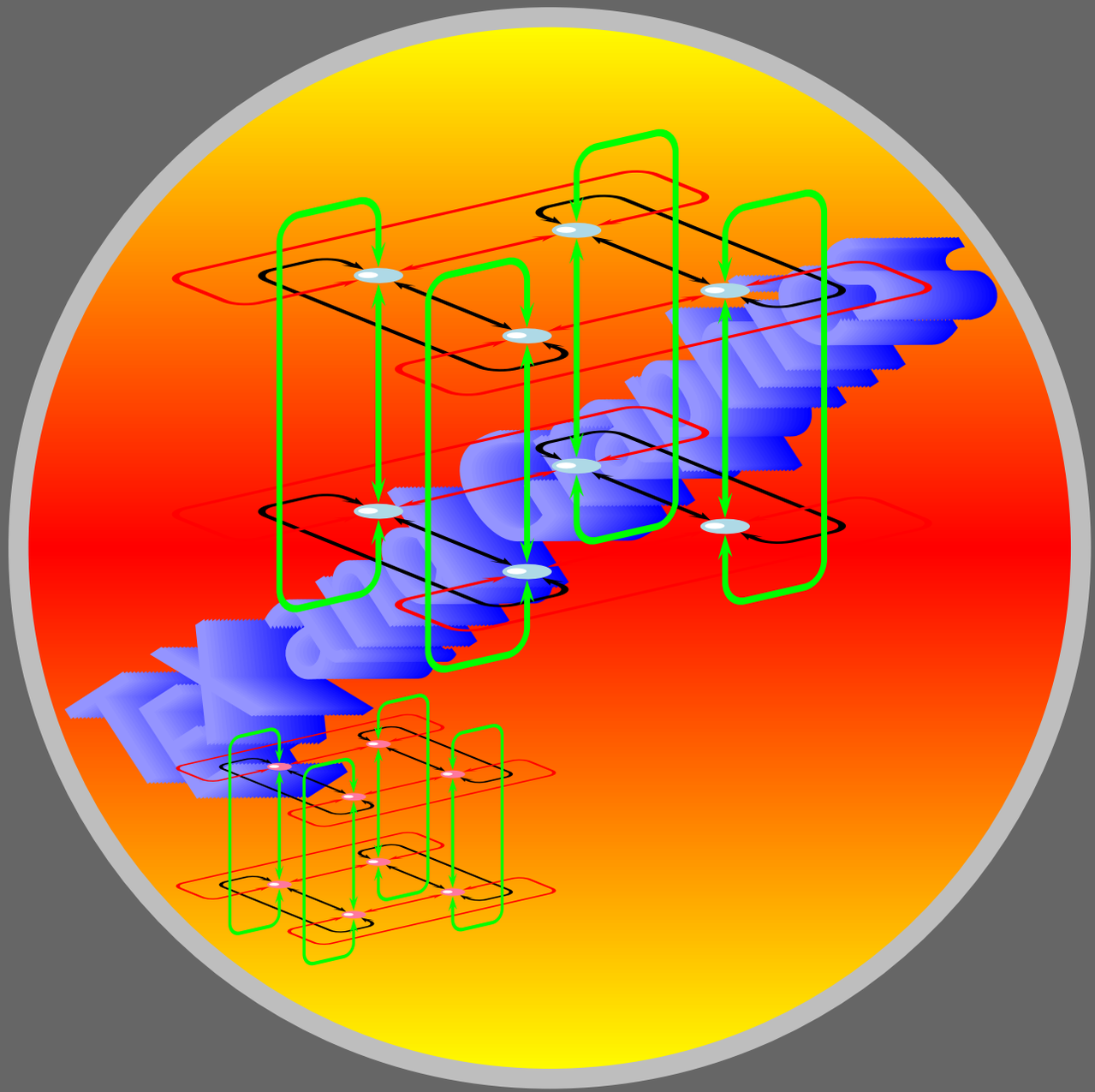
The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
`http://www.tug.org.in`

OPERATOR	MEANING	syntax		example	
				PostScript CODE	VALUE
floor	largest integer less than or equal to a number	*number*	floor	7.6 floor	7
round	round a number to the nearest integer	*number*	round	7.6 round 7.2 round	8 7
sin	sine of number in degrees	*number*	sin	30 sin	0.5
cos	cosine of number in degrees	*number*	cos	60 cos	0.5
atan	inverse tangent of number in degrees	*number*	atan	1 atan	45
ln	natural logarithm (base e) of number	*number*	ln	2.71828182 ln	1
log	logarithm of number to base 10	*number*	log	100 log	2

Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team.
Comments and suggestions may be mailed to
tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



Placing Things

Placing and rotating PSTricks objects
Placing and rotating TEX objects
Putting labels

Online LATEX Tutorial

Part II – Graphics

PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team.
Comments and suggestions may be mailed to
tutorialteam@tug.org.in

This document is generated by PDFTEX with hyperref, pstricks, pdftricks and pdfscreen packages in an intel PC running GNU/LINUX and is released under LPPL

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

http://www.tug.org.in

6. Placing Things

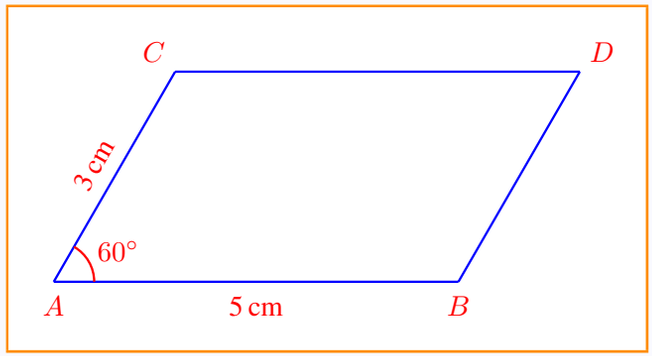
PSTricks have several commands to place graphics objects and normal LaTeX material where you want and rotate them, if need be. We discuss some of these in this chapter. Other such commands will be described in the chapter on nodes and connections.

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

6.1. Placing and rotating PSTricks objects

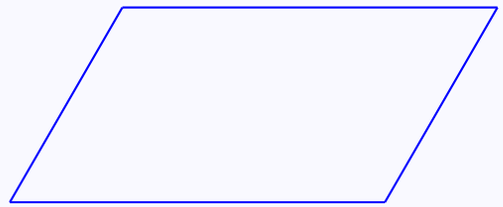
Suppose we want to draw a picture like the one given below using PSTricks:



Taking A as $(0,0)$, we can easily specify B as $(5,0)$ and C as $(3;60)$. (Remember `\SpecialCoor` of the last chapter?). Again, by shifting the origin to B, by setting `origin={-5,0}`, we can specify D as $(3;60)$. And then AB, AC and BD can be drawn with `\psline`. But how do we draw CD? Recall that C and D are specified using different coordinate systems.

One way out of this difficulty is, instead of trying to join C and D, to shift the line AB to the position of BD. The `\rput` command in PSTricks fills the bill here. Thus the above figure (without the labels) can be drawn thus:

```
\begin{pspicture}(0,0)(6,2.5)
  \psset{linecolor=Blue}
  \psline(0,0)(5,0)
  \SpecialCoor
  \psline(0,0)(3;60)
  \rput(3;60){\psline(0,0)(5,0)}
  \psline[origin={-5,0}](0,0)(3;60)
\end{pspicture}
```



Look at the way the `\rput` command is used. The `\psline(0,0)(5,0)` is

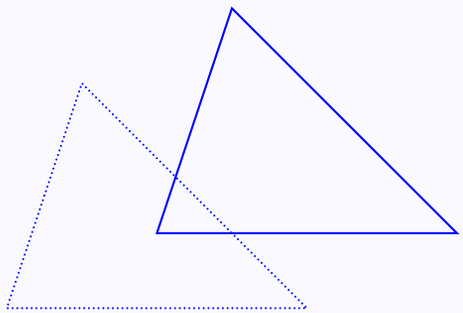
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

the object to place and the coordinates `(3;60)` is position to place the point $(0,0)$ of the line (that is the object).

As another example, let's draw a figure we did in the last chapter by shifting the origin this time using the `\rput` command:

```
\begin{pspicture}(0,-0.5)(6,4.5)
  \begin{pspicture}(0,0)(6,4)
    \pspolygon[linestyle=dotted,%
               dotsep=1pt,%
               linecolor=Blue]%
               (0,0)(4,0)(1,3)
    \rput(2,1){%
      \pspolygon[linecolor=Blue]%
                (0,0)(4,0)(1,3)}
  \end{pspicture}
\end{pspicture}
```

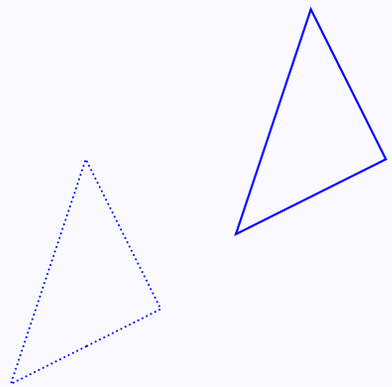


It must be carefully noted that *within* a command `\rput(`*coordinates*`)`, a new coordinate system is used, with origin at the point with the specified *coordinates*. In other words, what the command `\rput(`*coordinates*`){`*object*`}` does is to *draw* the *object* with reference to a new coordinate system with the axes translated to the point with *coordinates* as the origin. The example below illustrates this:

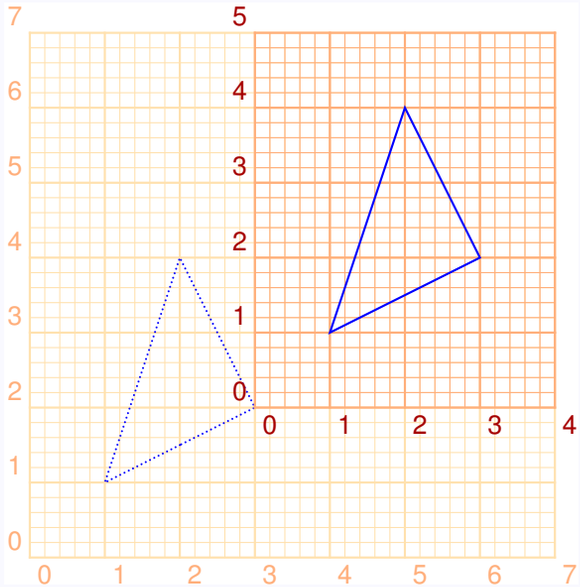
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(0,1)(6,6)
  \pspolygon[linecolor=Blue,%
             linestyle=dotted,%
             dotsep=1pt]%
             (1,1)(2,4)(3,2)
  \rput(3,2){%
    \pspolygon[linecolor=Blue]%
              (1,1)(2,4)(3,2)}
\end{pspicture}
```



The same picture is shown below with the original coordinate system in a lighter shade and that within the `\rput(3,2)` in a darker shade, to clarify this idea:



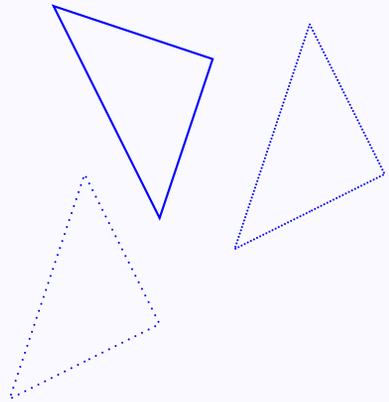
Another interesting feature of the `\rput` command is that it can be given an optional argument to *rotate* the object it puts in place. Thus `\rput{`*angle*`}(`*coordinates*`){`*object*`}`. draws the object with reference to

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

a new system of axes obtained by first translating the original axes to the point with *coordinates* as origin and then rotating them through *angle*. Look at the next example:

```
\begin{pspicture}(-1,0)(7,9)
  \pspolygon[linecolor=Blue,%
             linestyle=dotted,%
             dotsep=2pt]%
            (1,1)(2,4)(3,2)
  \rput(3,2){%
    \pspolygon[linecolor=Blue,%
               linestyle=dotted,%
               dotsep=0.5pt]%
              (1,1)(2,4)(3,2)}
  \rput{45}(3,2){%
    \pspolygon[linecolor=Blue](1,1)(2,4)(3,2)}
\end{pspicture}
```



The picture below, which shows the picture above with the different coordinate systems shown in different colors, will make this clear.

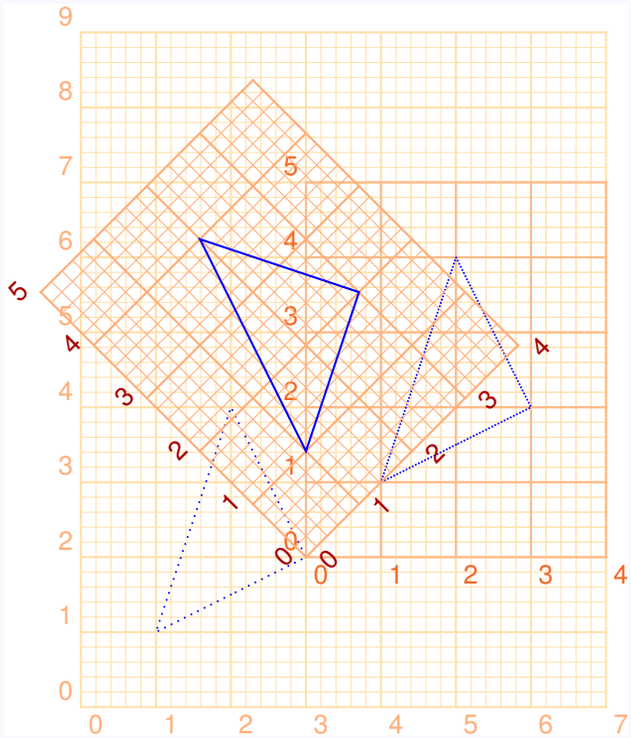
Online LaTeX Tutorial

Part II – Graphics

PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



Placing Things

Placing and rotating PSTricks objects
Placing and rotating TEX objects
Putting labels

Online LATEX Tutorial

Part II – Graphics

PSTricks

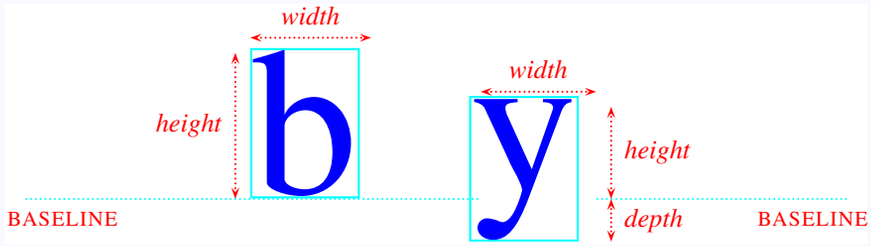
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team.
Comments and suggestions may be mailed to
tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

6.2. Placing and rotating TEX objects

The \rput command can also be used for placing and rotating TEX material, which is usually text of some sort. Before giving examples of this, something must be said about the way TEX does its typesetting job. Internally, TEX does this with *boxes*. For example, when you type {\Large boy} in your document, you get boy after TEX has complied it *and* the printer driver has processed it, but what TEX produces internally is something like  or more precisely,  with instructions on what to put in each box. In other words, individual characters are put in *character boxes* which are stringed together horizontally to form *line boxes* which again are packed vertically into *paragraph boxes* and so on.

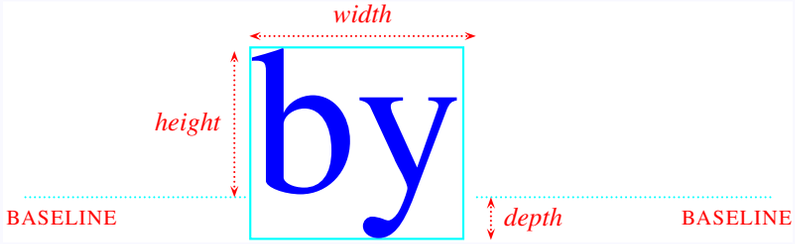
Note that in typesetting, the bottom of the letters without *descenders* (such as for example, a, b, c, d, e but not f and g) are aligned along a line. This is called the *baseline*. In TEX every box has a *width* (no explanation needed), *height* (which is its height *above the baseline*) and *depth* (which is the distance it extends below the baseline) as shown below for two character boxes:



Thus the “b” box has 0 depth, while the “y” box a positive depth. Note that when these are joined to form the word “by”, the resulting box has positive depth, equal to the depth of “y”

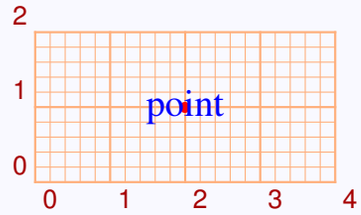
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

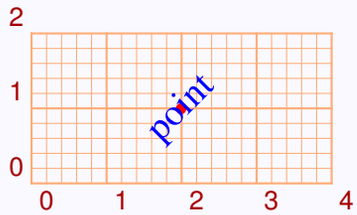


Now when we give a command `\rput(`*coordinate*`){`*text*`}`, TEX makes a box to fit *text* and `\rput` places the center of the box at the point with specified *coordinates*. Also, any rotation specified in the `\rput` command is about this point, as shown below:

```
\psdots[linecolor=Red,dotsize=0 5](2,1)
\rput(2,1){\Large\color{Blue} point}
```




```
\psdots[linecolor=Red,dotsize=0 5](2,1)
\rput{45}(2,1){\Large\color{Blue} point}
```

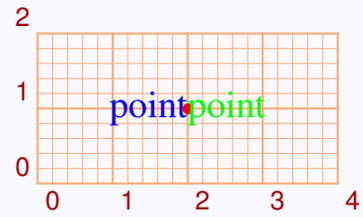


The textbox can be placed with the the midpoint of its left or right edge on the specified point (and then rotated about this point, if desired) using the optional arguments `l` or `r` immediately after `\rput` as shown below:

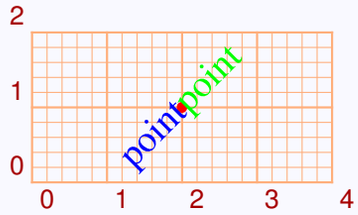
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\psdots[linecolor=Red,dotsize=0 5](2,1)
\rput[l](2,){\Large\color{Green}point}
\rput[r](2,1){\Large\color{Blue}point}
```

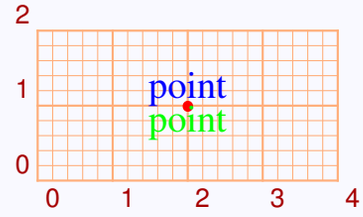



```
\psdots[linecolor=Red,dotsize=0 5](2,1)
\rput[l]{45}(2,){\Large\color{Green}point}
\rput[r]{45}(2,1){\Large\color{Blue}point}
```

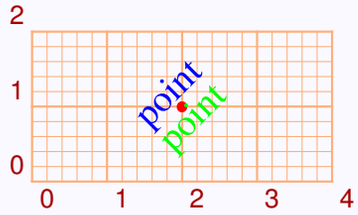


On the other hand, the textbox can be placed with the midpoint of its top or bottom edge at the specified point (and rotated about this point) using the `t` or `b` option.

```
\psdots[linecolor=Red,dotsize=0 5](2,1)
\rput[t](2,){\Large\color{Green}point}
\rput[b](2,1){\Large\color{Blue}point}
```




```
\psdots[linecolor=Red,dotsize=0 5](2,1)
\rput[t]{45}(2,){\Large\color{Green}point}
\rput[b]{45}(2,1){\Large\color{Blue}point}
```

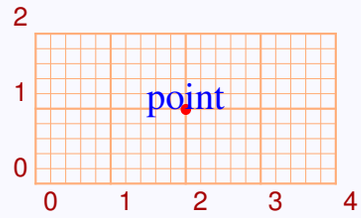


E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

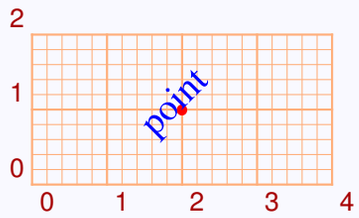
The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

For vertically shifting a textbox from its default placement, there is another option `B`, which places the box with the midpoint of the baseline at the specified point as in the picture below:

```
\psdots[linecolor=Red,dotsize=0 5](2,1)
\rput[B](2,){\Large\color{Blue}point}
```




```
\psdots[linecolor=Red,dotsize=0 5](2,1)
\rput[B]{45}(2,){\Large\color{Blue}point}
```

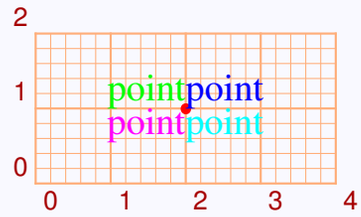


Using the combinations `bl`, `br`, `tl`, `tr`, the textbox can also be placed with any of its corners at the specified point (and rotated too).

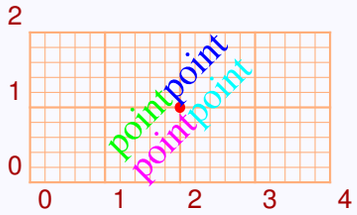
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\psdots[linecolor=red,dotsize=0 5](2,1)
\rput[bl](2,1){\Large\color{Blue} point}
\rput[br](2,1){\Large\color{Green} point}
\rput[tl](2,1){\Large\color{Cyan} point}
\rput[tr](2,1){\Large\color{Magenta} point}
```

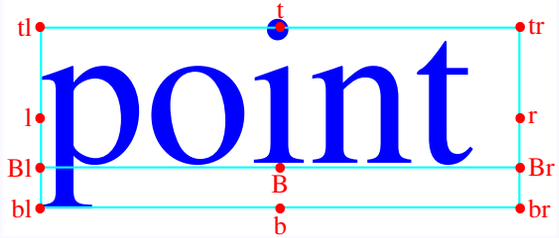



```
\psdots[linecolor=red,dotsize=0 5](2,1)
\rput[bl]{45}(2,1){\Large\color{Blue} point}
\rput[br]{45}(2,1){\Large\color{Green} point}
\rput[tl]{45}(2,1){\Large\color{Cyan} point}
\rput[tr]{45}(2,1){\Large\color{Magenta} point}
```



The combinations `Br` and `Bl` can also be used. (Try them!)

The picture below shows those points on the textbox placed at the specified point for various positional arguments to the `\rput` command:



Thus the full syntax of the command is

`\rput[`*boxpoint*`]{`*angle*`}{`*coordinates*`}{`*text*`}`

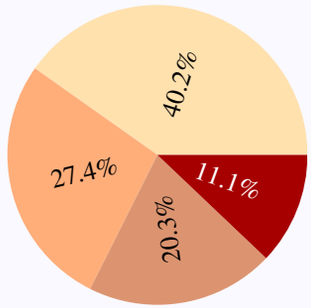
Since the pure graphic objects produced by PSTricks are zero-dimensional TEX boxes, the *boxpoint* argument is redundant for placing such objects.

As an example, let's see how we can put labels on the pie-chart drawn in the last chapter:

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

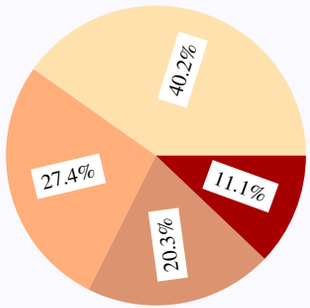
The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\definecolor{PaleApricot}{cmyk}{0,0.12,0.32,0}
\begin{pspicture}(-2,-2)(2,2)
   \degrees[100]
   \SpecialCoor
   \pswedge*[linecolor=PaleApricot](0,0){2}{0}{40.2}
   \pswedge*[linecolor=Apricot](0,0){2}{40.2}{67.6}
   \pswedge*[linecolor=Tan](0,0){2}{67.6}{87.9}
   \pswedge*[linecolor=Mahogany](0,0){2}{87.9}{100}
   \rput{20.1}(1;20.2){40.2\%}
   \rput{3.7}(1;53.7){27.4\%}
   \rput{27.15}(1;77.15){20.3\%}
   \rput{-5.55}(1;-5.55){\color{White} 11.1\%}
\end{pspicture}
```



The \rput has also a starred form, in which the text is first put into a white box (actually a \psframebox*, but that can wait) and then placed in the required point, blotting out whatever is behind. For example, the pie-chart above can be labeled differently as below:

```
\definecolor{PaleApricot}{cmyk}{0,0.12,0.32,0}
\begin{pspicture}(-2,-2)(2,2)
   \degrees[100]
   \SpecialCoor
   \pswedge*[linecolor=PaleApricot](0,0){2}{0}{40.2}
   \pswedge*[linecolor=Apricot](0,0){2}{40.2}{67.6}
   \pswedge*[linecolor=Tan](0,0){2}{67.6}{87.9}
   \pswedge*[linecolor=Mahogany](0,0){2}{87.9}{100}
   \rput*{20.1}(1;20.2){40.2\%}
   \rput*{3.7}(1;53.7){27.4\%}
   \rput*{27.15}(1;77.15){20.3\%}
   \rput*{-5.55}(1;-5.55){11.1\%}
\end{pspicture}
```



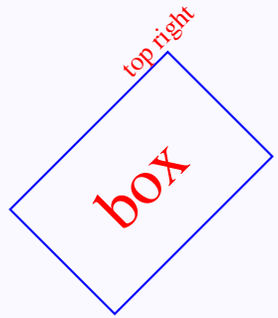
The \rput commands can be nested. Look at this example:

Online LATEX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

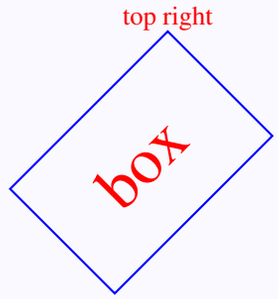
The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\definecolor{PaleApricot}{cmyk}{0,0.12,0.32,0}
\begin{pspicture}(0,0)(4,4)
  \rput[bl]{45}(0,0){%
    \psframe[linecolor=Blue](0,0)(3,2)
    \rput(1.5,1){\color{Red}\Huge box}
    \rput[b](3,2){\color{Red}top right}}
\end{pspicture}
```



The above picture will look better with the text “top right” set horizontally (that is, without the rotation through 45°. For this, the rotation due to the outer `\rput` has to be reset. This can be done by giving the angle argument `*0` to the inner `\rput`, where the `*` option resets the previous rotation.

```
\definecolor{PaleApricot}{cmyk}{0,0.12,0.32,0}
\begin{pspicture}(0,0)(4,4)
  \rput[bl]{45}(0,0){%
    \psframe[linecolor=Blue](0,0)(3,2)
    \rput(1.5,1){\color{Red}\Huge box}
    \rput[b]{*0}(3,2){\color{Red}top right}}
\end{pspicture}
```



Some of the frequently required angles for rotation have shorter representation as letters, listed in the table below:

Online LATEX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

ANGLE	LETTER	MEANING	ANGLE	LETTER	MEANING
0	U	Up	*0	N	North
90	L	Left	*90	W	West
180	D	Down	*180	S	South
270	R	Right	*270	E	East

Thus for example, instead of `\rput[90]`, we can also say `\rput[L}`.

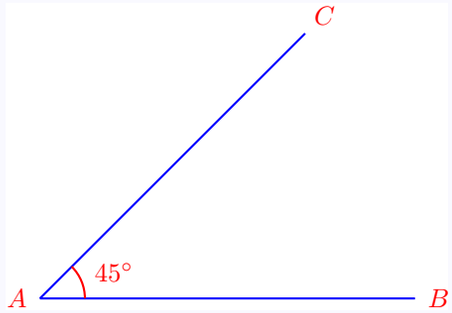
Online LATEX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

6.3. Putting labels

One of the frequent needs for placing objects in specific locations is in affixing labels to parts of technical (especially mathematical) drawings. Consider the simple example shown below:



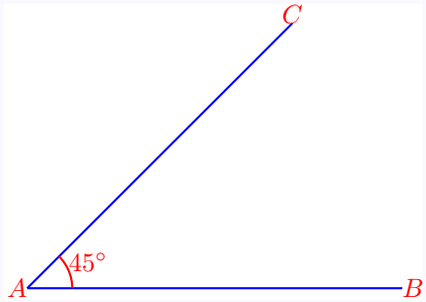
If we try to get this with something like

```
\begin{pspicture}(0,-1)(6,5)
  \psset{linecolor=Blue}
  \psline(0,0)(5,0)
  \SpecialCoor
  \psline(0,0)(5;45)
  \rput[r](0,0){\color{Red} $A$}
  \rput[l](5,0){\color{Red} $B$}
  \rput[b](5;45){\color{Red} $C$}
  \psarc[linecolor=Red](0,0){0.6}{0}{45}
  \rput[bl](0.6;22.5){\color{Red} $45^\circ$}
\end{pspicture}
```

we get

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



where the labels sort of stick to the points referred to. So, to `\rput` labels, we have to compute the coordinates of points a little away from the points referred to. The command `\uput` saves us this trouble. Thus the command

`\uput[`*dirangle*`](`*coordinates*`){`*label*`}`

puts *label* at a distance of 5 pt away from the point with the specified *coordinates* in the direction specified by *dirangle*. thus the first figure above is produced by

```
\begin{center}
  \begin{pspicture}(0,-1)(6,3.5)
    \psset{linecolor=Blue}
    \psline(0,0)(5,0)\SpecialCoor
    \psline(0,0)(5;45)
    \uput[180](0,0){\color{Red} $A$}
    \uput[0](5,0){\color{Red} $B$}
    \uput[45](5;45){\color{Red} $C$}
    \psarc[linecolor=Red](0,0){0.6}{0}{45}
    \uput[30](0.6;22.5){\color{Red} $45^\circ$}
  \end{pspicture}
\end{center}
```

The distance between the point and the label is controlled by the parameter `labelsep`. We can change its default value (5 pt) by using `psset`. We can

Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

also include this value as the first argument to `\uput` itself. Thus in the above picture, if we want the label 45° to be a little closer to the arc, then we change the last line in the above code as

```
\uput{3pt}[30](0.6;30){\color{Red} $45^\circ$}
```

(Try it!) Not that the value of `labelsep` is to be given within curly braces `{ }`.

As in the case of rotation angles in `\rput`, the frequently used direction angles in `\uput` also has letter abbreviations.

ANGLE	LETTER	MEANING	ANGLE	LETTER	MEANING
0	r	right	45	ur	up-right
90	u	up	135	ul	up-left
180	l	left	225	dl	down-left
270	d	down	315	dr	down-right

Thus the code for the first picture above can also be written

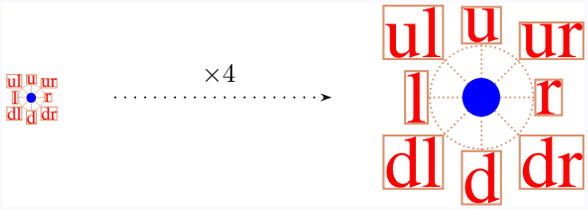
```
\begin{center}
  \begin{pspicture}(0,-1)(6,3.5)
    \psset{linecolor=Blue}
    \psline(0,0)(5,0)\SpecialCoor
    \psline(0,0)(5;45)
    \uput[l](0,0){\color{Red} $A$}
    \uput[r](5,0){\color{Red} $B$}
    \uput[ur](5;45){\color{Red} $C$}
    \psarc[linecolor=Red](0,0){0.6}{0}{45}
    \uput[22.5](0.6;22.5){\color{Red} $45^\circ$}
  \end{pspicture}
\end{center}
```


Online LATEX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

We show below The positions of the *label boxes* with direction specified by all these letters relative to a specified point, both in actual size and magnified four-fold (using the `\tiny` font for labels):



The `\uput` command also has an argument to rotate the object placed and this argument is to be included in curly braces `{ }`, between the *direction* and *coordinate* specifications. Thus the full form of the command `\uput` is

`\uput{`*labelsep*`}[`*dirangle*`]{`*rotangle*`}(`*coordinates*`){`*label*`}`

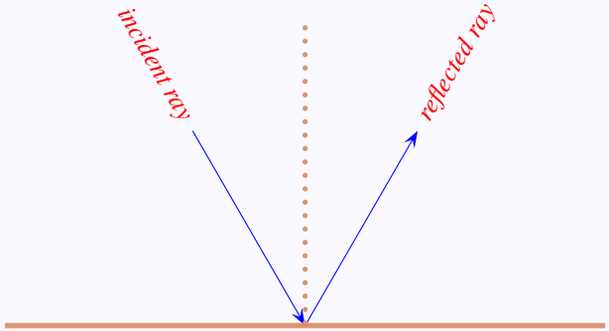
The example below shows the use of rotated labels with `\uput`

Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(-4,0)(4,5)
  \SpecialCoor
  \psline[linecolor=Tan,%
          linewidth=2pt]%
         (-4,0)(4,0)
  \psline[linecolor=Blue,%
          linewidth=0.4pt,%
          arrowsize=3pt 4]{->}%
         (3;120)(0,0)
  \psline[linecolor=Blue,%
          linewidth=0.4pt,%
          arrowsize=3pt 4]{->}%
         (0,0)(3;60)
  \psline[linecolor=Tan,%
          linewidth=2pt,%
          linestyle=dotted]%
          (0,0)(0,4)
  \uput[120]{300}(3;120){%
    \color{Red}\textit{incident ray}}
  \uput[60]{60}(3;60){%
    \color{Red}\textit{reflected ray}}
\end{pspicture}
```

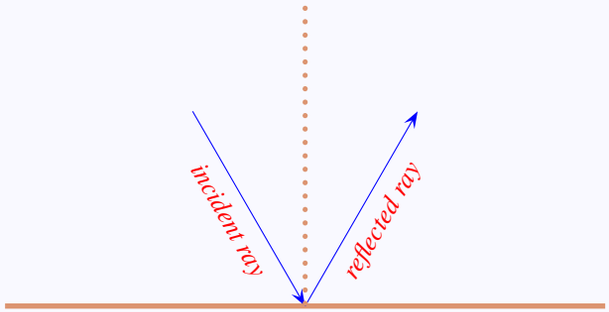


Sometimes for the correct placement of labels, a combination of `\rput` and `\uput` maybe the easiest solution, as in the next example:

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(-4,0)(4,3)
  \SpecialCoor
  \psline[linecolor=Tan,%
          linewidth=2pt]%
         (-4,0)(4,0)
  \psline[linecolor=Blue,%
          linewidth=0.4pt,%
          arrowsize=3pt 4]{->}%
         (3;120)(0,0)
  \psline[linecolor=Blue,%
          linewidth=0.4pt,%
          arrowsize=3pt 4]{->}%
         (0,0)(3;60)
  \psline[linecolor=Tan,%
          linewidth=2pt,%
          linestyle=dotted]%
          (0,0)(0,4)
  \rput[b]{300}(1.5;120){%
    \uput[d](0,0){\color{Red}%
      \textit{incident ray}}}
  \rput[b]{60}(1.5;60){%
    \uput[d](0,0){\color{Red}%
      \textit{reflected ray}}}
\end{pspicture}
```



Now we can give the complete code for the first picture of this chapter, labels and all:

Online LATEX Tutorial

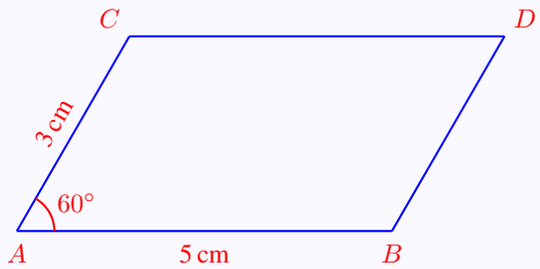
Part II – Graphics

PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(0,-1)(6,3.5)
  \psset{linecolor=Blue}
  \psline(0,0)(5,0)
  \SpecialCoor
  \psline(0,0)(3;60)
  \rput(3;60){%
    \psline(0,0)(5,0)
    \uput[ur](5,0){%
      \color{Red} $D$}}
  \psline[origin={-5,0}]%
         (0,0)(3;60)
  \psarc[linecolor=Red]%
        (0,0){0.5}{0}{60}
  \uput{0.6}[30](0,0){%
    \color{Red} $60^\circ$}
  \uput[d](2.5,0){%
    \color{Red} 5\,cm}
  \rput[b]{60}(1.5;60){%
    \uput[u](0,0){%
      \color{Red} 3\,cm}}
  \uput[d](0,0){\color{Red} $A$}
  \uput[d](5,0){\color{Red} $B$}
  \uput[ul](3;60){\color{Red} $C$}
\end{pspicture}
```



Placing Things

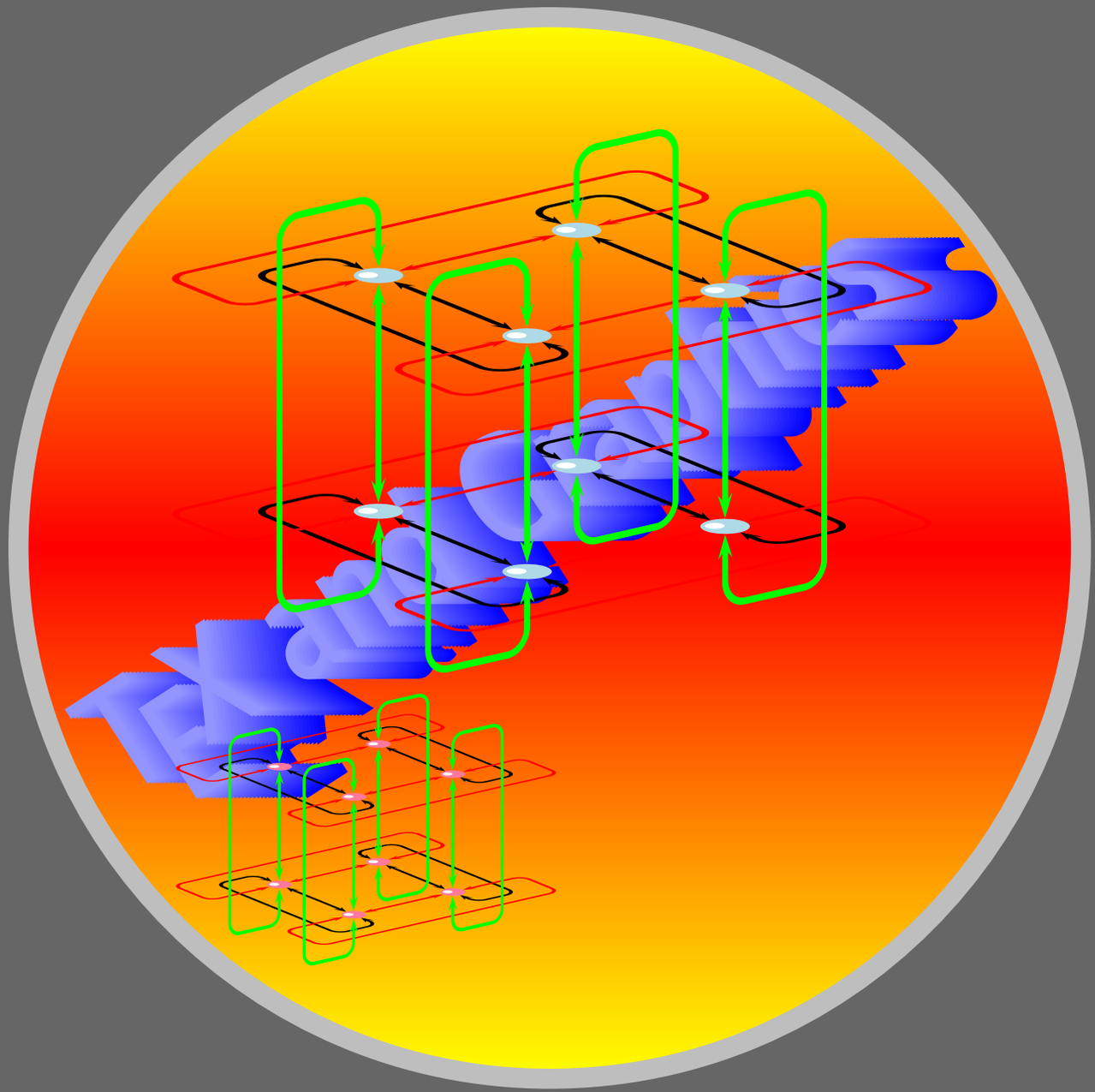
Online LaTeX Tutorial

Part II – Graphics

PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



Plotting Tricks

Function plotting
Axes of coordinates
Data plotting

Online LaTeX Tutorial

Part II – Graphics

PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to
tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

http://www.tug.org.in

7. Plotting Tricks

We have seen how we can join points to produce curves (for a mathematician, a *line* is also a curve!) with PSTricks, using such commands as `\psline`, `\pspolygon`, `\pscurve`, `\psccurve` and `\psecurve`. In all these commands, we'll have to *explicitly provide the coordinates* of the points joined to make the curve *within the command* itself. There are other methods of specifying the coordinates, either implicitly as a mathematical equation or as data piped in from another source. We discuss such tricks in this chapter.

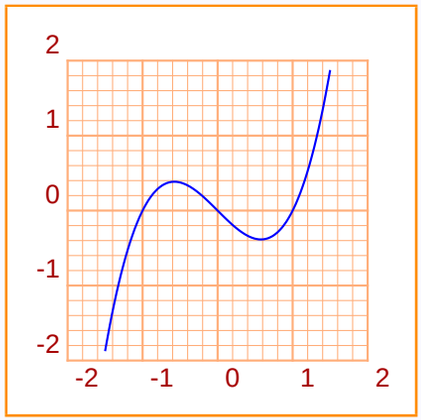
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to
tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

7.1. Function plotting

For a mathematician, analytical geometry serves as a means of translation between algebra and geometry: an algebraic equation can be geometrically represented as the curve joining the points with coordinates satisfying the equation (called the *graph* of the equation) (and on the other hand, a curve can be algebraically represented as an equation specifying the relation between the coordinates of the points on it).

Thus for example, the graph of the equation $y = x^3 - x$ for $-1.5 \leq x \leq 1.5$ is as shown below:



The package pst-plot contains the command `\psplot` to draw curves specified by such equations. (Throughout the following, we assume that the package pst-plot is loaded by the command `\usepackage{pst-plot}` in the preamble.) The equation must be written in the PostScript Language. Thus the above equation is produced by

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(-2,-3)(2,3)
  \psgrid[gridcolor=Apricot,%
          gridlabelcolor=Mahogany,%
          subgridcolor=Apricot]
  \psplot[plotstyle=curve,%
          linecolor=Blue]%
         {-1.5}{1.5}{x 3 exp x sub}
 \end{pspicture}
```

The first command `\psgrid` just draws the coordinate grid, as we have seen earlier. The actual curve is drawn by the `\psplot` command. Let's look at this a bit more closely. The expression `x 3 exp x sub` is just the PostScirpt way of saying $x^3 - x$. In general, we must specify the equation which gives *the y-coordinate of each point of the curve in terms of the x-coordinate*. (or as mathematicians prefer to say, y should be given as a function of x). The numbers `-1.5` and `1.5` give respectively the minimum and maximum value of the x-coordinate in the curve to be drawn. Thus the basic drawing command is

`\psplot{`*min x*`}{`*man x*`}{`*equation*`}`

Looking back at our example, we see that here, the command `\psplot` also includes some parameters. The setting `linecolor=Blue` just makes the curve blue. What about `plotstyle=curve`? This makes `\psplot` draw a `\pscurve` through a certain number of points with coordinates computed from the equation. The number of points used is by default 50, but it can be controlled by the parameter `plotpoints`. This is illustrated in the next example:

Online LaTeX Tutorial

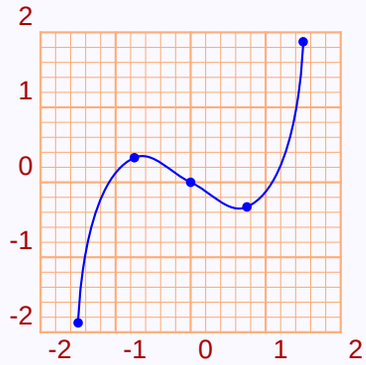
Part II – Graphics

PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

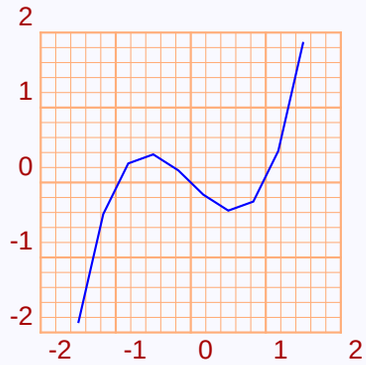
```
\begin{pspicture}(-2,-2)(2,2)
  \psgrid[gridcolor=Apricot,%
          gridlabelcolor=Mahogany,%
          subgridcolor=Apricot]
  \psplot[plotstyle=curve,%
          linecolor=Blue,%
          plotpoints=5,%
          showpoints=true]%
         {-1.5}{1.5}{x 3 exp x sub}
\end{pspicture}
```



The curve is not at all an accurate picture of the equation. In general, increasing the `plotpoints` increases the accuracy of the curve, but also increases the imaging time.

Coming back to the `plotstyle` parameter, we can also set it to `ccurve` or `ecurve` which draw a `\psccurve` (closed curve) or a `\psecurve` (suppressing the end-points in the picture). Other values are `line` and `polygon` which draw a `\psline` and a `\pspolygon` respectively. For example, with `plotstyle=line` the above graph is as shown below:

```
\begin{pspicture}(-2,-2)(2,2)
  \psgrid[gridcolor=Apricot,%
          gridlabelcolor=Mahogany,%
          subgridcolor=Apricot]
  \psplot[plotstyle=line,%
          linecolor=Blue,%
          plotpoints=10]%
         {-1.5}{1.5}{x 3 exp x sub}
\end{pspicture}
```



Online LaTeX Tutorial
Part II – Graphics
PSTricks

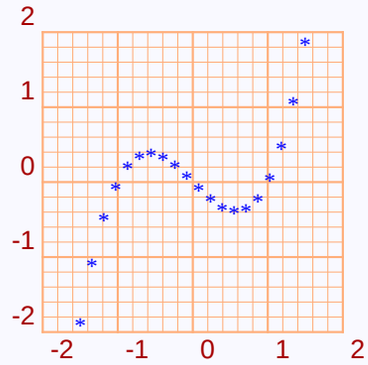
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

Note that with sufficiently many points specified in `plotpoints`, we get the same curve with `plotsyle=curve` and `plotstyle=line`, since PostScript draws any curve by joining short enough line segments).

There's one more value `plotstye=dots`, which draws only the dots used to draw the curve, without joining them. here, we can also use the various `dotstyle` values described in the first chapter. This is shown in the next example:

```
\begin{pspicture}(-2,-2)(2,2)
  \psgrid[gridcolor=Apricot,%
          gridlabelcolor=Mahogany,%
          subgridcolor=Apricot]
  \psplot[plotstyle=dots,%
          dotstyle=asterisk,%
          linecolor=Blue,%
          plotpoints=20]%
         {-1.5}{1.5}{x 3 exp x sub}
\end{pspicture}
```



Other relevant parameters can also be set for the `\psplot` command. Thus if we want to plot the function $f(x) = \sin x$ with x measured in radians (recall that PostScript computes $\sin x$ for x measured in degrees), we will have to scale down the x coordinates by $\frac{\dot{A}}{180} = 0.0174$. This can be done as shown below:

Online LaTeX Tutorial

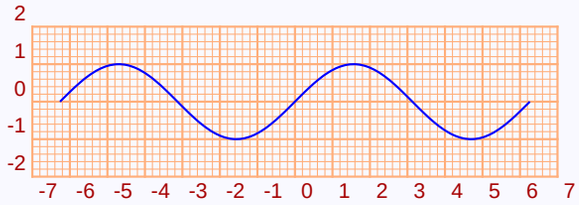
Part II – Graphics

PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\psset{unit=0.5}
\begin{pspicture}(-7,-2)(7,2)
  \psgrid[gridcolor=Apricot,%
          gridlabelcolor=Mahogany,%
          subgridcolor=Apricot,%
          gridlabels=8pt]
  \psplot[xunit=0.0174,%
          plotstyle=curve,%
          linecolor=Blue]%
         {-360}{360}{x sin}
\end{pspicture}
```



Sometimes, the relation between the x and y coordinates of points on a curve are not specified directly as an equation connecting x and y, but instead as an a pair of equations giving x and y in terms of a third parameter t. For example,

$$x = \frac{t}{1+t^2}$$
$$y = \frac{t^3}{1+t^2}$$

The graph of this function for $-3 \le t \le 3$ can be drawn as shown below:

Online LaTeX Tutorial

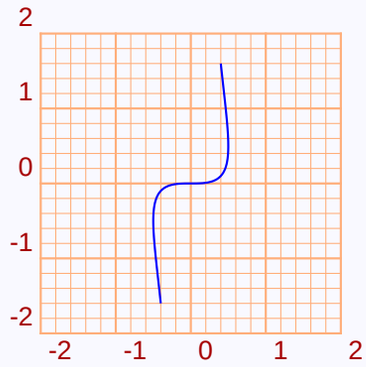
Part II – Graphics

PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

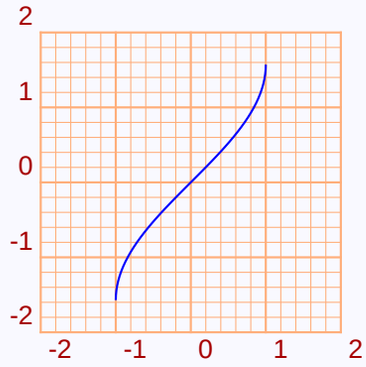
The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(-2,-2)(2,2)
  \psgrid[gridcolor=Apricot,%
                gridlabelcolor=Mahogany,%
                subgridcolor=Apricot]%
                (-2,-2)(2,2)
  \parametricplot[plotstyle=curve,%
                  linecolor=Blue]%
                  {-2}{2}%
                  {t t 2 exp 1 add div t 3 exp
                   t 2 exp 1 add div}
\end{pspicture}
```



The parametric plot provides an easy method to draw inverse functions. The example below shows how we can plot the inverse sine function:

```
\begin{pspicture}(-2,-2)(2,2)
  \colgrid
  \parametricplot[linecolor=Blue]%
                 {-1.5708}{1.5708}%
                 {57.2958 t mul sin t}
\end{pspicture}
```



Note the multiplication by $57.2958 = \frac{180}{\pi}$ for conversion to radians.

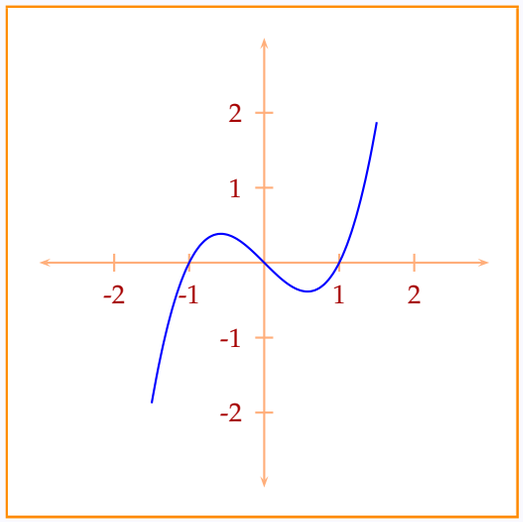
Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

7.2. Axes of coordinates

Often in mathematical documents, graphs of equations are shown, not with the complete coordinate grid, but only with the x and y axes as shown below:



We can draw such axes using `\psline`, but pst-plot also includes the command `\psaxes` to draw such axes more conveniently.

7.2.1. Drawing the axes

The command

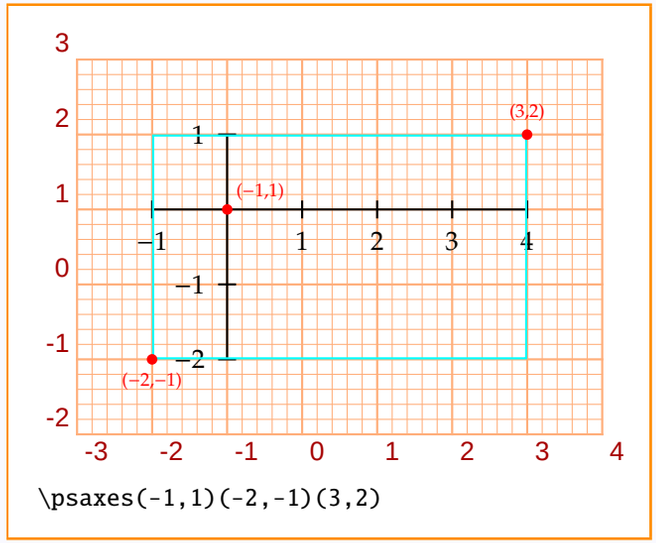
$$\texttt{\textbackslash psaxes}(x_0, y_0)\,(x_1, y_1)\,(x_2, y_2)$$

draws x and y axes with the x-axis from x_1 to x_2 and the y-axis from y_1 to y_2 with the origin (for the axes) at (x_0, y_0); that is, the axes are enclosed in an (imaginary) rectangle with opposite corners at (x_1, y_1) and (x_2, y_2) and the axes intersect at (x_0, y_0). This is illustrated in the picture below, where we have

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

shown the bounding rectangle in Cyan and the coordinates of the relevant points with respect to the current coordinate system in red:



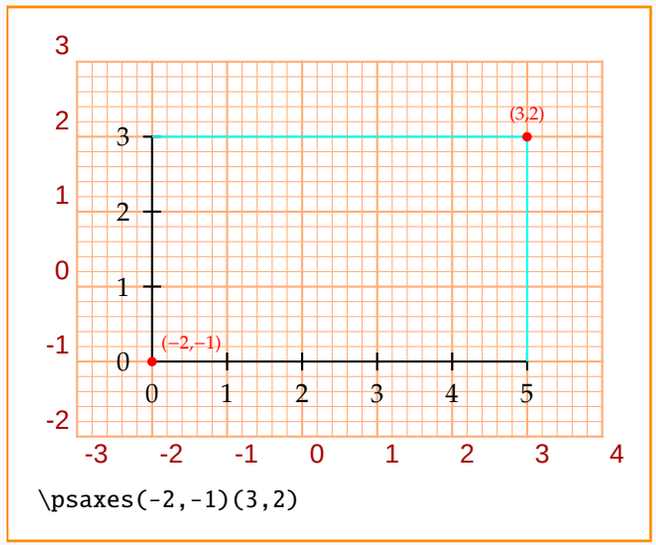
If the origin for the axes is to be the same as one of the corners of the bounding rectangle, we need not repeat it in the command: thus if we say

$$\texttt{\textbackslash psaxes}(x_1, y_1)(x_2, y_2)$$

then the origin and one corner of the bounding rectangle will be at (x_1, y_1) (and of course, the other corner will be at (x_2, y_2)), as shown below:

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



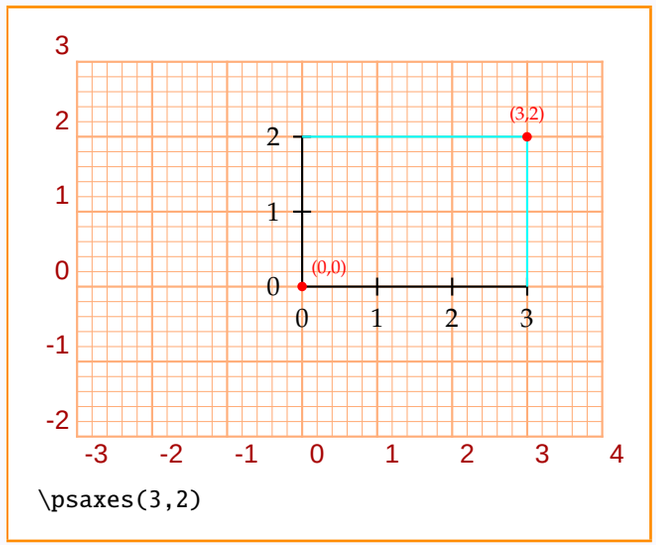
Thus \psaxes$(x_1, y_1)\,(x_2, y_2)$ is equivalent to \psaxes$(x_1, y_1)\,(x_1, y_1)\,(x_2, y_2)$. Again, if the origin for the axes is to be the same as one of the corners of the bounding rectangle and if this point is to be the same as the origin of the current coordinate system, then we need only specify only the other corner of the rectangle; that is,

\psaxes$(x, y)$

draws axes with the origin (of the axes) and one corner of the bounding rectangle at the origin of the coordinate system currently in effect, and the other corner of the rectangle at (x, y), as shown below:

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

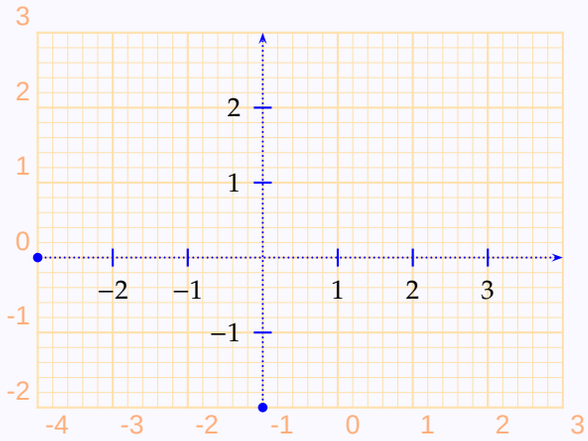
The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



Thus `\psaxes(`x, y`)` is equivalent to `\psaxes(0,0)(0,0)(`x, y`)`.

We can set various parameters such as `linecolor` within the `\psaxes` command and also specify arrow terminations. The example below illustrates some of these.

```
\begin{pspicture}(-2,-3)(3,4)
  \psaxes[linecolor=Blue,%
          linestyle=dotted,
          dotsep=1pt]%
         {*->}%
         (-1,0)(-4,-2)(3,3)
  \end{pspicture}
```



Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

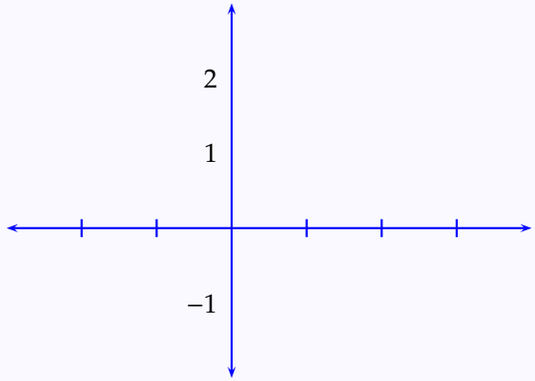
The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

We have included a background grid for reference and it is not produced by the code given. Note that the first arrowhead * is used for the tips at $(-4, 0)$ and $(0, -2)$ and the second arrowhead > is used at $(3, 0)$ and $(0, 3)$.

7.2.2. Marking the axes

The `\psaxes` command not only draws the axes, but puts equally spaced marks on the axes and names them. This is controlled by the parameters `ticks` and `labels`. Each of these has four possible values, `all` (which puts ticks or labels on both the axes, and is the default value), `x` (ticks or labels on the x-axes alone, `y` (on y-axis alone) and `none` (on neither axes). This is illustrated in the example below:

```
\begin{pspicture}(-4,-2)(3,3)
  \psaxes[linecolor=Blue,%
          ticks=x,%
          labels=y]%
         {<->}%
         (-1,0)(-4,-2)(3,3)
  \end{pspicture}
```



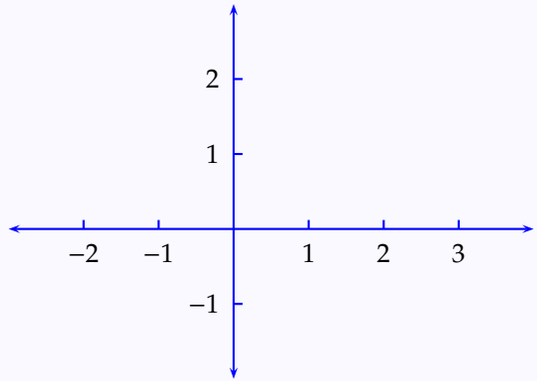
The position and size of the ticks can also be adjusted. Whether the ticks extend to either side of the origin or only to one side is determined by the parameter `tickstyle`. Its possible values are `full` (ticks extending to either side, which is the default), `top` (ticks extending only to the opposite side of the labels) and `bottom` (ticks on same side as the labels). The length of the

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

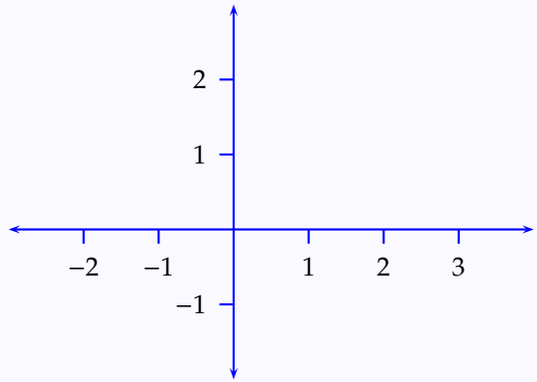
The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

ticks is controlled by the `ticksize` parameter with default value 3 pt. These are illustrated in the next two examples:

```
\begin{pspicture}(-4,-2)(3,3)
  \psaxes[linecolor=Blue,%
          tickstyle=top]%
         {<->}%
         (-1,0)(-4,-2)(3,3)
  \end{pspicture}
```




```
\begin{pspicture}(-2,-3)(3,4)
  \psaxes[linecolor=Blue,%
          tickstyle=bottom,%
          ticksize=5pt]%
         {<->}%
         (-1,0)(-4,-2)(3,3)
  \end{pspicture}
```



In some of our examples above, the origin is labeled (of course as (0, 0)), but in some others, it is not. The default behavior is to label the origin, unless the label will fall on any of the axes. We can turn off the labeling of origin (even in those cases, where it would be otherwise done), by setting the parameter `showorigin` to `false`. (By default it is `true`). Look at the example below:

Online LaTeX Tutorial

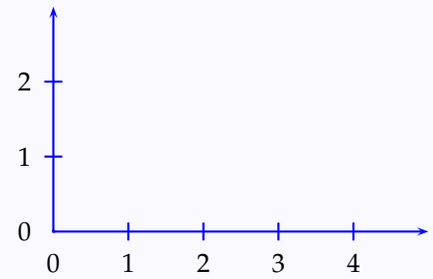
Part II – Graphics

PSTricks

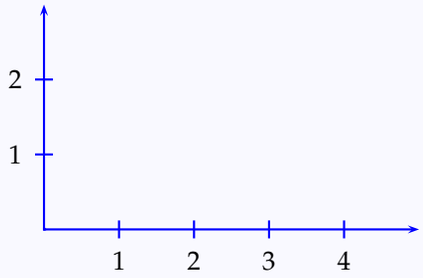
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(-2,-2)(3,2)
  \psaxes[linecolor=Blue]%
         {<->}%
         (-2,-1)(3,2)
\end{pspicture}
```




```
\begin{pspicture}(-2,-2)(3,2)
  \psaxes[linecolor=Blue,%
          showorigin=false]%
         {<->}%
         (-2,-1)(3,2)
\end{pspicture}
```



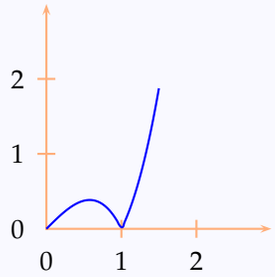
When the axes enclose a single quadrant of the plane, the labels are, by default, placed outside the quadrant (so as not to obstruct the graph within) as shown in the pictures below:

Online LaTeX Tutorial
Part II – Graphics
PSTricks

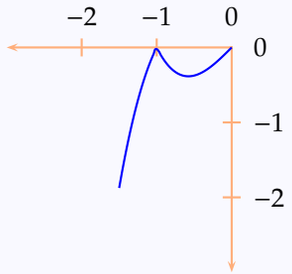
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(0,0)(3,3)
  \psaxes[linecolor=Apricot]%
         {<->}%
         (3,3)
  \psplot[plotstyle=curve,%
          linecolor=Blue]%
         {0}{1.5}%
         {x 3 exp x sub abs}
  \end{pspicture}
```




```
\begin{pspicture}(0,0)(-3,-3)
  \psaxes[linecolor=Apricot]%
         {<->}%
         (-3,-3)
  \psplot[plotstyle=curve,%
          linecolor=Blue]%
         {-1.5}{0}%
         {x 3 exp x sub abs neg}
\end{pspicture}
```



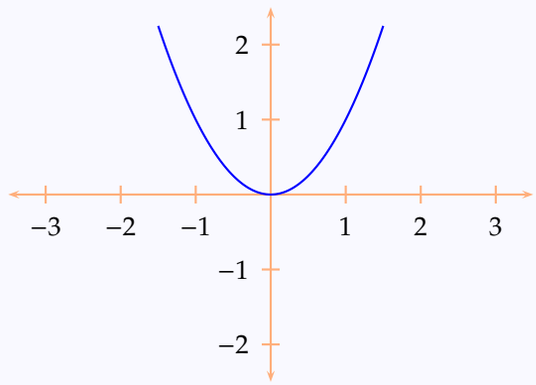
If the axes are specified by two corners lying in two different quadrants, the labels are marked *inside* the quadrant of the first corner specified. The examples below will make this clear:

Online LaTeX Tutorial
Part II – Graphics
PSTricks

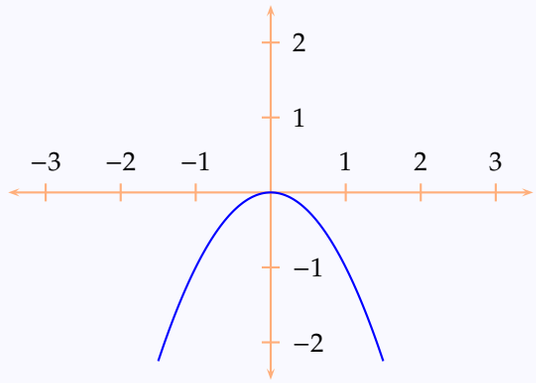
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(-3,-3)(3,3)
  \psaxes[linecolor=Apricot]%
         {<->}%
         (0,0)(-3.5,-2.5)(3.5,2.5)
  \psplot[plotstyle=curve,%
          linecolor=Blue]%
         {-1.5}{1.5}%
         {x 2 exp}
\end{pspicture}
```




```
\begin{pspicture}(-3,-3)(3,3)
  \psaxes[linecolor=Apricot]%
         {<->}%
         (0,0)(3.5,2.5)(-3.5,-2.5)
  \psplot[plotstyle=curve,%
          linecolor=Blue]%
         {-1.5}{1.5}%
         {x 2 exp neg}
  \end{pspicture}
```



Plotting Tricks

Online LaTeX Tutorial

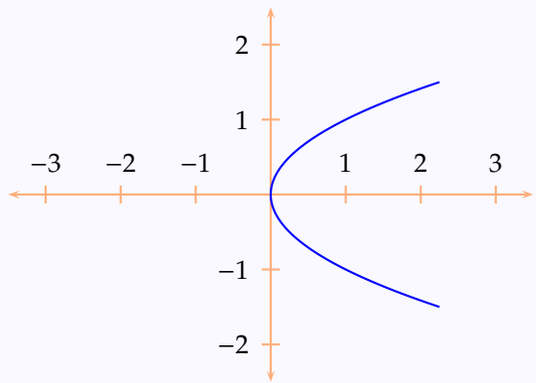
Part II – Graphics

PSTricks

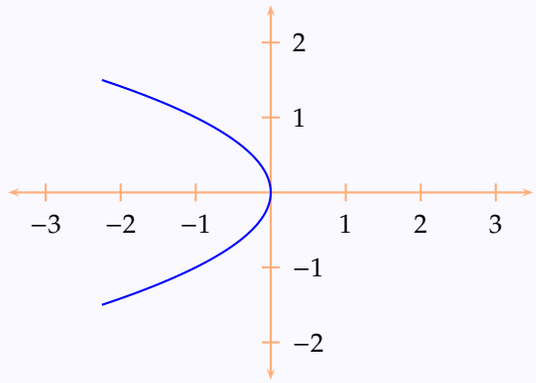
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(-3,-3)(3,3)
  \psaxes[linecolor=Apricot]%
         {<->}%
         (0,0)(-3.5,2.5)(3.5,-2.5)
  \parametricplot[plotstyle=curve,%
                  linecolor=Blue]%
                 {-1.5}{1.5}%
                 {t 2 exp t}
 \end{pspicture}
```




```
\begin{pspicture}(-3,-3)(3,3)
  \psaxes[linecolor=Apricot]%
         {<->}%
         (0,0)(3.5,-2.5)(-3.5,2.5)
 \parametricplot[plotstyle=curve,%
                  linecolor=Blue]%
                 {-1.5}{1.5}%
                 {t 2 exp neg t}
 \end{pspicture}
```



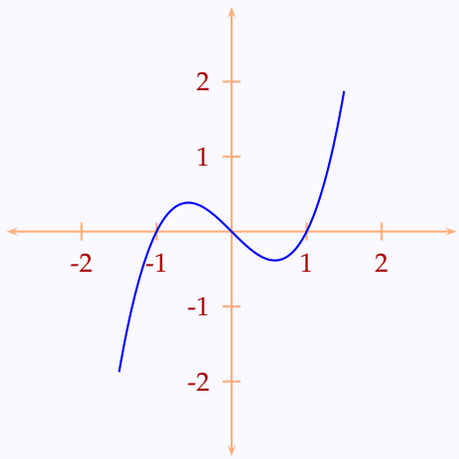
The distance between labels and axes can be controlled by the `labelsep` parameter. The style of the labels (such as font, color and so on) is controlled by the command `\pshlabel`, for horizontal (that is, along the x-axis) and `\psvlabel`, for the vertical (that is, along the y-axis). The complete code for the picture at the beginning of this section can be now given:

Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(-3,-3)(3,3)
  \renewcommand{%
    \pshlabel}[1]{%
    \color{Mahogany}#1}
  \renewcommand{%
    \psvlabel}[1]{%
    \color{Mahogany}#1}
  \psaxes[linecolor=Apricot]%
         {<->}%
         (0,0)(-3,-3)(3,3)
  \psplot[plotstyle=curve,%
          linecolor=Blue]%
         {-1.5}{1.5}%
         {x 3 exp x sub}
\end{pspicture}
```

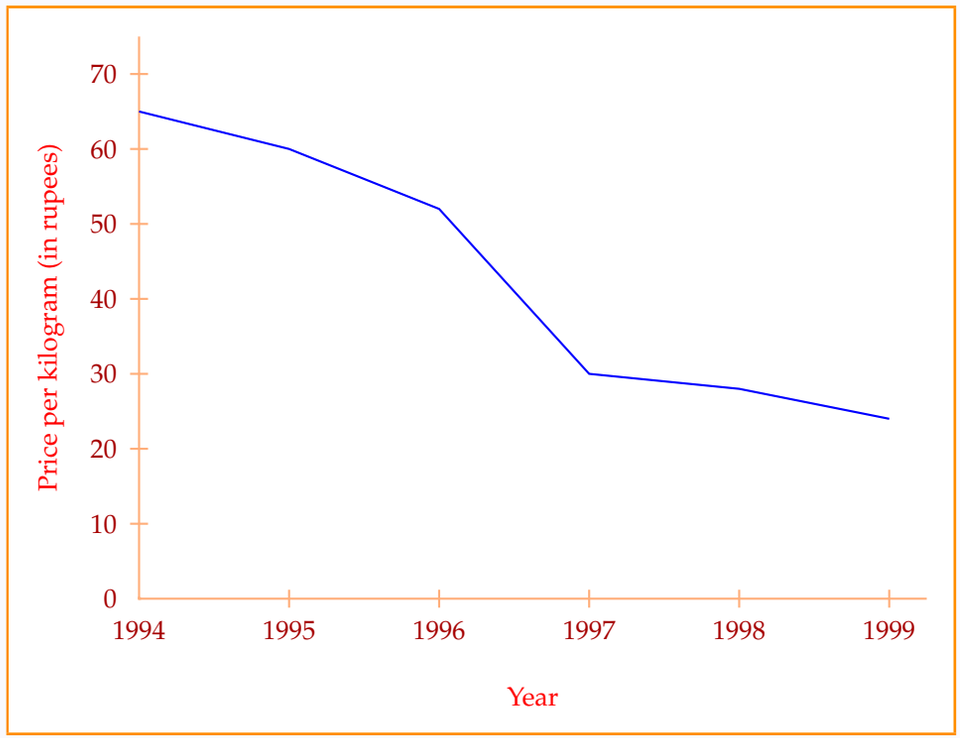


The command \psaxes, by default, produces equally spaced ticks on each axes at either side of the origin and labels them with consecutive integers. But sometimes, we do need axes marked differently as in the picture below:

Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



Here, we need the origin to be marked with the numbers 1994 (on the x-axis) and 0 (on the y-axis), the x-axis to be marked with the numbers $1994, 1995, \ldots$ spaced at 2 cm and the y-axis marked with $0, 10, 20, \ldots$ spaced at 1 cm. All these can be done automatically, by setting certain parameters of the `\psaxes` command. These are shown in the table below:

parameter	meaning	default
Ox	Label at the origin on the x-axis	0
Oy	Label at the origin on the y-axis	0
Dx	increment in labels for the x-axis	1
Dy	increment in labels for the y-axis	1
dx	distance between ticks along the x-axis	`Dx×\psxunit`
dy	distance between ticks along the y-axis	`Dy×\psyunit`

Online LaTeX Tutorial
Part II – Graphics
PSTricks

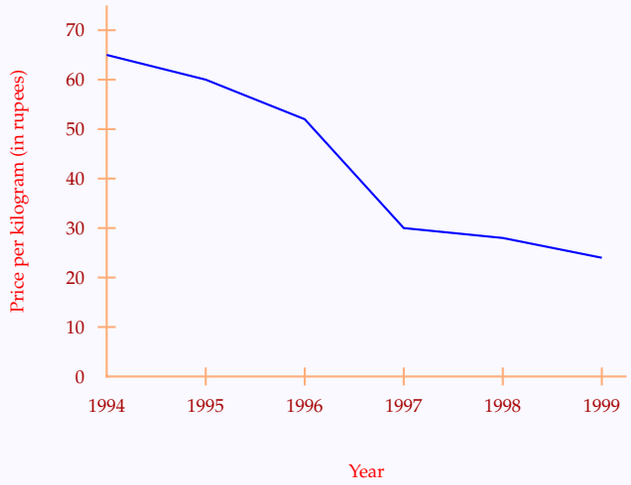
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to
tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

Note that the entries in the first column of the first row are "oh" (uppercase) and *not zero*. Also, `\psxunit` and `\psyunit` (in the last row) are TEX dimensions specifying the units along the x-axis and the y-axis.

Thus the picture above (actually, a scaled down version—to get the picture itself, simply remove the `\psset` command and all the `\scrtiptsize` commands) is produced as shown below:

```
\psset{unit=0.66}
 \renewcommand{%
   \pshlabel}[1]{%
    \scriptsize\color{Mahogany}#1}
 \renewcommand{%
  \psvlabel}[1]{%
   \scriptsize\color{Mahogany}#1}
 \begin{pspicture}(0,0)(10,7)
   \psaxes[linecolor=Apricot,%
           Ox=1994,Oy=0,%
           Dx=1,Dy=10,%
           dx=2,dy=1]%
          (10.5,7.5)
  \psline[linecolor=Blue]%
         (0,6.5)(2,6)(4,5.2)%
         (6,3)(8,2.8)(10,2.4)
  \uput[d](5.25,-1.5){%
    \scriptsize\color{Red} Year}
  \uput[l](-1.5,3.75){%
    \rput{90}{%
      \scriptsize\color{Red}%
       Price per kilogram (in rupees)}}
\end{pspicture}
```



These parameters can be set to non-integer values also. Thus the plot of $f(x) = \sin x$ with x measured in radians can be drawn as below:

Online LaTeX Tutorial

Part II – Graphics

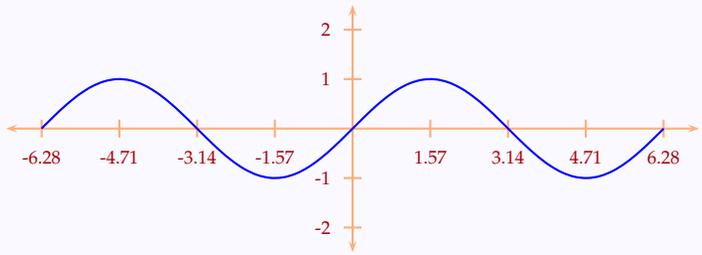
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

http://www.tug.org.in

```
\psset{unit=0.66}
\renewcommand{%
  \pshlabel}[1]{%
    \scriptsize\color{Mahogany}#1}
renewcommand{%
  \psvlabel}[1]{%
    \scriptsize\color{Mahogany}#1}
\begin{pspicture}(-7,-2)(7,2)
  \psaxes[linecolor=Apricot,%
          Dx=1.57,dx=1.57]%
          {<->}%
          (0,0)(-7,-2.5)(7,2.5)
  \psplot[plotstyle=curve,%
          linecolor=Blue]%
         {-6.2832}{6.2832}%
         {57.2958 x mul sin}
\end{pspicture}
```



In using such non-integer values for these parameters, we must keep in mind that `0x` and `Dx` or `0y` and `Dy` must have the same number of digits to the right of the decimal point. However, even if `Dx` and `Dy` are not integers, `0x` and `0y` can be so.

Finally, using the `axesstyle` parameter, we can choose to have a frame instead of axes (if the axes enclose a single quadrant) or no axes at all (but with ticks and labels). The possible values of this parameter are `axes` (which is the default), `frame` or `none`. These possibilities are illustrated in the examples below:

Online LaTeX Tutorial

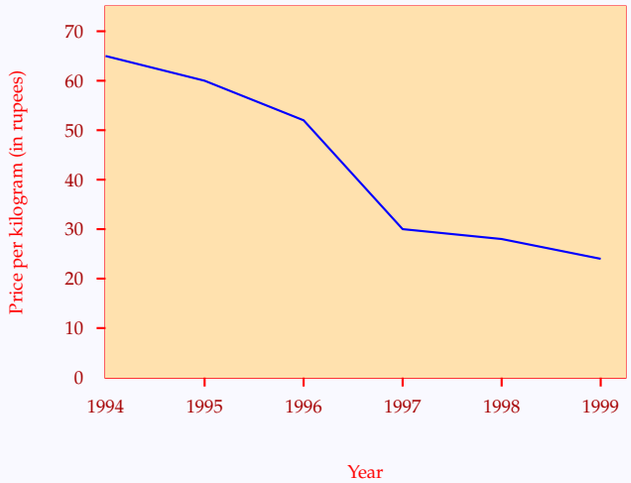
Part II – Graphics

PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\psset{unit=0.66}
 \renewcommand{%
   \pshlabel}[1]{%
    \scriptsize\color{Mahogany}#1}
 \renewcommand{%
  \psvlabel}[1]{%
   \scriptsize\color{Mahogany}#1}
\definecolor{LightApricot}%
  {cmyk}{0,0.27,0.47,0}
\begin{pspicture}(0,0)(10,7)
   \psaxes[axesstyle=frame,%
           fillstyle=solid,%
           fillcolor=PaleApricot,%
           linecolor=Apricot,%
           Ox=1994,Oy=0,%
           Dx=1,Dy=10,%
           dx=2,dy=1,
           tickstyle=bottom]%
          (10.5,7.5)
  \psline[linecolor=Blue]%
         (0,6.5)(2,6)(4,5.2)%
         (6,3)(8,2.8)(10,2.4)
  \uput[d](5.25,-1.5){%
    \scriptsize\color{Red} Year}
  \uput[l](-1.5,3.75){%
    \rput{90}{%
      \scriptsize\color{Red}%
       Price per kilogram (in rupees)}}
\end{pspicture}
```



Plotting Tricks

Online LaTeX Tutorial

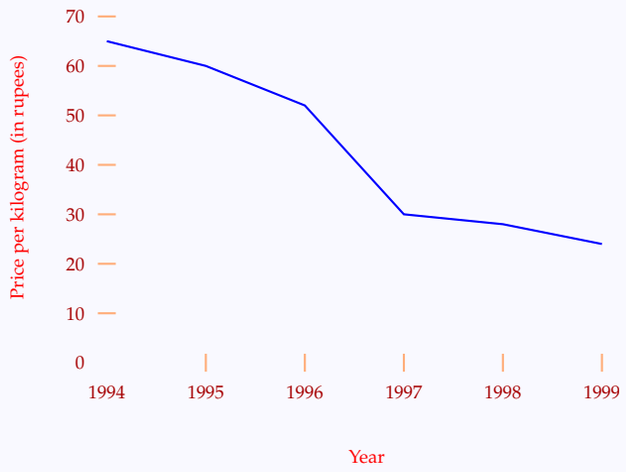
Part II – Graphics

PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\psset{unit=0.66}
 \renewcommand{%
   \pshlabel}[1]{%
    \scriptsize\color{Mahogany}#1}
 \renewcommand{%
  \psvlabel}[1]{%
   \scriptsize\color{Mahogany}#1}
 \begin{pspicture}(0,0)(10,7)
   \psaxes[axesstyle=none,%
           linecolor=Apricot,%
           Ox=1994,Oy=0,%
           Dx=1,Dy=10,%
           dx=2,dy=1]%
          (10.5,7.5)
  \psline[linecolor=Blue]%
         (0,6.5)(2,6)(4,5.2)%
         (6,3)(8,2.8)(10,2.4)
  \uput[d](5.25,-1.5){%
    \scriptsize\color{Red} Year}
  \uput[l](-1.5,3.75){%
    \rput{90}{%
      \scriptsize\color{Red}%
       Price per kilogram (in rupees)}}
\end{pspicture}
```



Online LaTeX Tutorial

Part II – Graphics

PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

7.3. Data plotting

The command `\psplot` is used to plot a curve from its algebraic equation. As mentioned above, the coordinates of points to be plotted is computed by PostScript and a curve (depending on the `plotstyle` parameter) connecting these is drawn. If instead, the coordinates of the points are to be explicitly specified, we use the `\pscurve` or `\psline` command. But, if the set of coordinates is large or is generated by some other program, then it's tedious (and inelegant) to include all these in the TEX code. The pst-plot package has some plotting commands by which a list of coordinates contained in an external file can be piped into TEX code.

For example, consider the file `data1.csv` with contents as shown below, exported from the gnumeric spreadsheet program:

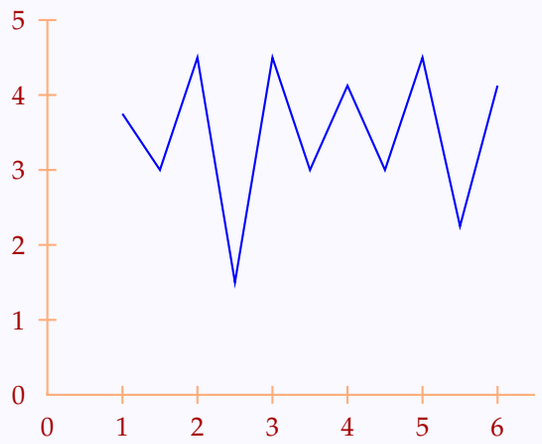
```
1,3.75
1.5,3
2,4.5
2.5,1.5
3,4.5
3.5,3
4,4.125
4.5,3
5,4.5
5.5,2.25
6,4.125
```

To plot a line graph of this data, we can use the command `\fileplot` as shown below:

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(0,0)(6.5,5)
  \renewcommand{%
    \pshlabel}[1]{%
       \color{Mahogany} #1}
  \renewcommand{%
    \psvlabel}[1]{%
      \color{Mahogany} #1}
  \psaxes[linecolor=Apricot]%
         (6.5,5)
  \fileplot[linecolor=Blue]%
           {data1.csv}
\end{pspicture}
```



In general, the command

\fileplot[*parameters*]{*filename*}

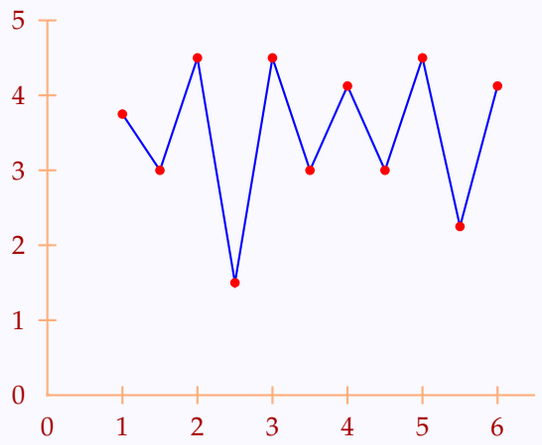
can be used to plot the graph connecting the points with coordinates listed in the file with name \textit{filename}, contained in the working directory. The file should only contain the coordinates and perhaps comments marked with % and nothing else. The numbers can be delimited by white space, commas, parentheses ( ) or curly braces, { }. (In the example above, both white spaces and commas are used).

The \fileplot command recognizes only the line, polygon and dots styles of plotting. Also, it ignores the parameters arrows, linearc and showpoints parameters. However , we can display the points plotted by including a second \fileplot with plotstyle=dots as shown below:

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(0,0)(6.5,5)
  \renewcommand{%
    \pshlabel}[1]{%
       \color{Mahogany} #1}
  \renewcommand{%
    \psvlabel}[1]{%
      \color{Mahogany} #1}
  \psaxes[linecolor=Apricot]%
         (6.5,5)
  \fileplot[linecolor=Blue]%
           {data1.csv}
  \fileplot[plotstyle=dots,%
            linecolor=Red]%
           {data1.csv}
\end{pspicture}
```



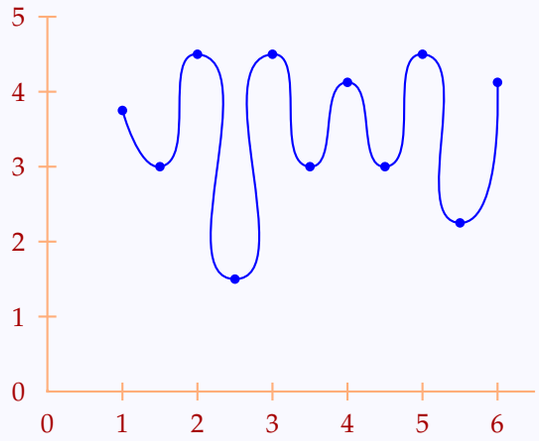
This example also illustrates an advantage of using external files to hold lists of coordinates, namely the re-usability of such lists.

Another command available for plotting data from external files is `\dataplot`. To use this, we must first name a command to read the data from the external file using the command `\readdata` and then use this new command as an argument to `\dataplot`. This is illustrated in the example below:

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(0,0)(6.5,5)
  \renewcommand{%
    \pshlabel}[1]{%
      \color{Mahogany} #1}
    \renewcommand{%
      \psvlabel}[1]{%
        \color{Mahogany} #1}
    \psaxes[linecolor=Apricot]%
           (6.5,5)
    \readdata{\picdata}{data1.csv}
    \dataplot[plotstyle=curve,%
              showpoints=true,%
             linecolor=Blue]%
             {\picdata}
\end{pspicture}
```



Note that the `\dataplot` command can accepts the `curve` style of plotting and also the `showpoints` parameter.

This command also has a facility to include coordinates directly (and then perhaps re-used) using the command `\savedata`. Thus in the last example, instead of `\readdata`...`\dataplot`, the code

```
\savedata{\grphdata}[%
    {1,3.75 1.5,3 2,4.5 2.5,1.5
     3,4.5 3.5,3 4,4.125 4.5,3
     5,4.5 5.5,2.25 6,4.125}]
\dataplot[plotstyle=curve,%
          showpoints=true,%
          linecolor=Blue]%
         {\grphdata}
```

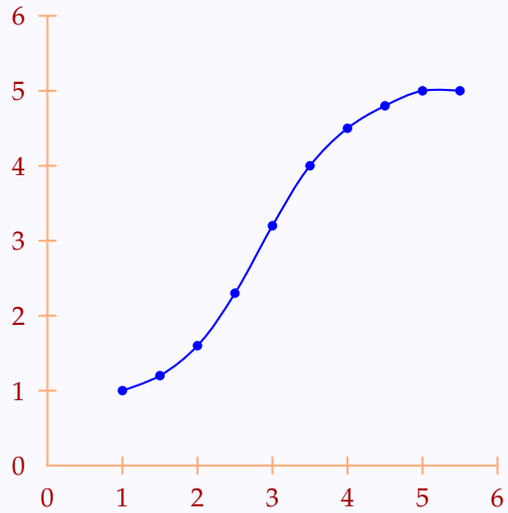
can also be used to produce the same picture.

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

The last command we describe for data plotting is `\listplot`. Here, we can give the data as coordinate pairs, *separated only by spaces*, as an argument to the command itself as in the example below:

```
\begin{pspicture}(0,0)(5,6)
      \renewcommand{%
        \pshlabel}[1]{%
          \color{Mahogany} #1}
      \renewcommand{%
        \psvlabel}[1]{%
          \color{Mahogany} #1}
        \psaxes[linecolor=Apricot]%
               (6,6)
        \listplot[plotstyle=curve,%
               showpoints=true,%
               linecolor=Blue]%
               {1 1
                1.5 1.2
                2 1.6
                2.5 2.3
                3 3.2
                3.5 4
                4 4.5
                4.5 4.8
                5 5
                5.5 5}
    \end{pspicture}
```



Here the data list is first expanded by TEX and then by PostScript. So, we can use PostScript commands to manipulate the data within this command. For example, we can draw the "inverse" graph of the above graph by interchanging the coordinates of each points using the exch command of PostScript.

Online LATEX Tutorial

Part II – Graphics

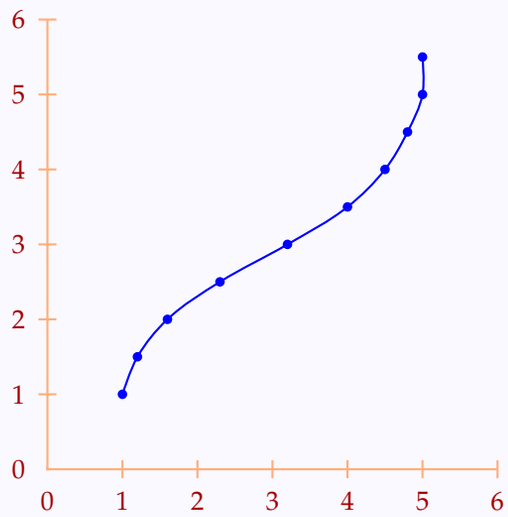
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

http://www.tug.org.in

```
\begin{pspicture}(0,0)(5,6)
      \renewcommand{%
        \pshlabel}[1]{%
          \color{Mahogany} #1}
      \renewcommand{%
        \psvlabel}[1]{%
          \color{Mahogany} #1}
        \psaxes[linecolor=Apricot]%
               (6,6)
        \listplot[plotstyle=curve,%
               showpoints=true,%
               linecolor=Blue]%
               {1 1 exch
                1.5 1.2 exch
                2 1.6 exch
                2.5 2.3 exch
                3 3.2 exch
                3.5 4 exch
                4 4.5 exch
                4.5 4.8 exch
                5 5 exch
                5.5 5 exch}
    \end{pspicture}
```



In fact, the "list" given as an argument to `\listplot` may even be a PostScript program which generates a sequence of coordinate pairs. Again, as in the case of `\dataplot`, we can use the commands `\savedata` or `\readdata` with `\listplot` also, as illustrated below:

Online LaTeX Tutorial

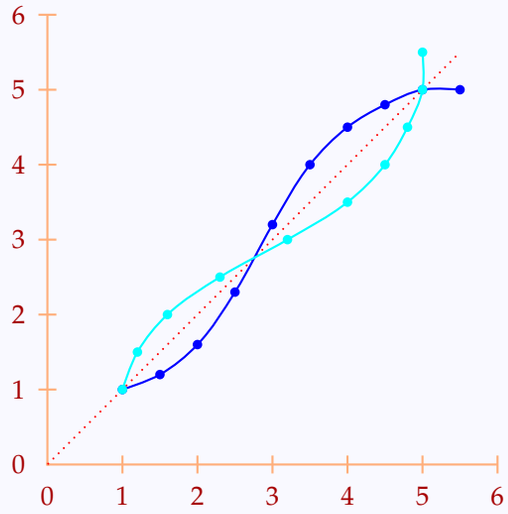
Part II – Graphics

PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

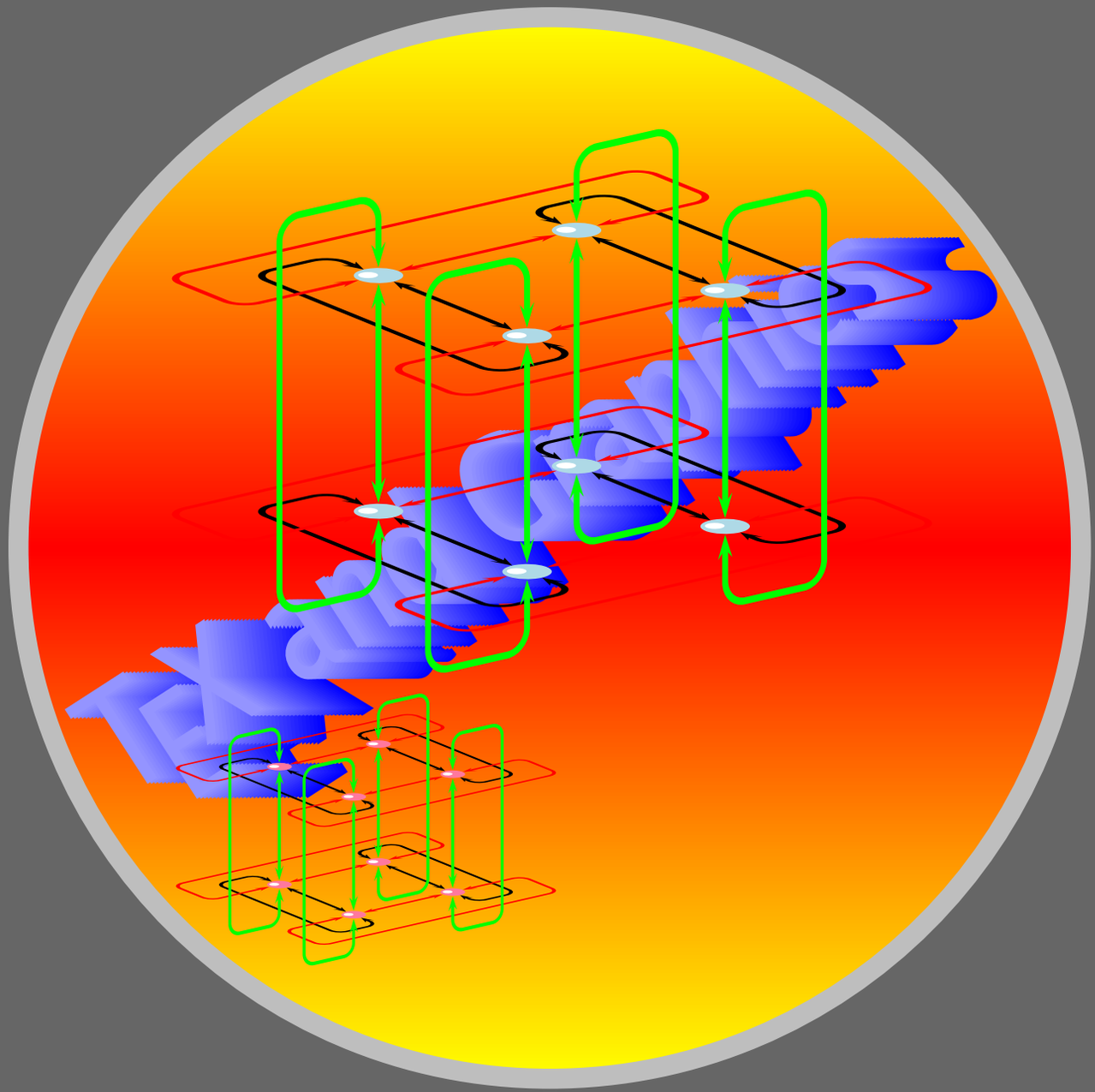
```
\savedata{\dirdata}[1 1 1.5 1.2 2 1.6 2.5 2.3
                    3 3.2 3.5 4 4 4.5 4.5 4.8
                    5 5 5.5 5]
\savedata{\invdata}[1 1 exch 1.5 1.2 exch
                    2 1.6 exch 2.5 2.3 exch
                    3 3.2 exch 3.5 4 exch
                    4 4.5 exch 4.5 4.8 exch
                    5 5 exch 5.5 5 exch]
\begin{pspicture}(0,0)(5,6.5)
  \renewcommand{%
    \pshlabel}[1]{%
    \color{Mahogany} #1}
  \renewcommand{%
    \psvlabel}[1]{%
    \color{Mahogany} #1}
  \psaxes[linecolor=Apricot]%
         (6,6)
  \psline[linestyle=dotted,%
          dotsep=2pt,%
          linecolor=Red]
         (0,0)(5.5,5.5)
  \listplot[plotstyle=curve,%
            showpoints=true,%
            linecolor=Blue]%
            {\dirdata}
  \listplot[plotstyle=curve,%
            showpoints=true,%
            linecolor=Cyan]%
            {\invdata}
 \end{pspicture}
```



Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in



Custom Graphics

Simple customization
Higher level customization

Online LaTeX Tutorial

Part II – Graphics

PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to
tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

http://www.tug.org.in

8. Custom Graphics

We've seen in previous chapters the wide array of graphics objects available off-the-shelf in PSTricks. In this chapter we'll take a look at some methods of creating our own graphics objects. We've also seen how we can produce variants of such predefined objects by tweaking their default parameter settings. The first level of customization is to save such variants as new objects and then reuse them.

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to
tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

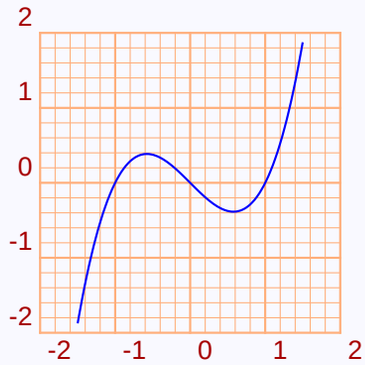
8.1. Simple customization

If a certain graphics object in PSTricks is repeatedly used with some custom parameter settings, we can save the tedium in typing in these values by saving it under a new name. (This also comes in handy to make changes in a consistent manner, if we happen to change our mind later).

8.1.1. New objects

You may have noticed that many of our examples contain a colored grid like the one below:

```
\begin{pspicture}(-2,-2)(2,2)
  \psgrid[gridcolor=Apricot,%
          gridlabelcolor=Mahogany,%
          subgridcolor=Apricot]
  \psplot[plotstyle=curve,%
          linecolor=Blue]%
         {-1.5}{1.5}{x 3 exp x sub}
\end{pspicture}
```



Since the same grid is used in all the examples, it'd be nice if we can avoid the tedium of typing all the parameter settings again and again; and its where the command `\newpsobject` comes in handy. Thus by defining

```
\newpsobject{colgrid}%
            {psgrid}%
            {gridcolor=Apricot,%
             gridlabelcolor=Mahogany,%
             subgridcolor=Apricot}
```


E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

http://www.tug.org.in

we can use \colgrid every time we need the grid of these specifications.

The syntax of the general command is:

```
\newpsobject{name}%
            {object}%
            {parameter1=value1}%
             parameter1=value1
             ............}
```

Where *name* is the name we give to the variant object (such as `colgrid` in the example above), *object* is the name of the graphics object predefined in PSTricks that we want to tweak (such as `psgrid` in the example) and *Parameter1*, *parameter2* are the parameters whose values we set by assigning values *value1*, *value2* and so on.

Once an object is defined using `\newpsobject`, we can change its parameters the usual way by enclosing the new values within square brackets. The examples below illustrates this:

```
\newpsobject{bigdots}%
            {psdots}%
            {dotsize=1 0,%
             linecolor=Red}
\begin{pspicture}(-0.5,-0.5)(1.5,1.37)
  \bigdots(0,0)(1,0)(0.5,0.866)
\end{pspicture}

\vspace{1cm}

\begin{pspicture}(-0.5,-0.5)(1.5,1.37)
  \bigdots[linecolor=Green](0,0)
  \bigdots[linecolor=Blue](1,0)
  \bigdots(0.5,0.866)
\end{pspicture}
```

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

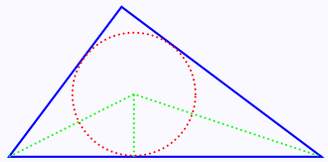
http://www.tug.org.in

8.1.2. New styles

We can also freeze certain parameter values under a new name using the command `\newpsstyle` and then use it with various graphics objects, as in the example below:

```
\newpsstyle{newdots}%
           {linestyle=dotted,%
            dotsep=1pt,%
            linecolor=Green}

\psset{unit=0.5cm}
\begin{pspicture}(0,-0.5)(8.33,4.5)
  \pspolygon[linecolor=Blue](0,0)(8.33,0)(3,4)
  \psline[style=newdots](0,0)(3.33,1.67)(8.33,0)
  \psline[style=newdots](3.33,1.67)(3.33,0)
  \pscircle[style=newdots,linecolor=Red](3.33,1.67){1.67}
  \end{pspicture}
```



The syntax of the general command is

```
\newpsstyle{name}%
           {parameter1=value1}%
            parameter1=value1
            ... ... ... ... }
```

Note that any parameter value set in `\newpsstyle` can be changed the usual way, as seen in the `\pscircle` of the above example.

Online LaTeX Tutorial

Part II – Graphics

PSTricks

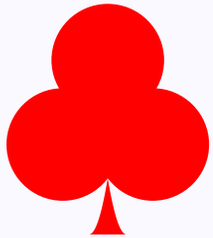
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

http://www.tug.org.in

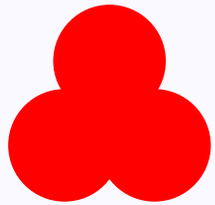
8.2. Higher level customization

Apart from creating variant objects by merely tweaking the parameter settings of predefined objects, PSTricks also allows us to create our own graphics objects by combining predefined objects in various ways, using the command `\pscustom`. Let's look at an example to see this command in action. Suppose (for whatever reason), we want to draw a red `club suit` shown below:



Drawing the top part is easy, using our `\bigdots`, of the first section.

```
\psset{unit=1.5cm}
\begin{pspicture}(-1,-1)(2,1.5)
  \bigdots(0,0)(0.8,0)(0.4,0.75)
\end{pspicture}
```



Next we draw the stem. The *shape* of the stem—it can be scaled to the needed *size* later—can be drawn using two parabolic arcs and a line segment as shown below:

We've included the arrows to show the *direction* of drawing. The code for producing this (without the arrows) is shown below:

Online LaTeX Tutorial

Part II – Graphics

PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

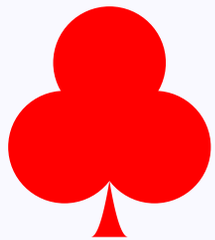
```
\begin{pspicture}(-1,-1)(1,0)
  \psplot{-1}{0}{x 1 add 2 exp 1 sub}
  \psplot{0}{1}{x 1 sub 2 exp 1 sub}
  \psline(1,-1)(-1,-1)
\end{pspicture}
```

So far, so good; but how do we *color* this? This is where `\pscustom` steps in. Look at the code and the output shown below:

```
\begin{pspicture}(-1,-1)(1,0)
  \pscustom[fillstyle=solid,%
            fillcolor=Red,%
            linestyle=none]{%
             \psplot{-1}{0}{x 1 add 2 exp 1 sub}
             \psplot{0}{1}{x 1 sub 2 exp 1 sub}
             \psline(1,-1)(-1,-1)}
\end{pspicture}
```

Now it's only a question of scaling the stem we've created and pasting at the bottom of the circles, to make our picture:

```
\psset{unit=1.5cm}
\begin{pspicture}(-1,-1)(2,2)
  \bigdots(0,0)(0.8,0)(0.4,0.75)
  \rput(0.4,-0.3){%
    \pscustom[xunit=0.25cm,%
              yunit=0.75cm,%
              linestyle=none,%
              fillstyle=solid,%
              fillcolor=Red]{%
                \psplot{-1}{0}{x 1 add 2 exp 1 sub}
                \psplot{0}{1}{x 1 sub 2 exp 1 sub}
                \psline(1,-1)(-1,-1)}}
\end{pspicture}
```



Online LaTeX Tutorial

Part II – Graphics

PSTricks

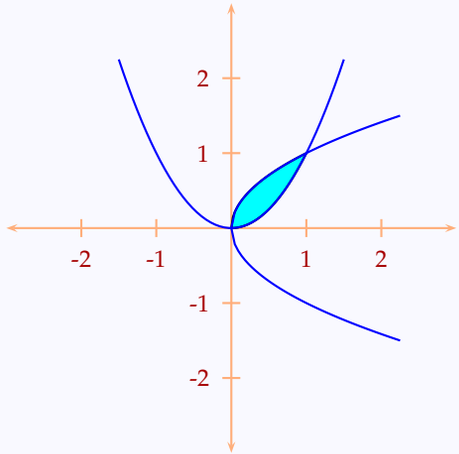
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

http://www.tug.org.in

Let's now take a closer look at the \pscustom command. In the above example, this command helped us to create our "stem" as a new graphics object, which could be subjected to the various parameter settings, as with a regular PSTricks object. This is highly useful in highlighting the region between two curves as in the example below:

```
\renewcommand{\pshlabel}[1]{\small\color{Mahogany}#1}
\renewcommand{\psvlabel}[1]{\small\color{Mahogany}#1}
\begin{pspicture}(-3,-3)(3,3)
  \colaxes(0,0)(-3,-3)(3,3)
  \pscustom[fillstyle=solid,fillcolor=Cyan]{%
    \psplot{0}{1}{x 2 exp}
    \psplot{1}{0}{x sqrt}}
  \psset{linecolor=Blue}
  \psplot{-1.5}{1.5}{x 2 exp}
  \psplot{0}{2.25}{x sqrt}
  \psplot{0}{2.25}{x sqrt neg}
\end{pspicture}
```

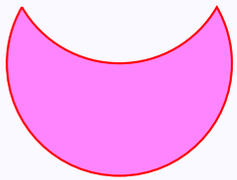


Note that \colaxes is not a native PSTricks command but is set by

```
\newpsobject{colaxes}{psaxes}{linecolor=Apricot,arrows=<->}
```

8.2.1. Of paths and points

As another example, suppose we want to draw a crescent like the one below:



E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

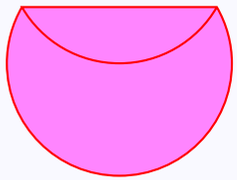
The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

http://www.tug.org.in

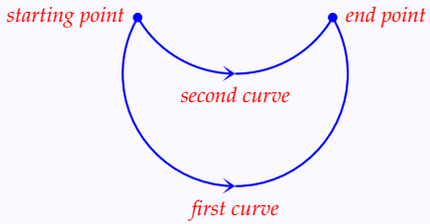
Armed with our (little) knowledge of the \pscustom, suppose we try

```
\begin{pspicture}(-1.5,-1.5)(1.5,1.5)
  \pscustom[fillstyle=solid,%
            fillcolor=Lavender,%
            linecolor=Red]{%
              \psarc(0,0){1.5}{150}{30}
              \psarc(0,1.5){1.5}{210}{330}}
\end{pspicture}
```

We get



To analyze this seemingly strange behavior, we must look at the way PSTricks (more precisely, PostScript) draws pictures. First recall that the command \psarc draws an arc of a circle in the *counter-clockwise direction*, so that our code above give instructions to draw curves as shown below:



In PostScript parlance, a curve is called a *path* and once a path is instructed to be drawn, the point where the path ends is called *current point*. So, when we say

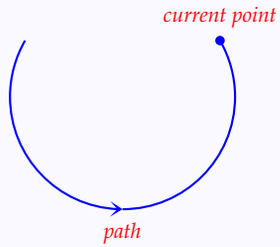
```
\psarc(0,0){1.5}{150}{30}
```


E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

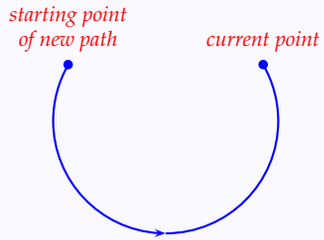
The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

http://www.tug.org.in

this is the state we are in:



The starting point of the second path to be drawn by `\psarc(0,1.5){1.5}{210}{330}` is the beginning of the first path:



The point to note (no pun!) is that here, the beginning of the new path is *not* the current point. Now comes the rub: *by default*, `\pscustom` *joins the current point with the beginning of a new path*. So

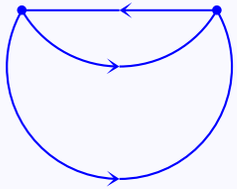
```
\begin{pspicture}(-1.5,-1.5)(1.5,1.5)
  \pscustom[linecolor=Blue]{%
    \psarc(0,0){1.5}{150}{30}
    \psarc(0,1.5){1.5}{210}{330}}
\end{pspicture}
```

produces (without the arrows)

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

http://www.tug.org.in



(Do you see why our earlier construction with `\pscustom` worked? There we had three paths, *with the beginning of each path the same as the current point*).

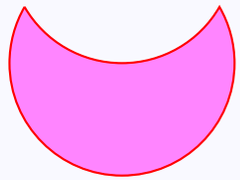
So, what's the way out? PSTricks also has a command `\psarcn` ("n" for negative) which draws the arc of a circle in the *clockwise* direction. Thus `\psarc(0,0){1.5}{210}{330}` and `\psarcn(0,0){1.5}{330}{210}` produce the same curve, but *drawn* in opposite directions:

`\psarc(0,0){1.5}{210}{330}`

`\psarcn(0,0){1.5}{330}{210}`

Now we can easily draw the crescent:

```
\begin{pspicture}(-1.5,-2)(1.5,0.5)
  \pscustom[fillstyle=solid,%
            fillcolor=Lavender,%
            linecolor=Red]{%
              \psarc(0,0){1.5}{150}{30}
              \psarcn(0,1.5){1.5}{330}{210}}
\end{pspicture}
```



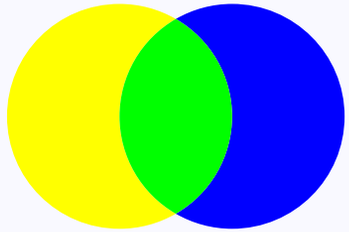
Here's another example, using `\psarcn`

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

http://www.tug.org.in

```
\psset{unit=1.5cm}
\begin{pspicture}(0,-1)(3,1)
  \psset{linestyle=none,fillstyle=solid}
  \pscustom[fillcolor=Yellow]{%
    \psarc(1,0){1}{60}{-60}
    \psarcn(2,0){1}{240}{120}}
  \pscustom[fillcolor=Blue]{%
    \psarc(2,0){1}{240}{120}
    \psarcn(1,0){1}{60}{-60}}
  \pscustom[fillcolor=Green]{%
    \psarc(1,0){1}{-60}{60}
    \psarc(2,0){1}{120}{240}}
 \end{pspicture}
```



The (default) behavior of PSTricks in joining the current point to the beginning of the new path can sometimes be of advantage. For example, in our construction of a "stem" earlier, we need not explicitly mention the "base", using`\psline`, if the two curves are specified to be drawn in suitable directions:

```
\begin{pspicture}(-1,-1)(1,0)
  \pscustom[linestyle=none,%
            fillstyle=solid,%
            fillcolor=Red]{%
              \psplot{0}{-1}{x 1 add 2 exp 1 sub}
              \psplot{1}{0}{x 1 sub 2 exp 1 sub}}
\end{pspicture}
```

8.2.2. Current point again

Another feature to note is that the paths drawn by `\psline` and `\pscurve` make use of the current point as the first point; so also does `\psplot` with

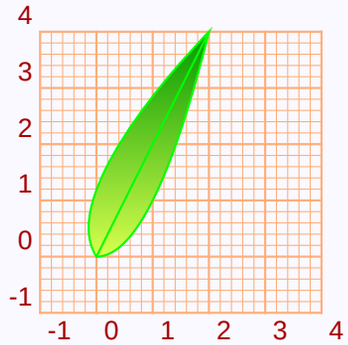
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

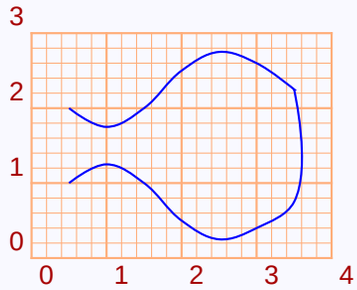
http://www.tug.org.in

plotstyle=curve and \psbezier with only three points specified. Look at these examples:

```
\psset{unit=0.75cm}
\begin{pspicture}(-1,-1.5)(4,4.5)
  \colgrid(-1,-1)(4,4)
  \pscustom[linecolor=Green,%
            fillstyle=gradient,%
            gradbegin=OliveGreen,%
            gradend=GreenYellow,%
            gradmidpoint=1]{%
              \psline(0,0)(2,4)
              \psplot[plotstyle=curve]{0}{2}{x 2 exp}}
\end{pspicture}
```




```
\begin{pspicture}(0,-0.5)(4,3.5)
  \colgrid(0,0)(4,3)
  \pscustom[linecolor=Blue]{%
    \pscurve(0.5,2)(1,1.75)(1.5,2)(2,2.5)%
            (2.5,2.75)(3,2.6)(3.5,2.25)
    \pscurve(3.5,0.75)(3,0.4)(2.5,0.25)%
            (2,0.5)(1.5,1)(1,1.25)(0.5,1)}
\end{pspicture}
```



8.2.3. Lifting the pen

The behavior of the commands such as \psline toward the current point can be modified by setting the parameter liftpen, which can take values 0, 1 or 2. The default value is 0 and this corresponds to the (default) behavior of

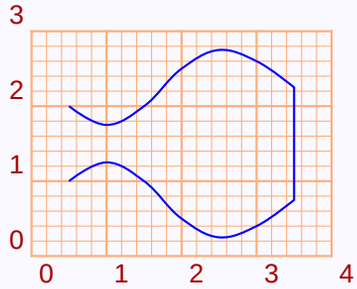
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

http://www.tug.org.in

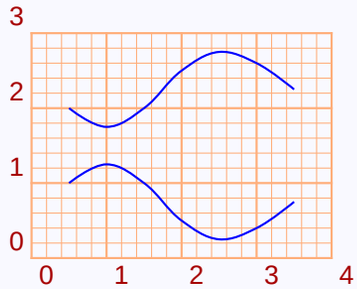
treating the current point as the first point. With `liftpen=1`, the current point is not treated as the first point, but a line is drawn from the current point to the first point of the path, as with the other path drawing commands:

```
\begin{pspicture}(0,-0.5)(4,3.5)
  \colgrid(0,0)(4,3)
  \pscustom[linecolor=Blue]{%
    \pscurve(0.5,2)(1,1.75)(1.5,2)(2,2.5)%
            (2.5,2.75)(3,2.6)(3.5,2.25)
    \pscurve[liftpen=1]%
            (3.5,0.75)(3,0.4)(2.5,0.25)%
            (2,0.5)(1.5,1)(1,1.25)(0.5,1)}
\end{pspicture}
```



With `liftpen=2`, the current point is not taken as the first point, nor is a line drawn from the current point to the first point of the path:

```
\begin{pspicture}(0,-0.5)(4,3.5)
  \colgrid(0,0)(4,3)
  \pscustom[linecolor=Blue]{%
    \pscurve(0.5,2)(1,1.75)(1.5,2)(2,2.5)%
            (2.5,2.75)(3,2.6)(3.5,2.25)
    \pscurve[liftpen=2]%
            (3.5,0.75)(3,0.4)(2.5,0.25)%
            (2,0.5)(1.5,1)(1,1.25)(0.5,1)}
\end{pspicture}
```

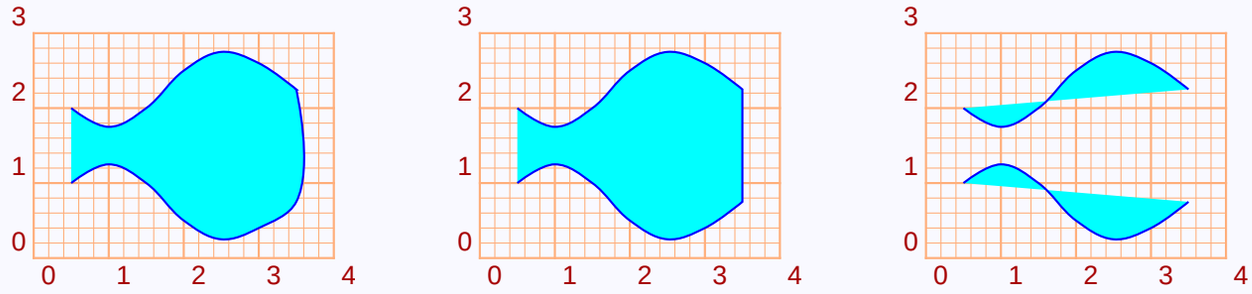


The pictures below show the last three pictures above with the parameter settings `\fillstyle=solid` and `\fillcolor=Cyan` for the `\pscustom` command:

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

http://www.tug.org.in



The last picture may need some explanation; note that when we instruct an "open" curve to be "filled", then the first and the last points are joined to make a closed region, which is then filled.

8.2.4. Other tricks

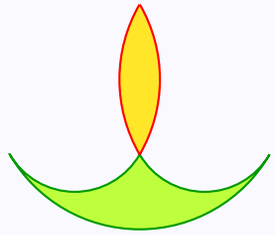
Before discussing some other techniques, we must take a closer look at the way PostScript handles graphics. When we give instructions to draw a path in PostScript, it is not immediately drawn; we will also have to give instructions to draw it, or *stroke* it in PostScript terminology. Again, we will have to give explicit commands to *fill* a region. The command `\pscustom` automatically does the stroking and filling at the end, subject to the various `linestyle` and `fillstyle` options. But within `\pscustom` we ourselves can stroke or fill using the commands `\stroke` or `\fill`. We can also use the command `\newpath` to clear the current path and current point. Look at the example below:

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

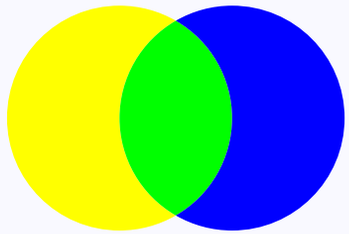
http://www.tug.org.in

```
\begin{pspicture}(-2,-2)(2,2)
  \pscustom[linestyle=none]{%
    \psarcn(0.865,-0.5){1}{330}{210}
    \psarcn(-0.865,-0.5){1}{330}{210}
    \psarc(0,0){2}{210}{330}
    \fill[fillstyle=solid,fillcolor=SpringGreen]
    \stroke[linestyle=solid,linecolor=OliveGreen]
    \newpath
    \psarc(1.73,0){2}{150}{210}
    \psarc(-1.73,0){2}{-30}{30}
    \fill[fillstyle=solid,fillcolor=Goldenrod]
    \stroke[linestyle=solid,linecolor=Red]}
\end{pspicture}
```



Here's one of our earlier examples, coded using a single \pscustom instead of the three used earlier:

```
\psset{unit=1.5cm}
\begin{pspicture}(0,-1)(3,1)
  \pscustom[linestyle=none]{%
    \psarc(1,0){1}{60}{-60}
    \psarcn(2,0){1}{240}{120}
    \fill[fillstyle=solid,fillcolor=Yellow]
    \newpath
    \psarc(1,0){1}{-60}{60}
    \psarc(2,0){1}{120}{240}
    \fill[fillstyle=solid,fillcolor=Green]
    \newpath
    \psarc(2,0){1}{240}{120}
    \psarcn(1,0){1}{60}{-60}
    \fill[fillstyle=solid,fillcolor=Blue]}
\end{pspicture}
```

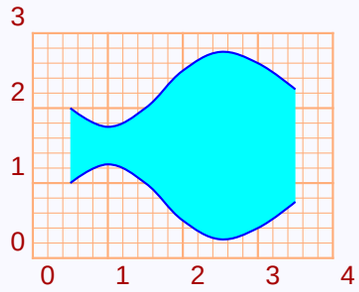


Now suppose we want to produce something like this:

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

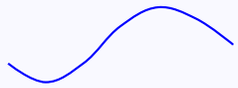
The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

http://www.tug.org.in

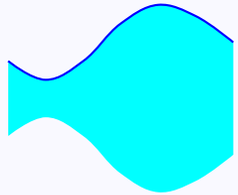


As the previous examples show, tweaking the `liftpen` values is of no use here, since in all these the curves are connected, one way or the other. Let's first list what we want to do:

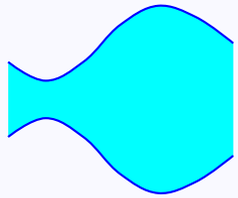
1. Draw the first curve



2. Give instructions to draw the second curve, with `liftpen=1`, without actually stroking it, and instructions to fill the region between the curves



3. Draw the the second curve with `liftpen=2`, but no filling.



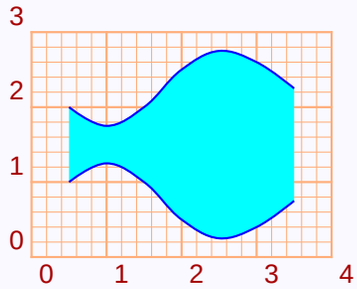
We know how to do Steps 1 and 3 above; it's Step 2 that is tricky; we use the pair of commands `\gsave...\grestore` to do the job. Let's see them in action:

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

http://www.tug.org.in

```
\begin{pspicture}(0,-1)(4,3)
  \colgrid(0,0)(4,3)
  \pscustom[linecolor=Blue]{%
    \pscurve(0.5,2)(1,1.75)(1.5,2)(2,2.5)%
            (2.5,2.75)(3,2.6)(3.5,2.25)
    \gsave
    \pscurve[liftpen=1]%
            (3.5,0.75)(3,0.4)(2.5,0.25)%
            (2,0.5)(1.5,1)(1,1.25)(0.5,1)
    \fill[fillstyle=solid,%
          fillcolor=Cyan]
    \grestore
    \pscurve[liftpen=2]%
            (3.5,0.75)(3,0.4)(2.5,0.25)%
            (2,0.5)(1.5,1)(1,1.25)(0.5,1)}
 \end{pspicture}
```



As can be seen from this code, the implementation of Step 2 is done within the `\gsave...\grestore` command. We've noted that PostScript strokes or fills a path only on explicit instructions to do so and `\pscustom` automatically strokes or fills a path at the end. Within the commands `\gsave...\grestore` however, `\pscustom` does not stroke or fill a path, unless told. (In PostScript, the command `gsave` saves the *current graphics state*, such as the current path, current point, color, linewidth and so on to a *stack* and `grestore` restores the saved state.) Thus in the example above, the code within this command pair does exactly what we want to do in Step 2 above.

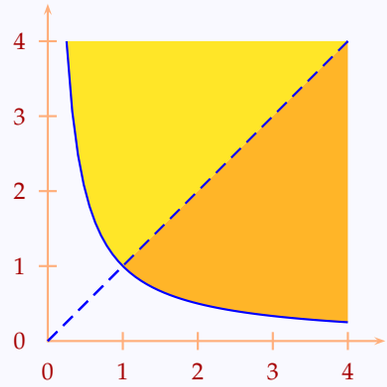
Here's another example using `\gsave...\grestore`

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, India

http://www.tug.org.in

```
\renewcommand{\pshlabel}[1]{\small\color{Mahogany}#1}
\renewcommand{\psvlabel}[1]{\small\color{Mahogany}#1}
\begin{pspicture}(0,0)(4.5,4.5)
  \colaxes(4.5,4.5)
  \pscustom[linecolor=Blue,%
            linestyle=dashed]{%
              \psline(0,0)(4,4)
              \stroke
              \gsave
                \psplot[liftpen=1]%
                       {0.25}{1}{1 x div}
                \fill[fillstyle=solid,%
                      fillcolor=Goldenrod]
              \grestore
              \gsave
                \psplot[liftpen=1]%
                       {4}{1}{1 x div}
                \fill[fillstyle=solid,%
                      fillcolor=Dandelion]
              \grestore}
    \psplot[linecolor=Blue]%
           {0.25}{4}{1 x div}
  \end{pspicture}
```



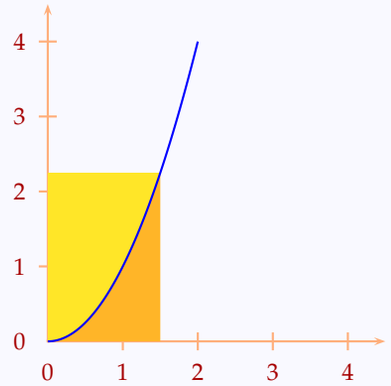
Within \pscustom we can also use the command \moveto to move the current point to a point with specified coordinates.

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

http://www.tug.org.in

```
\renewcommand{\pshlabel}[1]{\small\color{Mahogany}#1}
\renewcommand{\psvlabel}[1]{\small\color{Mahogany}#1}
\begin{pspicture}(0,0)(4.5,4.5)
  \colaxes(4.5,4.5)
  \pscustom[linecolor=Blue]{%
    \psplot{0}{1.5}{x 2 exp}
    \gsave
      \psline(0,2.25)
      \fill[fillstyle=solid,%
            fillcolor=Goldenrod]
    \grestore
    \gsave
      \psline(1.5,0)
      \fill[fillstyle=solid,%
            fillcolor=Dandelion]
    \grestore
    \moveto(1.5,2.25)
    \psplot{1.5}{2}{x 2 exp}}
  \end{pspicture}
```



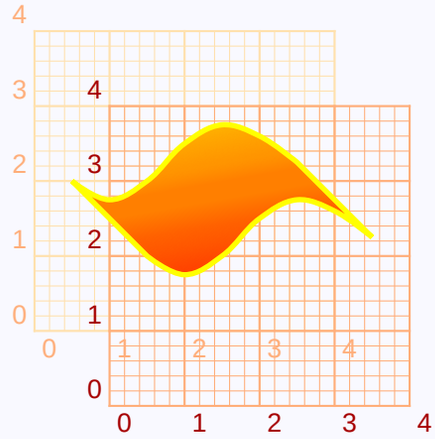
Within \pscustom, we can also transform the coordinate system in certain ways. For example, the command \translate can be used to shift the coordinate axes (parallel to themselves) to a new point, as in the example below:

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

http://www.tug.org.in

```
\begin{pspicture}(0,-1.5)(5,4.5)
  \pscustom[style=mystyle]{%
    \pscurve(0.5,2)(1,1.75)(1.5,2)(2,2.5)%
            (2.5,2.75)(3,2.6)(3.5,2.25)
    \translate(1,-1)
    \pscurve[liftpen=1]%
       (3.5,2.25)(3,2.6)(2.5,2.75)%
       (2,2.5)(1.5,2)(1,1.75)(0.5,2)
    \closepath}
\end{pspicture}
```



Here, the style named `mystyle` is set as

```
\newpsstyle{mystyle}%
           {linestyle=solid,%
            linewidth=2pt,%
            linecolor=Yellow,%
            fillstyle=gradient,%
            gradbegin=Yellow,%
            gradend=Red,%
            gradangle=10,%
            gradmidpoint=1}
```

We have also drawn two coordinate grids, the lighter one showing the original system and the darker one showing the changed system, the codes of which are not shown in the example.

We can also use (within `\pscustom`) the command `\scale` to scale the coordinate axes. It takes two numbers as arguments, the x-axis scaled by the first number and the y-axis scaled by the second number, as in this example:

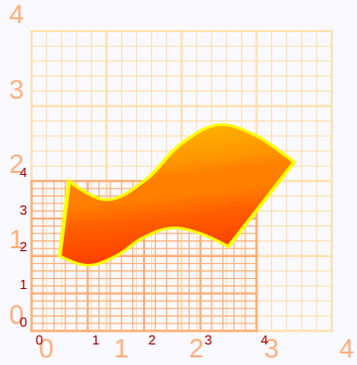
Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

http://www.tug.org.in

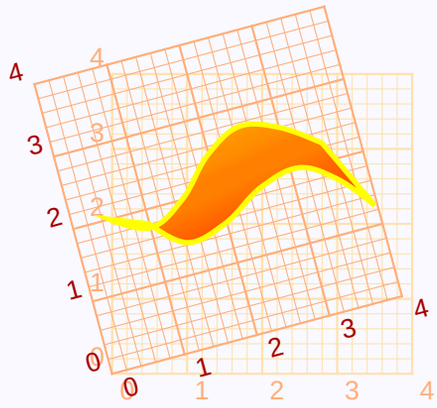
```
\begin{pspicture}(0,-1)(4,4.5)
  \pscustom[style=mystyle]{%
    \pscurve(0.5,2)(1,1.75)(1.5,2)(2,2.5)%
            (2.5,2.75)(3,2.6)(3.5,2.25)
    \scale{0.75 0.5}
    \pscurve[liftpen=1]%
      (3.5,2.25)(3,2.6) (2.5,2.75)%
      (2,2.5)(1.5,2)(1,1.75)(0.5,2)
    \closepath}
\end{pspicture}
```



If both axes are to be scaled by the same factor, we need only specify this number only once within the `\scale` command.

The command `\rotate` (surprise!) rotates the axes by the angle specified in degrees, as in the example below:

```
\begin{pspicture}(-2,-1)(4,5)
  \pscustom[style=mystyle]{%
    \pscurve(0.5,2)(1,1.75)(1.5,2)(2,2.5)%
            (2.5,2.75)(3,2.6)(3.5,2.25)
    \rotate{15}
    \pscurve[liftpen=1]%
            (3.5,2.25)(3,2.6) (2.5,2.75)%
            (2,2.5)(1.5,2)(1,1.75)(0.5,2)
    \closepath}
\end{pspicture}
```



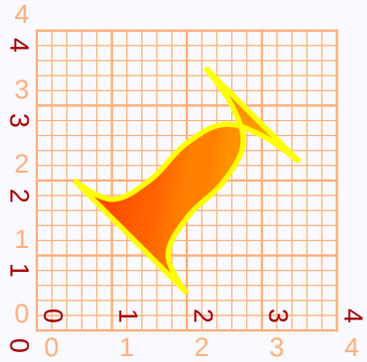
The coordinate axes can be interchanged with the `\swapaxes` command.

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

http://www.tug.org.in

```
\begin{pspicture}(0,-1)(4,4.5)
  \palecolgrid(0,0)(4,4)
  \rput{-90}{\colgrid[xunit=-1cm,yunit=1cm](0,0)(4,4)}
  \pscustom[style=mystyle]{%
    \pscurve(0.5,2)(1,1.75)(1.5,2)(2,2.5)%
            (2.5,2.75)(3,2.6)(3.5,2.25)
    \swapaxes
    \pscurve[liftpen=1]%
            (3.5,2.25)(3,2.6) (2.5,2.75)%
            (2,2.5)(1.5,2)(1,1.75)(0.5,2)
    \closepath}
\end{pspicture}
```



Custom Graphics

Simple customization

Higher level customization

Online LaTeX Tutorial

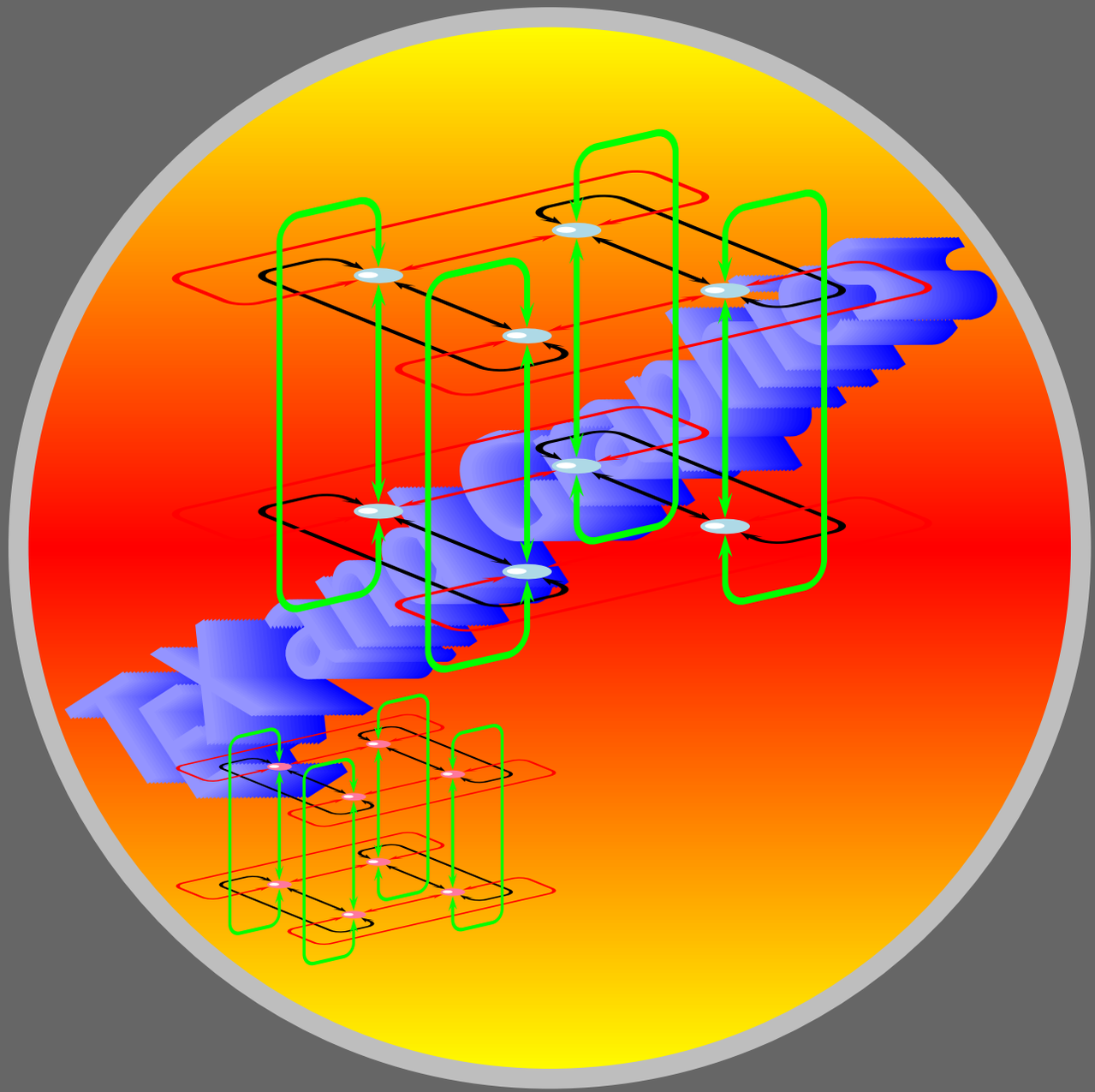
Part II – Graphics

PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA

http://www.tug.org.in



Custom Graphics

Boxing Text
Clipping
Rotating and scaling
Text along a path
Text as graphic

Online LaTeX Tutorial

Part II – Graphics

PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to
tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

9. Tricks with Text

In our discussions so far, we've been focusing on *graphic objects* and we've treated *text* only incidentally in Chapter 6, as labels in pictures. We now see how text can be manipulated in various ways using PSTricks.

Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to
tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

9.1. Boxing Text

LaTeX has various macros for putting text in boxes (or putting boxes around text) and PSTricks defines its own boxing macros. The advantage of using these is the ease of adorning these boxes using colors, shadows and so on. The simplest of such commands is the `\psframebox` as in the example below:

```
\psframebox[fillstyle=solid,%
            fillcolor=Cyan,%
            linecolor=RoyalBlue]%
            {\color{Red}
             \LARGE\bfseries
             Text In A Box}
```

Text In A Box

The distance between the sides of the box and the enclosed text is controlled by the `framesep` parameter. By default, its value is 3 point, but as with other parameters, can be set to any desired value, as shown in the next example:

```
\psframebox[framesep=10pt,%
            fillstyle=solid,%
            fillcolor=Cyan,%
            linecolor=RoyalBlue]%
            {\color{Red}
             \LARGE\bfseries
             Text In A Box}
```

Text In A Box

A variant of the `\psframebox` is the `\psdblframebox` which, as the name indicates, doubles each line of the frame

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\psdblframebox[framesep=10pt,%
               fillstyle=solid,%
               fillcolor=Cyan,%
               linecolor=RoyalBlue,%
               doublecolor=Apricot,%
               doublesep=3pt]%
              {\color{Red}
               \LARGE\bfseries
               Text In A Box}
```

Recall that the `doublesep` parameter determines the width of the space between the double lines and the `doublesep` the color of this space, as mentioned in Chapter 3. The default value of `doublesep` for the `\psdblframebox` is `\pslinewidth` and the default value of `doublecolor` is white.

Another variant is the `\psshadowbox` which, obviously enough, draws a (single) frame with a shadow, as shown below:

```
\psshadowbox[framesep=10pt,%
             fillstyle=solid,%
             fillcolor=Cyan,%
             linecolor=RoyalBlue,%
             shadowcolor=Blue,%
             shadowsize=5pt]%
            {\color{Red}
             \LARGE\bfseries
             Text In A Shadow Box}
```

Note that the parameters `shadowsize` and `shadowcolor` are discussed in Chapter 3.

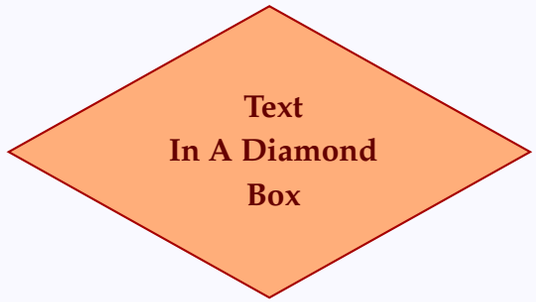
If you are tired of plain old rectangular boxes, you can try `\psdiabox`, which draws a diamond shaped box:

Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

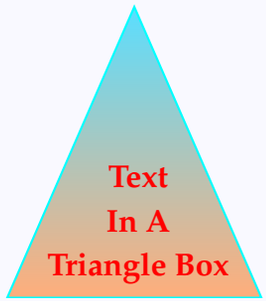
The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\psdiabox[fillstyle=solid,%
          fillcolor=Apricot,%
          linecolor=Mahogany]%
         {\color{Brown}
          \large\bfseries
          \renewcommand{%
            \arraystretch}{1.2}
          \begin{tabular}{c}
            Text\\
            In A Diamond\\
            Box
          \end{tabular}}
```



or \pstribox, which draws a triangular box:

```
\pstribox[fillstyle=gradient,%
          gradbegin=CornflowerBlue,%
          gradend=Apricot,%
          gradmidpoint=1,%
          linecolor=Cyan]%
         {\color{Red}
          \large\bfseries
          \renewcommand{%
            \arraystretch}{1.2}
          \begin{tabular}{%
              @{\hspace{-15pt}}c@{\hspace{-15pt}}}
                Text\\
                In A\\
                Triangle Box
          \end{tabular}}
```



Recall that the gradient style of filling requires the pst-grad package, as explained in Chapter 2.

For those who are inclined towards curves than angles, there's a \pscirclebox:

Online LaTeX Tutorial

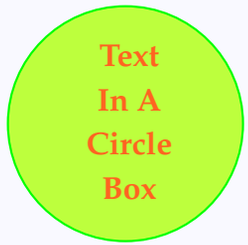
Part II – Graphics

PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\pscirclebox[fillstyle=solid,%
             fillcolor=SpringGreen,%
             linecolor=Green]%
            {\color{Orange}
             \large\bfseries
             \renewcommand{%
               \arraystretch}{1.2}
             \begin{tabular}{c}
               Text\\
               In A\\
               Circle\\
               Box
             \end{tabular}}
```



or even a `\psovalbox`:

```
\psovalbox[fillstyle=solid,%
           fillcolor=Orange,%
           linecolor=BrickRed]%
          {\color{SpringGreen}
           \large\bfseries
           \renewcommand{%
             \arraystretch}{1.2}
           \begin{tabular}{c}
                  Text\\
                  In An\\
                  Oval\\
                  Box
           \end{tabular}}
```

Another parameter for the various boxes is the `boxsep` whose default value is `true`. In this case, the box that is produced (in the TEXnical sense) is the size of the "frame" around it. If it is set to `false`, then the box produced is the size of what's inside, so that the frame is transparent to TEX. This is apparent only when the boxes are used within some surrounding text as illustrated below:

Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\color{Blue}
Thus we find that $x+y=3$ and
using this together with
\psovalbox[linecolor=Red]%
          {$x^2+y^2=3$}
found earlier, we see that
$x=2$ and $y=1$

\vspace{1cm}

Thus we find that $x+y=3$ and
using this together with
\psovalbox[linecolor=Red,%
           boxsep=false]%
          {$x^2+y^2=3$}
found earlier, we see that
$x=2$ and $y=1$
```

Thus we find that $x + y = 3$ and using this together with $x^2 + y^2 = 3$ found earlier, we see that $x = 2$ and $y = 1$

Thus we find that $x + y = 3$ and using this together with $x^2 + y^2 = 3$ found earlier, we see that $x = 2$ and $y = 1$

Each of the boxing commands above has a *starred* version, which draws a *solid* shape around the enclosed text instead of just a frame. This is similar to the starred versions of graphic objects we've seen earlier, but the color of the boxes is determined by `fillcolor` instead of `linecolor` for other graphic objects.

Online LaTeX Tutorial

Part II – Graphics

PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\SaveVerb{box}=\psframebox=
\psframebox[fillstyle=solid,%
            fillcolor=Cyan]%
            {\color{Red}
             \LARGE\bfseries
             Text In A \UseVerb{box}}

\vspace{1cm}

\SaveVerb{starbox}=\psframebox*=
\psframebox*[fillstyle=solid,%
             fillcolor=Cyan]%
            {\color{Red}
             \LARGE\bfseries
             Text In A \UseVerb{starbox}}
```

Text In A \psframebox

Text In A \psframebox*

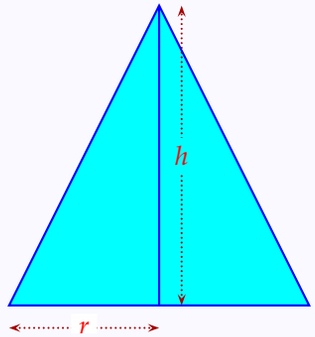
(Here, the command pair \SaveVerb and \UseVerb come from the package fancyvrb and are used to get the control sequence strings \psframebox and \psframebox* as arguments of the commands.)

We've mentioned in Chapter 6 that the \rput* command puts the text first in a \psframebox*. But there are occasions when we've to use both \rput and \psframebox together instead of a single \rput*, as in the example below:

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\psset{linecolor=Blue}
\begin{pspicture}(0,0)(4,5)
  \pspolygon[fillstyle=solid,%
             fillcolor=Cyan]%
            (0,0)(4,0)(2,4)
  \psline(2,0)(2,4)
  \psset{linecolor=Mahogany,%
         linestyle=dotted,%
         dotsep=1pt,%
         arrows=<->}
  \psline(2.3,0)(2.3,4)
  \rput(2.3,2){%
    \psframebox*[boxsep=false,%
                 fillcolor=Cyan]%
                {\color{Red} $h$}}
  \psline(0,-0.3)(2,-0.3)
  \rput*(1,-0.3){\color{Red} $r$}
\end{pspicture}
```



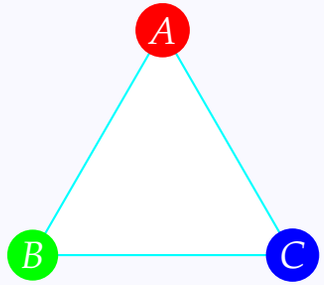
Note that here, we cannot use `\rput*` directly for the label *h*, since the default color of `\psframebox*` is white, (which is OK for the label *r*) but we want the color of the box for *h* to be cyan, to blend it with its background.

While on the topic of "putting", we should also mention the command `\cput` (and of course `\cput*`) which combines the functions of `\pscirclebox` and `\rput` (or `\rput*`), as shown in the next example:

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\Large
\psset{fillstyle=solid,unit=2cm}
\begin{pspicture}(0,-1)(1,1)
  \SpecialCoor
  \pspolygon[linecolor=Cyan]%
            (0,1)(1;210)(1;330)
  \cput*[fillcolor=Red]%
        (0,1)
        {\color{White} $A$}
  \cput*[fillcolor=Green]%
        (1;210)
        {\color{White} $B$}
  \cput*[fillcolor=Blue]%
        (1;330)
        {\color{White} $C$}
\end{pspicture}
```



Online LaTeX Tutorial

Part II – Graphics

PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

9.2. Clipping

We can *clip* text, that is, cut off everything outside a specified boundary, using the `\psclip....\endpsclip` commands. A simple example is given below:

```
\psclip{%
  \psdiamond[linecolor=Red,%
             fillstyle=solid,%
             fillcolor=Yellow]%
            (2,0.25)(2.5,0.5)}
  \color{Blue}\Huge Cut Diamond
\endpsclip
```

By careful use of coordinates, we can create an overlay effect with suitable clipping, as in the next example:

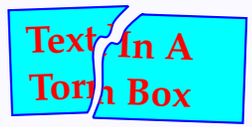
```
\begin{pspicture}(0,0)(6,2)
  \rput[bl](0.5,1){%
    \color{Blue}\Huge Cut Diamond}
  \psclip{\psdiamond*[linecolor=Yellow]%
                     (3,1.25)(2.25,0.5)}
    \rput[bl](0.5,1){%
      \color{OliveGreen}\Huge Cut Diamond}
  \endpsclip
\end{pspicture}
```

Or special effects like this:

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(0,0)(2.5,2)
  \rput{2}{%
    \psclip{\pscustom[style=bluestyle]{%
              \psline(1.6,1.4)(0,1.4)(0,0)(1,0)
              \tearcurve}}
      \rput[bl](0,0){\color{Red}\Large\bfseries
        \begin{tabular}{c}
          Text In A\\
          Torn Box
        \end{tabular}}
    \endpsclip}
\end{pspicture}
\hspace{-2.6cm}
\begin{pspicture}(0,0)(2.5,2)
  \rput{-2}{%
    \psclip{\pscustom[style=bluestyle]{%
              \tearcurve
              \psline(1.6,1.4)(3,1.4)(3,0)(1,0)}}
      \rput[bl](0,0){\color{Red}\Large\bfseries
        \begin{tabular}{c}
          Text In A\\
          Torn Box
        \end{tabular}}
    \endpsclip}
\end{pspicture}
```



Here, the custom style `bluestyle` and the custom curve `\tearcurve` are defined as follows:

```
\newpsstyle{bluestyle}{%
  linecolor=Blue,%
  fillstyle=solid,%
  fillcolor=Cyan}
\newcommand{\tearcurve}{%
  \pscurve(1,0)(1.1,0.3)(1.2,1)(1.3,1)%
          (1.4,1.2)(1.6,1.6)}
```

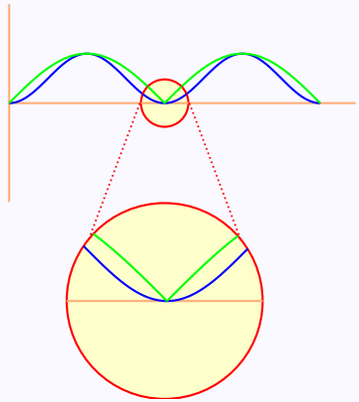

Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

Incidentally,note that the `\psclip...\endpsclip` commands can be used to clip not only text, but graphic objects also, as shown in the example below:

```
\psset{unit=0.66,linecolor=Red}
\begin{pspicture}(0,-7)(6.5,2)
  \coloraxes(0,0)(0,-2)(7,2)
  \plotsqsin
  \plotabssin
  \psclip{\ycirc(3.1416,0){0.5}}
  \coloraxes(0,0)(0,-2)(7,2)
  \plotsqsin
  \plotabssin
  \endpsclip
  \dotline(2.6416,0)(1.1416,-4)
  \dotline(3.6416,0)(5.1416,-4)
  \psclip{\ycirc(3.1416,-4){2}}
  \psset{origin={3.1416,4},unit=1.33cm}
  \coloraxes(0,0)(0,-2)(7,2)
  \plotsqsin
  \plotabssin
  \endpsclip
\end{pspicture}
```



where the various customized commands used are as follows:

```
\newcommand{\plotsqsin}{%
  \psplot[plotpoints=500,plotstyle=curve,linecolor=Blue]%
         {0}{6.2832}{57.2958 x mul sin 2 exp}}
\newcommand{\plotabssin}{%
   \psplot[plotpoints=500,plotstyle=curve,linecolor=Green]%
          {0}{6.2832}{57.2958 x mul sin abs}}
\definecolor{PaleYellow}{cmyk}{0,0,0.2,0}
\newpsobject{ycirc}{pscircle}{fillstyle=solid,fillcolor=PaleYellow}
\newpsobject{dotline}{psline}{linestyle=dotted,dotsep=1pt}
\newpsobject{coloraxes}{psaxes}%
            {linestyle=solid,linecolor=Apricot,labels=none,ticks=none}
```

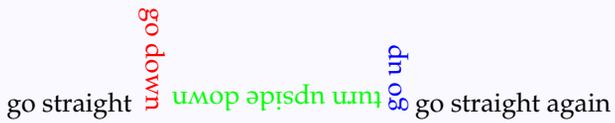

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

9.3. Rotating and scaling

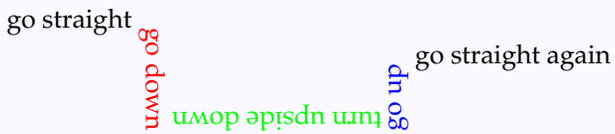
There are are also ready to use commands for rotating text left, right or down, leaving the needed amount of spaces.

```
go straight
\rotateright{\color{Red} go down}
\rotatedown{\color{Green} turn upside down}
\rotateleft{\color{Blue} go up}
go straight again
```



A better effect can be obtained by computing the heights of various upright boxes and raising and lowering them by the appropriate lengths:

```
\newlength{\dlen}
\settoheight{\dlen}{%
  \rotateright{\color{Red} go down}}
\newlength{\ulen}
\settoheight{\ulen}{%
  \rotateleft{\color{Blue} go up}}
go straight
\raisebox{-\dlen}{%
  \rotateright{\color{Red} go down}
\rotatedown{%
  \color{Green} turn upside down}
\rotateleft{\color{Blue} go up}}
\raisebox{\ulen-\dlen}{%
   go straight again}
```



For such manipulation of long pieces of text, these commands also have the "environmental" forms `\begin{Rotateleft}...\end{Rotateleft}` and others.

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\textbf{\color{Red} Question}:
\color{Blue} Why did the tachyon
cross the street?\\[10pt]
\begin{Rotatedown}
  \parbox{\linewidth}{%
    \textbf{\coor{Red} Answer}:
    \color{Blue} Because it's already
    on the other side}
\end{Rotatedown}
```

Question: Why did the tachyon cross the street?

Answer: Because it's already on the other side

Text can also be *scaled*, using the command `\scalebox`. The general form of this command is

`\scalebox{`*number1 number2*`}{`*text*`}`

where *number1* is the horizontal scaling and *number2* is the vertical scaling. If only one number is specified, it is used for scaling in both directions. This is illustrated in the examples below:

```
\scalebox{0.8 4}{%
  \color{Red}tall and lean}

\bigskip

\scalebox{4 0.8}{%
  \color{Green}short and fat}

\bigskip

\scalebox{2}{%
  \color{Blue}large but proportional}
```

tall and lean

short and fat

large but proportional

Using negative numbers for scaling, we can flip text around either axis:

Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
{\Large\bfseries\color{Red} MIRROR}
 {\color{Blue}\rule[-0.3cm]{0.1cm}{1cm}}
 {\Large\bfseries\color{Red}
   \scalebox{-1 1}{MIRROR}}

\vspace{1cm}

{\Large\bfseries\color{Red} MIRROR}\\
{\color{Blue}\rule{2.5cm}{0.1cm}}\\
{\Large\bfseries\color{Red}
  \scalebox{1 -1}{MIRROR}}
```

MIRROR MIRROR

MIRROR

MIRROR

We also have the `\scaleboxto` command with the general form

`\scaleboxto(`*number1*`,`*number2*`){`*text*`}`

With this command *text* is scaled to have width *number1* units and height plus depth equal to *number2* units. If one of the numbers is set to 0, then the box s scaled to have width and height (plus depth) equal to the other number. (Of course, we cannot set *both* numbers equal to 0).

```
\scaleboxto(1.5,1){\color{Red}
   tall and lean}

\bigskip

\scaleboxto(7,0.2){\color{Green}
   short and fat}

\bigskip

\scaleboxto(3,0){\color{Blue}
   small but proportional}
```

tall and lean

short and fat

small but proportional

Online LaTeX Tutorial
Part II – Graphics
PSTricks

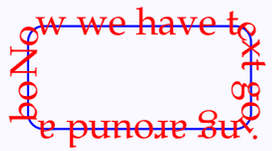
E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

9.4. Text along a path

One of the interesting features of the PostScript language is that it treats text as graphical object. This allows various manipulations of text. The package pst-text provides the command `\pstextpath` to set text along a specified path. Look at this example:

```
\begin{pspicture}(0,0)(3,2)
  \pstextpath{%
    \psframe[framearc=0.3,linecolor=Blue](0,0)(3,1.4)}{%
    \color{Red}\Large  Now we have text going around a box}
\end{pspicture}
```



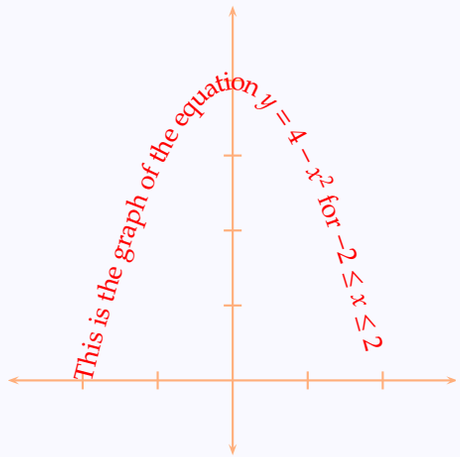
Note that the general form of the command `\pstextpath` is

`\pstextpath{`*graphic*`}{`*text*`}`

where, *graphic* specifies the path along which the specified *text* is to be set.

By default, `\pstextpath` draws also the graphic specified, but this can be suppressed by setting `linestyle=none`, as shown below:

```
\begin{pspicture}(-3,-1)(3,5)
  \colaxes[labels=none](0,0)(-3,-1)(3,5)
  \pstextpath{%
    \psplot[linestyle=none]{-2}{2}{4 x 2 exp sub}}{%
      \color{Red} This is the graph of the
                  equation $y=4-x^2$ for
                  $-2\le x\le 2$}
\end{pspicture}
```

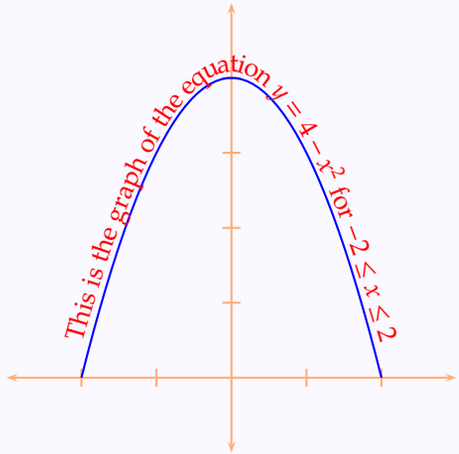


E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

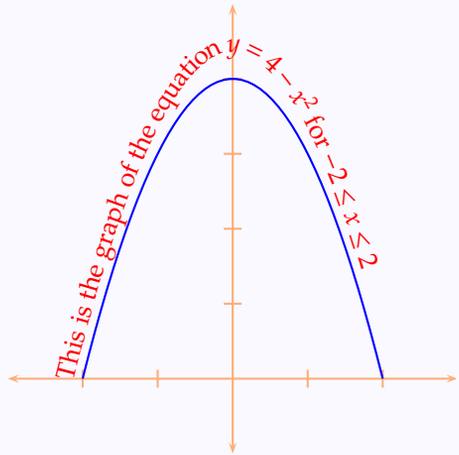
(Note that \colaxes used in he above example is a custom command, discussed in Chapter 7).

What if we need something like this?



The trick is to first draw the curve and then use \pstextpath to set the text along a slightly scaled up version of the curve, without actually drawing the second curve:

```
\begin{pspicture}(-3,-1)(3,5)
  \colaxes[labels=none](0,0)(-3,-1)(3,5)
  \psset{linecolor=Blue}
  \psplot{-2}{2}{4 x 2 exp sub}
  \psset{linestyle=none,unit=1.12cm}
  \pstextpath{%
    \psplot{-2}{2}{4 x 2 exp sub}}{%
    \color{Red} This is the graph of the
                equation $y=4-x^2$ for
                $-2\le x\le 2$}
\end{pspicture}
```

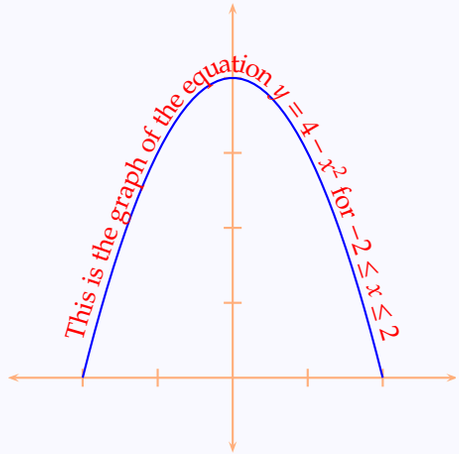


E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

But this is not exactly what we want. The trouble is that the command `\pstextpath`, by default, places the beginning of the text at the beginning of the path; however, it has an optional parameter which can be used to shift the position of the text:

```
\begin{pspicture}(-3,-1)(3,5)
  \colaxes[labels=none](0,0)(-3,-1)(3,5)
  \psset{linecolor=Blue}
  \psplot{-2}{2}{4 x 2 exp sub}
  \psset{linestyle=none,xunit=1.13cm,yunit=1.05cm}
  \pstextpath[c]{%
    \psplot{-2}{2}{4 x 2 exp sub}}{%
    \color{Red} This is the graph of the
                equation $y=4-x^2$ for
                $-2\le x\le 2$}
\end{pspicture}
```

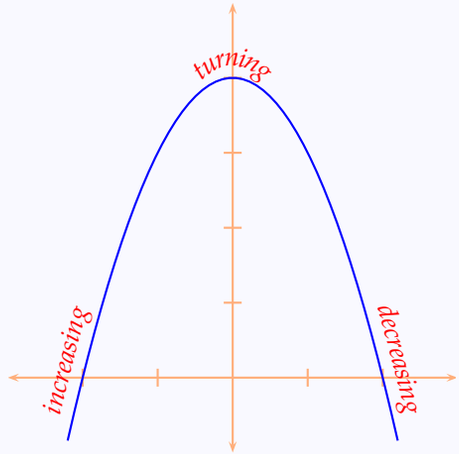


Note how we used the optional value `c` to center the text relative to the curve. (Note also the slight difference in `xunit` and `yunit` to get the text at the top just right). Other optional values are `l` (the default) for left justification and `r` for right justification. These are illustrated in the next example:

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to `tutorialteam@tug.org.in`

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
`http://www.tug.org.in`

```
\begin{pspicture}(-3,-1)(3,5)
  \colaxes[labels=none](0,0)(-3,-1)(3,5)
  \psset{linecolor=Blue}
  \psplot{-2.2}{2.2}{4 x 2 exp sub}
  \psset{unit=1.15cm,linestyle=none}
  \pstextpath[l]{%
    \psplot{-2.1}{2.1}{4 x 2 exp sub}}{%
    \color{Red}\textit{increasing}}
  \pstextpath[r]{%
    \psplot{-2.1}{2.1}{4 x 2 exp sub}}{%
    \color{Red}\textit{decreasing}}
  \psset{unit=1.07cm}
  \pstextpath[c]{%
    \psplot{-2.1}{2.1}{4 x 2 exp sub}}{%
    \color{Red}\textit{turning}}
\end{pspicture}
```



Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

9.5. Text as graphic

The package pst-char provides the command \pscharpath which can be used to embellish text with colors and the like just as it were a graphic object. We give a couple of examples to illustrate this:

```
\begin{pspicture}(0,-1)(8,2)
  \DeclareFixedFont{\bigrm}{T1}{ptm}{m}{n}{1.5cm}
  \pscharpath[fillstyle=solid,%
              fillcolor=SkyBlue,%
              linecolor=Red]%
             {\bigrm PSTricks}
\end{pspicture}
```

PSTricks

Here, the command \DeclareFixedFont is the LaTeX way of specifying the font to be used.

```
\begin{pspicture}(0,-1)(8,2)
  \DeclareFixedFont{\bigsf}{T1}{phv}{b}{n}{1.5cm}
  \pscharpath[linecolor=Yellow,%
              fillstyle=gradient,%
              gradbegin=Yellow,%
              gradend=Red,%
              gradmidpoint=1,%
              gradangle=5]%
             {\bigsf PSTricks}
\end{pspicture}
```

PSTricks

This package also contains the command air \pscharclip...\endpscharclip, which like \psclip...\endpsclip pair, clips any object within them,but this time to the shape of the specified text:

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\begin{pspicture}(-1.5,-1)(8,2)
  \DeclareFixedFont{\bigsf}{T1}{phv}{b}{n}{1.5cm}
  \DeclareFixedFont{\smallrm}{T1}{ptm}{m}{n}{2mm}
  \pscharclip[linestyle=none,%
              fillstyle=solid,%
              fillcolor=Cyan]%
             {\rput[bl](0,0){%
                 \bigsf PSTricks}}
     \rput[t]{90}(0,0){%
       \begin{minipage}{8cm}
         \offinterlineskip
         \newcounter{pscount}
         \setcounter{pscount}{500}
         \whiledo{\value{pscount}>0}{%
           \addtocounter{pscount}{-1}
           \color{Blue}\smallrm  Postcript}
       \end{minipage}}
  \endpscharclip
\end{pspicture}
```

(Here, the text to be clipped is “PostScript” written 500 times, in small font, specified by `\smallrm`, which is generated by the code starting with `\newcounter`, put in a box 8 centimeters wide and turned through ninety degrees.).

As in the case of `\psclip` this can also be used to produce an overlay effect.

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

```
\DeclareFixedFont{\bigsf}{T1}{phv}{b}{n}{1.75cm}
\begin{pspicture}(0,-0.5)(8,3)
  \rput[bl](0,0){%
    \begin{minipage}{8cm}
      \color{RoyalBlue}
      \firstpara
    \end{minipage}}
  \pscharclip[linestyle=none,%
              fillstyle=solid,%
              fillcolor=CornflowerBlue]%
             {\rput[bl](0.25,0.15){%
                 \bigsf PSTricks}}
      \rput[bl](0,0){%
        \begin{minipage}{8cm}
          \color{Blue}
          \firstpara
        \end{minipage}}
  \endpscharclip
\end{pspicture}
```

PSTricks

LaTeX has only limited drawing capabilities, while PostScript is a page description language which has a rich set of drawing commands; and there are programs (such as dvips) which translate the dvi output to PostScript. So, the natural question is whether one can include PostScript code in a TeX source file itself for programs such as dvips to process after the TeX compilation? This is the idea behind the PSTricks package of Timothy van Zandt. The beauty of it is one need not know PostScript to use it—the necessary PostScript code can be generated by TeX macros defined in the package

Here, the command `\firstpara` is defined by

```
\newcommand{\firstpara}{%
  \scriptsize
  \LaTeX\ has only limited drawing capabilities, while
  PostScript is a page description language which has a rich set of
  drawing commands; and there are programs (such as \textsf{dvips})
  which translate the \texttt{dvi} output to PostScript. So, the
  natural question is whether one can include PostScript code in a
  \TeX\ source file itself for programs such as \textsf{dvips} to
  process after the \TeX\ compilation? This is the idea behind the
  \textsf{PSTricks} package of Timothy van Zandt. The beauty of it is
  one need not know PostScript to use it---the necessary PostScript
  code can be generated by \TeX\ macros defined in the package}
```

Custom Graphics

Online LaTeX Tutorial

Part II – Graphics

PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to tutorialteam@tug.org.in

The Indian TeX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in

just typesets the opening paragraph of our tutorial in `\tiny` font in an 8 centimeter wide box.

Online LaTeX Tutorial
Part II – Graphics
PSTricks

E Krishnan, CV Radhakrishnan and AJ Alex constitute the graphics tutorial team. Comments and suggestions may be mailed to
tutorialteam@tug.org.in

The Indian TEX Users Group
Floor III, SJP Buildings, Cotton Hills
Trivandrum 695014, INDIA
http://www.tug.org.in