The psgrapha Environment

Michael Sharpe

msharpe at ucsd.edu

1 Quick overview

This package contains a number of utility macros along with an environment that is similar to the `psgraph` environment of `pst-plot`, but with several substantial differences. The final 'a' may be thought of as standing for 'alternative'. It is intended to be used to create a single graphic that is to be included in a final LaTeX document. Recall that

```
\psset{llx=<value>,lly=<value>.urx=<value>,ury=<value>}
\begin{psgraph}[<axes settings>](x1,y1)(x2,y2){<width>}{<height>}
<plot commands>
\end{psgraph}
```

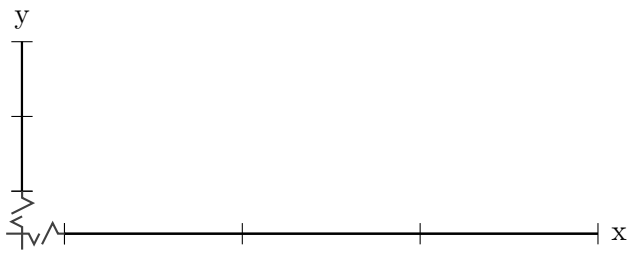
does the following:

- Computes values of `xunit` and `yunit` so that `(x2-x1)*xunit=width` and `(y2-y1)*yunit=height`.
- Adds borders determined by the parameters `llx`, `lly`, `urx`, `ury`, changing the overall size of the picture to allow space for labels. It constructs a `pspicture` of width `width+urx-llx` and height `height+ury-lly`.
- Constructs axes from `x1` to `x2` and `y1` to `y2` respectively, with tick marks, arrows and labels as specified by the parameters.
- Allows then the further insertion of any picture objects such as plots, lines, and so on, allowing access to all the plotting options from the `pst-plot` and `pstricks-add` packages.
- If an additional coordinate pair `(x0,y0)` is prepended to the argument list, that is used as the point where the axes cross—the origin, in PSTricks parlance. If not, `(x0,y0)` is set equal to `(x1,y1)`.

By contrast, the `psgrapha` package handles a few items in a different way.

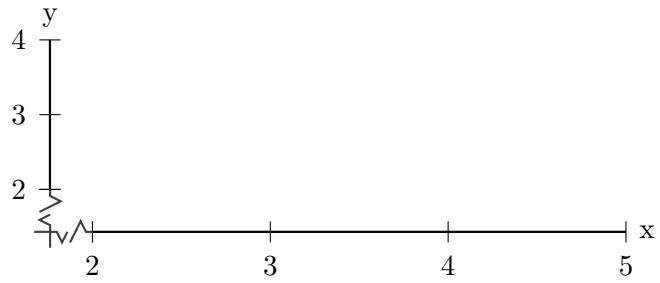
- There is no option to specify the point where the axes cross. Instead, one sets the keyword `AxisShift` (default value 16pt) whose effect is that if the true origin (0,0) is not within the coordinate canvas, then the axes are broken by that amount to indicate a break in coordinates. A broken zigzag effect is used to indicate this. Allowance is always made for the additional space this may require. The `\xAxis` keyword `(true | false)` determines whether the x axis is rendered. If `true`, it is rendered with basic ticks but no numeric labels if the keyword `xTicks` is `true`. Both `xAxis` and `xTicks` are `true` by default, and similarly for `yAxis, yTicks`. Here is an example resulting from

  ```
  \begin{psgrapha}(2,2)(5,4){3in}{1in}\end{psgrapha}
  ```

  The next example mimics `psgraph`, except for the broken axes.

  ```
  \def\xNumCount{-1} \def\yNumCount{-1}
  \begin{psgrapha}(2,2)(5,4){3in}{1in}\end{psgrapha}
  ```

- The background grid is more flexible, with different spacings allowed in the x and y directions.

- The given width and height are the size of the finished picture—any assigned values of `llx`, `lly`, `urx`, `ury` subtract from rather than add to the picture size in computing the graph size. Note that, unlike `psgraph`, these keywords may be set in either a `\psset{}` line prior to

`\begin{psgrapha}` or as keyword arguments to `\begin{psgrapha}`. The same goes for the keyword `equalunits` which forces the units in the x and y directions to be the same, at the cost of leaving more white space above or to the right of the picture.

- The axis numeric labels are not computed automatically from the given data. It is expected that you provide an explicit description of all ticks and numeric labels—see below for details.
- There is provision for trigonometric units on one or both of the axes, though handled differently from current usage in `pstricks-add`.
- The `psgrapha` environment allows the use to enter data in a more elementary manner than in most PSTricks packages, taking advantage of the `arrayjobx` package (an update of the `arrayjob` package which no longer conflicts with the `array`, `cases` and `tabular` environments of `amsmath`) for data entry. With this, the degree of PSTricks expertise required is lowered, and the user can simply enter lists of data that will show up in the picture. See below for the details. Data entered in arrays reduces the risk of inadvertently adding unwanted spaces within a `\pspicture` environment — one of the most common causes of difficulties in using tt pstricks.
- Arrows in `psgrapha` must be entered as keywords, not as an argument. That is, you must write `\begin{psgrapha}[arrows=->]...` rather than `\begin{psgrapha}{->}...`.

2 Typical layout of a psgrapha file

```
<usual LaTeX preamble>
\input{PersonalDefinitions}---can be any file
\begin{document}
<psgrapha definitions particular to this document>
\begin{grapha}...
<further pstricks code (optional)>
\end{psgrapha}
\end{document}
```

The plan is that you will keep many of your definitions in an input file, entering only the definitions that are to be changed for this particular graphic.

Options added as keywords to `\begin{psgrapha}` should be limited to axes specifications.

3 Keywords

3.1 New keywords specific to psgrapha

- `tickdim` (default value `4pt`) Sets the length of major ticks used on both axes.
- `tickpref` (default value `<empty>`) If left `<empty>`, the keyword is ignored. The other possible settings are `A`, meaning ticks should point in toward the main quadrant, `C`, meaning that ticks point outward from the main quadrant, and `B`, pointing in Both directions by `tickdim`.
- `axesarrows` (default value `-`) Specifies arrow settings for axes only.
- `GridDotRadius` (default value `.3pt`. Radius of dots used in grids.
- `GridColor` (default value `gray`) Color to use for grid points. Color expressions such as `black!60!yellow` are acceptable.
- `AxisColor` (default value `black`) Color to use for drawing axes only.
- `BrokenAxisColor` (default value `black!75`) Color to use for drawing broken axes.
- `AxisThickness` (default value `\pslinewidth` Line width to use for axes only.
- `AxisShift` (default value `16pt`) Amount by which axes should be shifted to show broken axis sign.
- `xLabelSpace` (default value `12pt`) Normally, distance below x axis to set baseline for labels.
- `yLabelSpace` (default value `5pt`) Normally, distance from y axis to right edge of labels on y axis.
- `xLowerLabelSpace` (default value `22pt`) Normally, distance below x axis for secondary labels.
- `GridMajorLinewidth` (default value `.5pt`) Line width for major grid lines.

- `GridMinorLinewidth` (default value `.3pt`) Line width for minor grid lines.
- `GridMajorLinecolor` (default value `blue!35`) Color for major grid lines.
- `GridMinorLinecolor` (default value `gray!35`) Color for minor grid lines.
- `trigx` (no default value). Boolean keyword. If `true`, labels on x axis are multiplied by π, though the underlying coordinates are not modified. This requires functions to be specified with the argument x replaced by $\pi * x$. There is also a corresponding `trigy`. See example.
- `equalunits` (default value `false`.) If true, xunit is forced to be equal to yunit, leading to white space either above or to the right of the picture.
- `xTicks` (default value `true`.) If true, ticks are drawn on the x axis. There is also a boolean keyword `yTicks`.
- `fullcanvas` (default value `false`.) If true, an invisible frame is drawn around the canvas so no part of the canvas will be removed when trimming with an external program such as `ps2eps`.
- `showGrid` (default value `false`.) If true (and `graphPaper=false`), a grid is drawn with position based on the values in the macros `\xGridMin`, `\xGridMax`, `\yGridMin`, `\yGridMax`, `\xGrid`, `\yGrid` (separations between major lines), `\xGridDiv`, `\yGridDiv` (numbers of divisions between major lines.)
- `graphPaper` (default value `false`.) If true, draws graph-paper-like grid, same layout as with `showGrid`.

3.2 Keywords from psgraph no longer necessary

The following keywords are not needed by `psgrapha`.

- `xAxisLabel, yAxisLabel` will be used in the `psgraph` manner if defined, and should otherwise be set to empty in the preamble:

  ```
  \psset{xAxisLabel=,yAxisLabel=}
  ```

The commands `\xAxisLabel`, `\yAxisLabel`, `\yAxisLabelPosition` take their places.

- `ticksize` is constructed internally by `psgrapha` from `tickdim` and `tickpref` before it draws its axes. If you draw supplementary axes using the normal `pst-plot` command `\psaxes`, it will expect a setting for `ticksize`.
- `trigLabels` should not be defined. Use `trigx`.
- `trigLabelBase` should not be defined. Use `trigx`.

3.3 Keywords from psgraph used differently

- `llx, lly, urx, ury` subtract from the picture rather than adding to it, and the computation of `xunit` and `yunit` is performed after the subtraction, not before.

4 Macros

The principal difference between `psgraph` and `psgrapha` lies in the way data may be entered. In the latter, you define macros that determine the size and variable ranges, and you may as an option to the usual command style of `pstricks`, set arrays of data describing points, lines, curves, numeric labels, axis labels and other labels using array notation. Here is a brief (incomplete) example:

```
% Width and height of the output (supply usual TeX units)
\def\picwidth{2in}
\def\picheight{2in}
% x- and y-ranges
\def\xmin{-10}
\def\xmax{120}
\def\ymin{-5}
\def\ymax{24}
%Numeric labels on x axis
\def\xNumCount{3}
\xNum(1)={25}
\xDelta(1)={1.5}% move numeric label 1 1.5 units to right
```

```
\xNum(3)={75}
\xNumLoc(3)={74}% place numeric label 3 at 74, not 75
\xNum(4)={100}
```

The first six lines instruct `psgrapha` about the overall size of the graphic, and value ranges for the part of the the graphic that is not within the border areas. The code block

```
\def\picwidth{3in}\def\picheight{2in}
\def\xmin{-10}\def\xmax{120}
\def\ymin{-5}\def\ymax{24}
\begin{psgrapha}
```

is equivalent to

```
\begin{psgrapha}(-10,-5)(120,24){3in}{2in}
```

both in pictorial effect, and in that the second form results in the definitions of `\picwidth`, `\xmin` etc.

The `\def\xNumCount{3}` line means: use `\xNum()` data with indices from 1 to 3: any indices greater than 3 are ignored, and any unspecified index (2 in this example) is ignored. The effect is that numeric labels are placed along the x axis at 25+1.5 and 74, using `labelFontSize`, and with baseline as specified by `xLabelSpace`. The numeric labels are set in math mode by default, but can be set as text by setting `mathLabel=false`. There is one special case. If `\xNumCount<0`, all indices are ignored, and the normal `\psaxes` numeric label routines are used — `xLabelSpace` is ignored in this case.

4.1 Dimension issues

The ranges of `x` and `y` values must be kept with reasonable bounds. You will get error messages if any one of `\xmax`, `\xmin`, `\ymax`, `\ymin`, `\xmax-\xmin`, `\ymax-\ymin` exceeds 16383 in absolute value, and TEX will fail with error message `Dimension too large` if the scales are chosen so that the true origin would be more than about 19ft from any point in the picture. This can happen quite easily. For example, if x ranges from 1990 to 2010 in a canvas of width 3in, then `xunit=3in/20=.15in`, and `2010*.15in>300in>19ft`. The only solution is to work with a rescaled `xunit`. For example, setting `\xmin` to 199 and `\xmax` to 201 would satisfy the dimension constraint, and allow for easy numeric conversions. To get the numeric labels correct requires setting

\xNumLoc to the scaled x value instead of the label value. For example, you would write

```
\xNum(1)={1990}
\xNumLoc(1)={199}% place numeric label 1990 at location x=199
```

Naturally, for all specifications of coordinates and curves, you must work with x scaled down to the range [199:201].

Here is a complete listing of all possible categories of data and the arrays associated with them.

4.2 Points

```
\def\PointCount{1}
\PointName(1)={P1}
\PointPos(1)={1,2}
```

Point positions may be specified using any form understood by PSTricks \SpecialCoor—eg, radial coordinates {3;30}. They may even be node expressions—see the last section of these notes. Note that all values, no matter what the type, must be enclosed by braces. Coordinates need not use the usual parentheses. Note that there are a several pre-defined points within the psgrapha environment which serve as starting points for expressions.

4.3 Dots

Dots are controlled by three entries — \DotPos() to specify the location of the center (any form understood by \SpecialCoor), \DotOpts(), where you set the options that specify the node color, size, style (and possibly the linewidth, fillcolor and linecolor), and \DotType(), where you may specify a custom dot type. Eg,

```
\def\DotCount{1}\DotOpts(1)={linecolor=red,dotscale=2}%twice normal size
```

By default, dotstyle=*, which generates a solid dot. For dots with an open center, like dotstyle=o and dotstyle=Bo, the center is filled with fillcolor. The last two allow you to draw small circles. The default diameter is 2pt+2\linewidth, and this can be magnified using the keyword dotscale (eg, dotscale=3 to scale up by a factor of 3), or by spec-

ifying `dotsize`, say with `dotsize=3pt 2` which would make the diameter `3pt+2\linewidth`. The `\DotType()`, if present, may refer to one of variant forms of `\psdot` defined in your preamble. You could, for example, define macros `\psdotA, \psdotB` by

```
\def\psdotA(#1){\psdot[dotstyle=Bo,dotsize=4pt](#1)}
% 4pt diameter open circle filled with white
\def\psdotB(#1){\psdot[dotstyle=Bo,fillcolor=yellow,dotsize=6pt](#1)}
% 6pt diameter open circle filled with yellow
```

Then, setting `\DotType(1)={A}` will cause the first dot to be rendered using the settings in the `\psdotA` macro. This is a useful way to get a number of uniform dot styles in your diagrams.

4.4 Circles

Circles have only three arguments—`\CirclePos()` to specify the location of the center (any form understood by `\SpecialCoor`), `\CircleRadius()` for the radius, and `\CircleOpts`, which lists the options that detail the circle color and linewidth. Eg,

```
\CircleOpts(1)={linecolor=red,linewidth=1.5pt}
```

4.5 Lines

```
\def\LineCount{1}
% Next line specifies options for all lines
\def\everyLine{linecolor=blue,linewidth=.4pt,arrows=-D>}
%uncomment next line to change line options for this index only
%\LineOpts(1)={}
\LineArrow(1)={->}
\LineStart(1)={0.2,5}
\LineEnd(1)={AxesInt}
%\LineDelta(1)={1cm,-2}
```

The points at the start and end of the lines may be Cartesian coordinates, or any other form acceptable to `\SpecialCoor`, including named points. If `\LineArrow` is not defined, it is understood to take the value `-`, giving a line without arrowheads, or whatever might have been assigned in `\everyLine`.

Likewise, if `\LineOpts` is undefined, the keywords currently in force (including those specified in `\everyLine`) are not modified. If `\LineDelta(1)` is uncommented and `\LineEnd(1)` is commented, the line will be drawn instead using with a relative displacement to the endpoint, which may be dimensions or coordinate values. In the example above, the endpoint will be `1cm` to the right and 2 y units below the start of the line.

4.6 Curves

```
% Curves--specify algebraically, not with PS
\CurveCount{1}
\Curvex(1)={c1*t}% constant c1 is defined with PS in \CurveConsts(1)
\Curvey(1)={.02*t^2}
\CurveMin(1)={0} % Initial value of parameter  t  for curve 1
\CurveMax(1)={100}  % Final value of parameter  t  for curve 1
\CurveParams(1)={linecolor=red}% optional---add keywords
\CurveConsts(1)={/eps .7 def /c1 eps dup mul neg 1 add sqrt def}%optional
```

All curves are to be specified as parametric curves using parameter `t`, and using algebraic expressions rather than PostScript. The names of functions are sometimes not the familiar ones. For example, `Ex` is the exponential function, `ch`, `sh` and `th` are the hyperbolic functions. See the `pstricks-add` documentation. You may define named points on these curves using either `\PointOnCurve` or a `\PointPos()` specification starting with an = sign—see below and the final section of these notes.

4.7 Ticks

```
%  Spacing of large tick marks
\def\xTick{25} % x ticks are 25 x units apart
\def\yTick{5}
% Numbers of subdivisions for small tick marks
\def\xSubTicks{5} % 5 subticks per tick division
\def\ySubTicks{5}
% Ranges of Tick marks--comment out for same as grid
\def\xTickMin{0}
\def\xTickMax{100}
\def\yTickMin{0}
\def\yTickMax{20}
```

The tick settings for each axis are translated into tick parameter settings for `\psaxes`, and applied separately for each axis.

4.8 Grid

```
%Grid options
\def\xGrid{25} % each grid rectangle is 25 x units
\def\yGrid{5}% each grid rectangle is 5 y units
\def\xGridDiv{5}% number of horiz. dots on each grid rectangle
\def\yGridDiv{5}% number of vert. dots on each grid rectangle
% Ranges of Grid marks
% comment out for same as \xmin...
\def\xGridMin{0}
\def\xGridMax{100}
\def\yGridMin{0}
\def\yGridMax{20}
```

This offers more flexibility than the PSTricks `\psgrid` command, and is especially useful when the scales on the axes are not comparable. The keywords `GridColor` and `GridDotRadius` allow further customization. Their default values are respectively `gray`, `.3pt`. To make a dotted grid, use the keyword `showGrid`. (The keyword `graphPaper` must be set to `false`.)

There is another form of grid resembling graph paper. It uses the same commands (`\xGrid, \xGridDiv, \xGridMin` etc) to determine the layout, and takes account of the following keywords.

- `GridMajorLinewidth` determines the width of the major lines. It defaults to `.5pt`.
- `GridMinorLinewidth` determines the width of the minor lines. It defaults to `.3pt`.
- `GridMajorLinecolor` determines the color of the major lines. It defaults to `blue!35` — 35% blue, 65% white.
- `GridMinorLinecolor` determines the color of the minor lines. It defaults to `gray!25` — 35% gray, 65% white.
- `graphPaper` causes the graph-paper to be drawn.

Both types of grid are drawn before all other graphic elements.

4.9 Startup commands

You may insert the following command before `\begin{psgrapha}`.

```
\def\AtGraphaStart{<commands>}
```

The `<commands>` so placed will be executed before all other picture placement commands. For example, to get a light gray background, you could insert

```
\def\AtGraphaStart{\psframe*[linecolor=black!10](\xmin,\ymin)
(\xmax,\ymax)}
```

4.10 Labels

```
\def\LabelCount{3}
\Label(1)={$v=v(P)$}
\LabelPos(1)={\xmax,\ymax}
%\LabelPos(1) may be specified also as {\subxaxis{x}}, which
% specifies a point with specified x coordinate and y
% coordinate determined to be xLabelSpace below x axis.
% Similarly, {\subsubxaxis(x)} specifies point
% \xLowerLabelSpace below x axis, and {\subyaxis{y}} gives
% point yLabelSpace to left of y axis.
\LabelRefPt(1)={tr} % top right
%---also B, b, t, r, c, tl, Bl, Br, bl, br.
\Label(2)={(percent hydrogen)}
\LabelPos(2)={\subsubxaxis{\xmax}}
\LabelRefPt(2)={Br} % Baseline right
\Label(3)={(feet per second)}
\LabelPos(3)={[nodesep=.2in]YA2}
\LabelRefPt(3)={l}
\LabelBlankBG(3)={1}%blank the background
\LabelRotation(3)={45}% and rotate through 45 deg
```

The `\LabelRotation()` value specifies the angle of rotation to be applied to the label. The `\LabelPos()` values may be any coordinate form understood by `\SpecialCoor`. The `\LabelRefPt` takes one of two separate forms, determining whether `pstricks` places the label using the `\rput` macro or the `\uput` macro. If `\LabelRefPt` is set to one of the values `B, b, t, r, c, tl, tr, Bl, Br, bl, br` then the item is placed with `\rput`, so that

for example, with value `Bl` (Baseline, left), the left edge baseline of the text is translated to `\LabelPos`. If, on the other hand, `\LabelRefPt={10pt;30}` (anything containing a semi-colon), then the label will be set using `\uput` so that it is at distance `10pt` from `\LabelPos` in a direction making an angle of `30` degrees to the positive x axis. The form {`;30`} is also understood—the distance is set to the current value of the parameter `labelsep`, which defaults to `5pt`. This mode is highly suited to placing the name of a point beside the point. A final optional item is `\LabelBlankBG` which, if set to 1, blanks out the background to the label before applying the label.

5 The order of placement

The arrays of information should be defined prior to `\begin{psgrapha}`. The effect of that statement is:

- Fill in any missing values from existing definitions. For example, `psgrapha` arguments are used to override definitions of `\xmin`, `\picwidth`, etc, and if `\xTickMin` is undefined, it is set equal to `\xmin`.
- Compute `xunit` and `yunit` taking into account `llx` etc.
- Create a `\pspicture` environment of the correct size to hold the graphic elements and labels.
- Define the key positions in the picture as nodes.
- If a macro `\AtGraphaStart` has been defined, its code is executed next. A colored background for the picture may be generated this way.
- If `\showGrid=true`, lay down a dotted grid. If `\graphPaper=true`, draw a graph-paper style grid.
- If a background `.eps` is specified, insert it now, generating it with `gnuplot` if indicated.
- If the keyword `fullcanvas=true`, draw a slightly off-white frame around the edge of the canvas so that the resulting picture will always contain the full canvas, even if trimming is applied by your LaTeX engine.
- Draw axes and ticks as specified by values of the keywords `xAxis`, `yAxis`, `xTicks, yTicks`. If `\xNumCount<0`, add automatic numeric

labels to x axis. If \yNumCount<0, add automatic numeric labels to y axis.

- If the pst-plot keywords xAxisLabel, yAxisLabel are not empty, place them as in psgraph. (By default, they are respectively x, y, and the placement depends on xAxisLabelPos, yAxisLabelPos.) Otherwise, place the axis labels using the macros (not keywords) \xAxisLabel and \yAxisLabel.
- If \CurveCount>0, draw the corresponding curves.
- If \PointCount>0, add the nodes they define to the node list. Points are named in index order, so the point defined in \PointName(1) may be used in defining \PointName(2).
- If \FrameCount>0, place any frames defined by \FrameStart() etc.
- If \LineCount>0, place any lines defined by \LineStart() etc.
- If \xNumCount>0 or \yNumCount>0 add numeric labels on the corresponding axis.
- If \DotCount>0, place any dots defined by \DotPos() etc.
- If \CircleCount>0, place any circles defined by \CirclePos() etc.
- Labels specified with \Label() are placed.

Following that, all pstricks commands specified before \end{psgrapha} are executed.

6 New macros

6.1 PointOnCurve

\PointOnCurve{<curve index>}{<tvalue>}{<node name>} gives a name to the point at the specified value of t. For example:

```
\PointOnCurve{1}{1.2}{P}%assigns name P to xx(1),yy(1)
```

This macro must be used only within the psgrapha environment—it must not be used prior to the line \begin{psgrapha}. An equivalent form that

can be used outside the psgrapha environment is to use \PointPos() specified with an = sign—see the final section of these notes.

6.2 LinesToAxes

\LinesToAxes[<optional keywords>](<coords>) draws lines from the point to both coordinate axes. For example:

```
\LinesToAxes[linestyle=dashed,linewidth=.4pt](50,14.9)
```

This macro must be used only within the psgrapha environment—it must not be used prior to the line \begin{psgrapha}.

6.3 subxaxis, subsubxaxis, subyaxis

These macros simplify the placement of arbitrary labels beside the axes.

\subxaxis{<xval>} expands to the coordinates of a point at (xval,y) at distance xLabelSpace below the x axis;
\subsubxaxis{<xval>} expands to the coordinates of a point at (xval,y) at distance xLowerLabelSpace below the x axis;
\subyaxis{<yval>} expands to the coordinates of a point at (x,yval) at distance yLabelSpace to the left of the y axis;
These macros may be used prior to \begin{psgrapha}, where they may be used to specify positions of points, lines and labels.

6.4 psrline

The first coordinate pair defines the start of the line, and each succeeding coordinate pair represents a displacement from the previous point rather than the absolute coordinates of the next point. This is quite useful for adding dimensional markers. For example,

```
\psrline[linecolor=green]{->}(3,4)(1cm,0)
```

makes a horizontal green line (with an arrowhead) 1cm long starting at (3,4).

This macro must be used only within the psgrapha environment—it must not be used prior to the line \begin{psgrapha}.

7 Background pictures

There is an option to include a background graphic, such as a one generated by gnuplot, to draw a graph using functions or options unavailable within pstricks. To start the process, before the `\begin{psgrapha}` command, insert the lines

```
\def\background{myepsfile}%actual file has extension .eps
% the following four lines are optional
% If omitted, the values for \xTickMin, etc are substituted
%\def\bgxmin{}
%\def\bgxmax{}
%\def\bgymin{}
%\def\bgymax{}
%\def\bgstyle{set style line 1 lc -1 lw 3.32}
\def\bgfn{<expression in x>}
% gnuplot expression, eg x**3, not x^3
\def\bginfile{gnuplin1.txt}
\def\bgoutfile{myepsfile.eps}
```

The first line above will force `\includegraphics{myepsfile}`, and make an entry in the `.log` telling you the size of the inserted image, in inches.

A `gnuplot` input file will be generated provided the last three lines above are uncommented, generating a new file named `gnuplin1.txt` containing the following lines (but changing the entries appropriately).

```
set terminal postscript eps size <size>
set output 'myepsfile.eps'
unset title; unset border; unset xtics; unset ytics
unset key; set lmargin 0; set rmargin 0
set tmargin 0; set bmargin 0
set xrange [<bgxmin>:<bgxmax>]; set yrange [<bgymin>:<bgymax>]
set style line 1  lc -1 lw 3.32
#contents of \bgstyle inserted here
plot <bgfn> w lines ls 1
```

(With default settings, output to a PostScript `eps` file named `myepsfile.eps`. A `gnuplot` linewidth (`lw` 1 translates to a PostScript linewidth of about .24pt, so the `lw 3.32` produces a linewidth of about .8pt in the `eps` file. Instead of 3.32, the program uses `4.15\pslinewidth`.)

There is an optional setting which allows you to over-ride the `set style` line. Just add a line like

```
\def\bgstyle{set style line 1 lc <a> lw <b>}
```

substituting your own color specification for `<a>` and your own `gnuplot` linewidth parameter for `<b>`. In fact, since you may place arbitrary `gnuplot` commands in `\def\bgstyle{}`, (separated by ;) it is possible to specify a parametric plot by writing, for example

```
\def\bgstyle{set parametric; set trange [0:2*pi]}
\def\bgfn{cos(t),2*sin(t)}
```

or to plot a data file `myfile.txt` using, for example

```
\def\bgfn{'myfile.txt'}
```

or plot more than one graph, specifying one or more complete `plot` statements of your own to add to `\bgstyle`. You will have to learn a bit of `gnuplot` to do this effectively.

If you are willing to enable `\write18`, usually by setting the command line option `--shell-escape` to the processing engine, TEX will try to run `gnuplot` on the file just created, making the `eps` file automatically. Otherwise, you will have to run `gnuplot` manually—type the following in a Terminal window, and press `<return>`:

```
gnuplot gnuplin1.txt
```

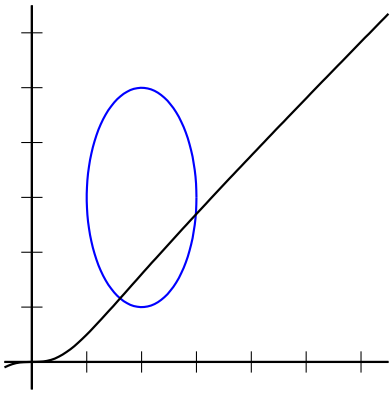
which should produce a file by the name `myepsfile.eps` containing the graph only, with no space above, below, or to the sides, and no axes. The graphic will be included in the right place, ready for adding labels and ticks in the usual way.

Once the background graphic is in its final form, you may avoid regenerating it every time by commenting out one of the three lines defining `\bgfn`, `\bginfile`, `\bgoutfile`.

7.1 A `gnuplot` example

Here is an example file for a `gnuplot`-drawn picture with two separate graphs, one parametric in blue and the other a function graph in black, using the methods described above.

```
\documentclass[dvips]{article}
\usepackage{graphicx}
\usepackage[dvipsnames]{pstricks}
\usepackage{pst-all,pst-grapha}
\pagestyle{empty}
\begin{document}
\def\background{gnuepsfile}%file will have extension .eps
\def\bgstyle{set style line 1 lc -1 lw 3.32 ;% used by bgfn
set parametric ; set trange [0:2*pi]; %
plot 2+cos(t),3+2*sin(t) lc 3 lw 3.2; % ellipse in blue (lc 3)
unset parametric} % now back to main plot in style 1
\def\bgfn{x**3/(1+x**2)}%gnuplot expression to plot
\def\bginfile{gnuplin3.txt}
\def\bgoutfile{gnuepsfile.eps}\psset{}%
\begin{psgrapha}(-.5,-.5)(6.5,6.5){2in}{2in}\end{psgrapha}
\end{document}
```



8 Trigonometric axis option

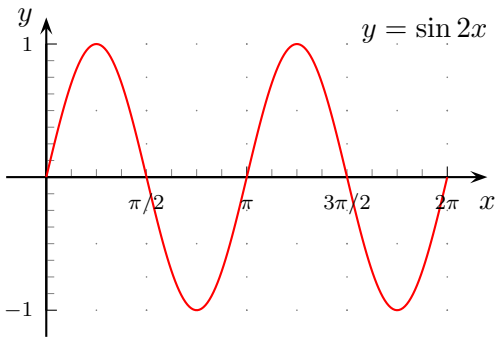
If `trigx=true` is set before starting `psgrapha`, the specified x range values `\xmin` and `\xmax` remain unchanged, but the labels are modified by a factor of π in the display. For example `\xNum(3)={5/2}` will print as the label $5\pi/2$ at $x = 2.5$. As a result, the parametric curves and plots have to adjusted to give correct results. For a (mathematical) parametric curve $x = x(t), y = y(t)$, $a \leq t \leq b$, one need only replace $x(t)$ by $x(t)/\pi$. It is your responsibility to make this adjustment when `trigx=true`. See the example later in these notes.

9 Trigonometric examples

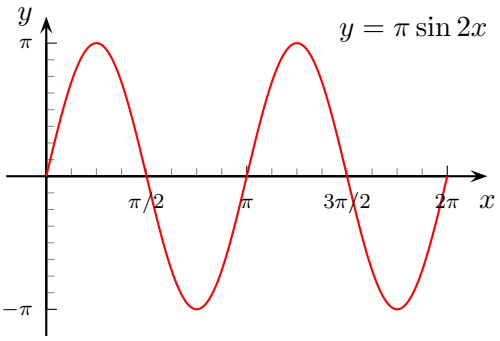
Here are two examples, one using trigonometric labels on the x axis only, the other using trigonometric labels on both. It's important to note that the coordinates underlying the picture omit the scaling by π, so to refer to $(\pi/2, 0)$, one specifies simply (.5,0).

```
\psset{%
trigx=true,
xAxisLabel=,%unset the normal x axis label used by psgraph
yAxisLabel=,%unset the normal y axis label used by psgraph
axesarrows=->,%affect only axes
labelFontSize=\scriptstyle,%
tickpref=A,%ticks pointing into main quadrant
yLabelSpace=5pt,% separation of y axis numbers from y axis
xLabelSpace=12pt,%baseline below x axis for numbers on x axis
llx=1cm,% left border
lly=.6cm,%lower border
urx=.1in,%right border
ury=.1in%upper border
}%
\begin{document}
\def\picwidth{3in}
\def\picheight{2in}
% x- and y-ranges
\def\xmin{-.2}
\def\xmax{2.2}% will appear to be 2.2\pi
\def\ymin{-1.2}
\def\ymax{1.2}
%  Spacing of large tick marks
\def\xTickMin{0}\def\xTickMax{2}
\def\yTickMin{-1}\def\yTickMax{1}
\def\xTickVal{1}
\def\yTickVal{1}% will appear to be \pi
% Numbers of subdivisions for small tick marks
\def\xSubTicks{8}
\def\ySubTicks{8}
% Numbers on the x-axis
\def\xNumCount{4}
\xNum(1)={1/2}% place \pi/2 at x=1/2
```

```
\xNum(2)={1}
\xNum(3)={3/2}
\xNum(4)={2}
% Numbers on the y-axis
\def\ynumCount{2}
\yNum(1)={-1}
\yNum(2)={1}
%Grid options
\def\xGrid{.5} % width of grid rectangles
\def\yGrid{.5}% height of grid rectangles
\def\xGridDiv{2}
% number of dots on the bottom and top of grid rectangles
\def\yGridDiv{2}
% number of dots on the sides of grid rectangles
% Ranges of Grid marks
%leave commented out for same as \xmin...
\def\xGridMin{0}
\def\xGridMax{2}
\def\yGridMin{-1}
\def\yGridMax{1}
% Axis labels
\def\xAxisLabel{$x$}
\def\yAxisLabel{$y$}
\def\LabelCount{1}
\Label(1)={$y=\sin 2x$}
\LabelPos(1)={\xmax,\ymax}
\LabelRefPt(1)={tr}
% top right---also B, b, t, r, c, tl, Bl, Br, bl, br.
% Curves--specify algebraically and parametrically, not with PS
\def\CurveCount{1}
% x(t)=t/\pi, y(t)=sin(2t), 0\le t\le 2\pi
\Curvex(1)={t/3.1416}
\Curvey(1)={sin(2*t)}
\CurveMin(1)={0} % Initial value of the parameter t  for curve 1
\CurveMax(1)={6.283}  % Final value of the parameter t  for curve 1
\CurveParams(1)={linecolor=red,plotpoints=200}
\begin{psgrapha}[showGrid,arrowscale=1.5]
% other pstricks commands may be placed here
\end{psgrapha}
```



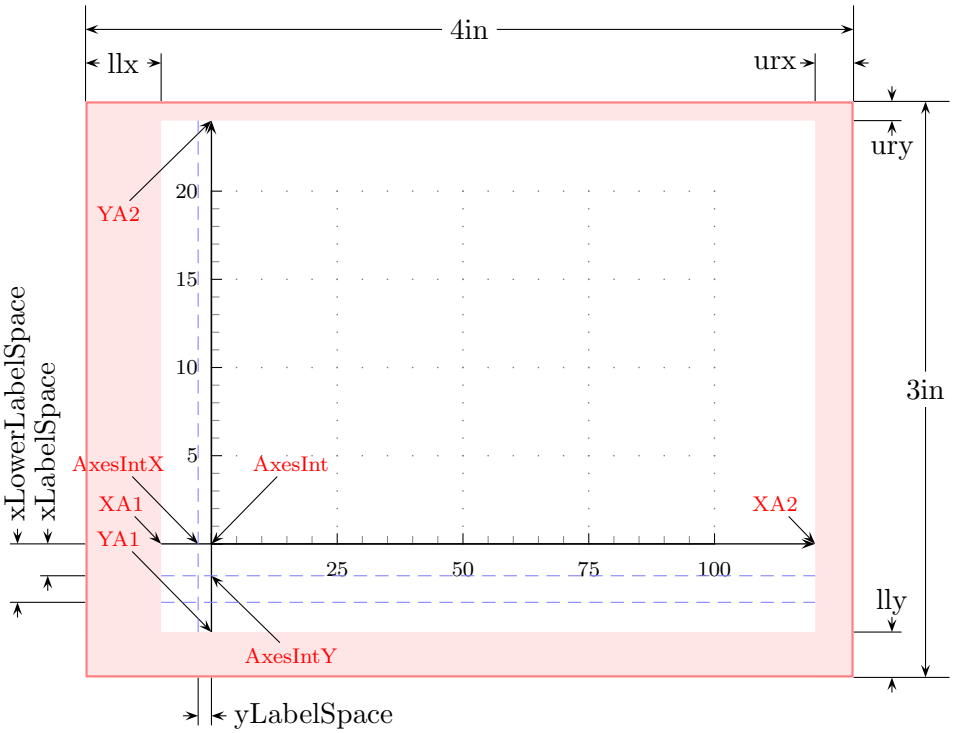
By adding the keyword `trigy=true` one gets a factor of π on the y axis. The function label then need to be changed to $y = \pi \sin 2x$, though the curve definition does not require change.



9.1 Reusing data

The astute reader will recognize that we just drew two similar pictures with just the addition of one command to the existing data. This can have bad effects, carrying data over into other pictures in a document. For this reason, there is a command `\resetpsgrapha` which gives null or default values to a number of key variables. It does not wipe out all existing data, but makes data spill-over less problematic.

10 The psgrapha canvas



Points marked in red are the named locations on the `psgrapha` canvas.

11 A TeX workflow

If you are using TEXShop to process your psgrapa projects, you may find it useful to make the following personal script. It trims away almost all white space around the output and converts it to a pdf.

```
#! /bin/bash
docroot=`basename $1 .tex`
```

```
thedir=`dirname $1`
pushd $thedir
if [ ! -f $docroot.tex ]; then
echo "No tex file $docroot.tex"
exit 1
fi
latex --shell-escape --synctex=1 $docroot
if [ ! -f $docroot.dvi ]; then exit 1
fi
dvips -R0 -Poutline -o$docroot.ps $docroot.dvi
if [ ! -f $docroot.ps ]; then exit 1
fi
ps2eps -l -f -g -r=3600 $docroot.ps
if [ ! -f $docroot.eps ]; then exit 1
fi
epstopdf $docroot.eps
```

This script should be saved using Unix line endings in a folder that's in the Unix `PATH`—eg, as `/usr/local/bin/makeeps`. You must set it to be executable: in a terminal window, type

```
chmod 755 /usr/local/bin/makeeps
```

If you are using TEXShop on OS X, then, in TEXShop preferences, in the `Misc/Personal Script` section, enter `/usr/local/bin/makeeps` in the LaTeX Program field, and under the Typesetting tab, change Default Script to Personal Script. The output will be both an `.eps` and a `.pdf` trimmed of most white space.

12 Node Expressions

When using the data array method for entering graph information, node expressions may be used to specify positions in almost all cases—specifically, they may be used to specify `\PointPos()`, `\FrameStart()`, `\FrameEnd()`, `\LineStart()`, `\LineEnd()`, `\LineDelta()`, `\DotPos()`, `\CirclePos()` and `\LabelPos()`. Node expressions are described in the documentation to `pst-node`. They are simple linear combinations like

```
.3(1,2)+.2(!90 cos 90 sin)-.6(3;30)-(P)
```

where each expression in parentheses must be understood by `\SpecialCoor`. There is one more special form that may be used with `\PointPos()`, which names a point on a curve. Suppose we have defined a curve by

```
\def\CurveCount{1}
\Curvex(1)={t}
\Curvey(1)={t^3+Ex(-t)}
\CurveMin(1)={0} % Initial value of parameter t for curve 1
\CurveMax(1)={2} % Final value of parameter t for curve 1
```

Then

```
\def\PointCount{1}
\PointName(1)={P1}
\PointPos(1)={=1:1.2}
```

gives the name `P1` to the point on curve 1 at $t = 1.2$.

Note that there are a several pre-defined nodes within the `psgrapha` environment which serve as starting points for constructions. They are

`AxesInt` The position where the axes cross, which in simple cases will be (0,0).

`AxesIntX` The position on the x axis marking the right edge for labels on the y axis. (Controlled by `yLabelSpace`.)

`AxesIntY` The position on the y axis marking the lower edge for labels on the x axis. (Controlled by `xLabelSpace`.)

`AxesIntYY` The position on the y axis marking the lower edge for lower level labels on the x axis. (Controlled by `xLowerLabelSpace`.)

`AxesIntX` The position on the x axis marking the edge for labels on the y axis.

`XA1,XA2` Left and right ends of x axis.

`YA1,YA2` Lower and upper ends of y axis.

`PSTxx` Nine additional named positions are created: `PSPbl`, `PSPbl`, `PSPcl`, `PSPbr`, `PSPcl`, `PSPcc`, `PSPcr`, `PSPtl`, `PSPtc`, `PSPtr`. They are set at the corners and centers of the entire `\pspicture`. For example, `PSPtc` means 'top center'. These node names were predefined in `pstricks` by

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

and can be redefined similarly in your personal settings.

`Ivec,Jvec` These nodes may be thought of as horizontal and vertical displacement vectors having the width and height of the canvas. This allows specification of a point in the canvas 30% from left to right and 60% from bottom to top by the node expression

```
\PointPos(1)={(PSTbl)+.3(Ivec)+.6(Jvec)}
```