The xpicture package*
(http://www.upv.es/~rfuster/xpicture)
Several extensions of the picture standard environment
Reference manual and documented source

Robert Fuster
Universitat Politècnica de València
rfuster@mat.upv.es

2012/12/17

Abstract

The xpicture package extends the graphic abilities of the picture standard environment and packages pict2e and curve2e, adding the ability to work with arbitrary In addition to other utilities, the greater interest of xpicture lies in its capacity to draw function graphs, conic sections and arcs, and parametrically defined curves.

This is the technical documentation and reference manual of package xpicture, but not its user manual. User manual is on file xpicture-doc.pdf, distributed together with the package.

Contents

1 Introduction **3**

2 Usage **3**

3 The user interface **4**
3.1 Color selection 4
3.2 Reference systems 4
3.3 The Picture environment 5
3.4 Cartesian and polar coordinate axes and grids 5
3.4.1 The style of the axes 5
3.4.2 Axes position 5
3.4.3 Style of numerical labels 6
3.4.4 Position of numerical labels 6
3.4.5 Style of cut marks 6

*This document corresponds to xpicture 1.2a, dated 2012/12/17.

3.4.6 Grid style . . . 6
3.4.7 Removing cut marks, labels and grids . . . 7
3.5 Directly printing cuts and labels . . . 7
3.6 \put and \multiput extensions . . . 7
3.7 Drawing lines, vectors and polylines . . . 9
3.7.1 Lines and vectors . . . 9
3.7.2 Polylines and polygons . . . 9
3.8 Drawing curves . . . 9
3.8.1 Conic sections and arcs . . . 9
3.8.2 Real variable functions . . . 11
3.8.3 Parametrically defined curves . . . 11
3.8.4 Drawing curves from a table of values . . . 11

4 Implementation **11**
4.1 Package options . . . 12
4.2 Booleans for some command options . . . 12
4.3 Required packages . . . 13
4.4 Error, Warning and Info messages . . . 13
4.5 Internal counters and lengths and a special number . . . 13
4.6 Declarations and parameters controlling axes style . . . 14
4.7 Color selection . . . 14
4.8 Reference systems . . . 14
4.9 Coordinates . . . 15
4.10 The Picture environment . . . 16
4.11 \put extensions . . . 17
4.12 \multiput extensions . . . 24
4.13 Strigth lines and vectors . . . 25
4.14 Polygons and polylines . . . 27
4.15 Quadratic curves . . . 28
4.16 Conic sections and arcs . . . 30
4.17 Graphing functions . . . 33
4.18 Graphing parametric curves . . . 34
4.19 Cartesian axes and grids . . . 35
4.20 Polar grids . . . 43
4.21 Configurable parameters . . . 45
4.22 Commands to be ignored if draft option or \draftPicture declaration is active . 47

5 Change history **48**

1 Introduction

The `xpicture` package introduces several new graphical instructions, and some enriched versions of standard instructions used inside the `picture` environment. All these new instructions can be classified as follows:

1. Reference systems and coordinates:
   - Declaration and use of different reference systems, with Cartesian or polar coordinates.
   - Instructions to show Cartesian or polar reference systems.
2. An alternative to the `picture` environment, compatible with the new reference systems.
3. Alternative instructions or extensions of the standard `picture` commands and those defined by the packages `pict2e` and `curve2e`:
   - Enriched versions of marks `\put` and `\multiput`, providing an adequate control of the precise position in which objects are composed (this functionality is especially useful in the composition of not strictly graphical objects, such as formulas or labels).
   - Instructions for drawing straight segments, vectors (in any direction and using any reference system), polylines, polygons, regular and arbitrary polygons .
4. Regular curves:
   - Instructions for drawing conic sections (circles, ellipses, hyperbolas and parabolas) and arcs of these curves.
   - Instructions to graph functions and parametrically defined curves (this is the most interesting feature of this package).

To enjoy this package you need to have an adequate knowledge of the commands defined in packages `calculator` and `calculus`, especially concerning to the definition of functions and operations with functions.

2 Usage

This package is loaded as usual, using the instruction `\usepackage[`⟨*list of options*⟩`]{xpicture}`. Then, packages `pict2e`, `curve2e`, `xcolor`, `calculator`, and `calculus` are automatically loaded. This package is compatible with any system that supports `xcolor` and `pict2e` packages.[1]
Options are passed directly to packages `pict2e`, `curve2e`, and `xcolor`, excerpt option `draft`, which disables all the instructions defined in this package, replacing each picture set in a `Picture` environment by a blank rectangle.[2] Using this option is very convenient throughout the production of the document, since the composition of the drawings slows considerably the compilation time. The `\draftPictures` declaration performs a similar work, allowing the user to locally disable `Picture` commands.

[1] Earlier versions supports `dvi` option, which was compatible with a pure `dvi` output, but this possibility has been eliminated in version 2.1a, because it was too expensive and probably unhelpful.

[2] If you use an instruction not directly defined by `xpicture`, this instruction may take effect.

An interesting option (from package `pict2e`) is `pstarrows`; if used, arrowheads in vectors are drawn in PSTricks style (instead of the standard LaTeX style).
If exists, the local configuration file `xpicture.cfg` is loaded. This file allows the user to customize all configurable `xpicture` parameters; if you want to use it, copy the file `xpicture.cfgxmpl`, which is distributed along with package `xpicture`, call your copy as `xpicture.cfg`, put it in your local `texmf` tree and edit this file to modify everything agreed.

3 The user interface

3.1 Color selection

`\pictcolor{`⟨*color*⟩`}` select a color without spurious spaces.
Example: `\pictcolor{blue}`

3.2 Reference systems

`\referencesystem(`⟨*x0,y0*⟩`)(`⟨*x1,y1*⟩`)(`⟨*x2,y2*⟩`)` selects the affine reference system with origin in point (*x0,y0*) and coordinate vectors (*x1,y1*) and (*x2,y2*).
Coordinates are refered to the standard reference system.
Example: `\referencesystem(0,0)(1,-1)(1,1)`

`\changereferencesystem(`⟨*x0,y0*⟩`)(`⟨*x1,y1*⟩`)(`⟨*x2,y2*⟩`)` selects the affine reference system with origin in point (*x0,y0*) and coordinate vectors (*x1,y1*) and (*x2,y2*).
Coordinates are refered to the active reference system.
Example: `\changereferencesystem(0,0)(1,-1)(1,1)`

`\translateorigin(`⟨*x0,y0*⟩`)` translates origin to the given point.
Coordinates are refered to the active reference system.
Example: `\translateorigin(2,-3)`

`\rotateaxes{`⟨*angle*⟩`}` rotates the axes. The angle parameter is interpreted as the rotation angle in radians (if the `\radiansangles` declaration is active) or in sexagesimal degrees (if the `\degreesangles` declaration is active).
Coordinates are refered to the active reference system.
Example: `\rotateaxes{\numberQUARTERPI}`

`\symmetrize{`⟨*angle*⟩`}` performs a symmetry, being *angle* the angle between the symmetry axis and the x axis. The `\radiansangles` and `\degreesangles` declarations determine if angles are interpreted as radians or degrees.
Coordinates are refered to the active reference system.
Example: `\symmetrize{\numberPI}`

`\radiansangles` declares that angles are measured in radians (default).

`\degreesangles` declares that angles are measured in degrees.

`\cartesianreference` declares Cartesian coordinates (default).

`\polarreference` declares polar coordinates.

`\polarcoor(`⟨*radius,angle*⟩`)(`⟨*x,y*⟩`)` changes from polar to Cartesian coordinates.

3.3 The Picture environment

`Picture` `\begin{Picture}[`⟨*color*⟩`](`⟨*x0,y0*⟩`)(`⟨*x1,y1*⟩`)` starts a picture, refered to rectangle $[x0, y0] \times [x1, y1]$. If optional argument is present, background is colored with this *color*. By default, background is not colored.
Coordinates are refered to the active reference system and are always Cartesian coordinates.
Example: `\begin{Picture}[black!10!white](-3.5,-4)(3.5,4)`

`xpicture` `\begin{xpicture}` is an alias for `\begin{Picture}`.
Example: `\begin{xpicture}[black!10!white](-3.5,-4)(3.5,4)`

`\draftPictures` disables `Picture` commands, showing only the picture area.
Example: `\begin{xpicture}[black!10!white](-3.5,-4)(3.5,4)`

3.4 Cartesian and polar coordinate axes and grids

`\cartesianaxes(`⟨*x0,y0*⟩`)(`⟨*x1,y1*⟩`)` draws the coordinate axes corresponding to the $[x0, y0] \times [x1, y1]$ rectangle.
Example: `\cartesianaxes(-3.25,-4.5)(3.25,4.25)`

`\cartesiangrid(`⟨*x0,y0*⟩`)(`⟨*x1,y1*⟩`)` draws a coordinate grid corresponding to the $[x0, y0] \times [x1, y1]$ rectangle.
Example: `\cartesiangrid(-3.25,-4.5)(3.25,4.25)`

`\polargrid{`⟨*radius*⟩`}{`⟨*circledivs*⟩`}` draws a polar grid. `radius` is the radius of the circle and `circledivs` (an integer) the number of angular divisions.
Example: `\polargrid{3.5}{16}`

3.4.1 The style of the axes

`\axescolor` User can change the axes color by redefining the `\axescolor` declaration.
Example: `\renewcommand{\axescolor}{orange}` (default is `black`).

`\axesthickness` Length determining the thickness of axes (default `1 pt`).
Example: `\setlength{\axesthickness}{1mm}`

`\xunitdivisions` Number of subdivisions of the unit in the x axis (must a positive integer).
Example:`\renewcommand{\xunitdivisions}{5}` (default is `1`).

`\yunitdivisions` Number of subdivisions of the unit in the y axis (must a positive integer).
Example:`\renewcommand{\yunitdivisions}{3}` (default is `1`).

`\runitdivisions` Number of subdivisions of the unit in the polar axis (must a positive integer).
Example:`\renewcommand{\runitdivisions}{3}` (default is `1`).

3.4.2 Axes position

`\internalaxes` Cartesian axes lies on $y = 0$ and $x = 0$ (default).

`\externalaxes` Cartesian axes lies on $y = y0$ and $x = x0$.

3.4.3 Style of numerical labels

`\axeslabelcolor` User can change the color of labels by redefining the `\axeslabelcolor` declaration.
Example: `\renewcommand{\axeslabelcolor}{red}` (default is equal to the axes color).

`\axeslabelsize` User can change the size of labels by redefining the `\axeslabelsize` declaration.
Example: `\renewcommand{\axeslabelsize}{\tiny}` (default is `\small`).

`\axeslabelmathversion` User can change the mathversion of labels by redefining the `\axeslabelmathversion` declaration.
Example: `\renewcommand{\axeslabelmathversion}{bold}` (default is `normal`).

`\axeslabelmathalphabet` User can change the math alphabet of labels by redefining the `\axeslabelmathalphabet` declaration.
Example: `\renewcommand{\axeslabelmathalphabet}{\mathsf}` (default is `\mathrm`).

`\radianspolarlabels` when this declaration is active, angular labels in polar grids are printed in radians (default).

`\degreespolarlabels` when this declaration is active, angular labels in polar grids are printed in degrees.

`\axislabelsep` Distance between tags and cut marks, measured in `\unitlength` units; the distance between axes and tags equals `\ticssize`+`\axislabelsep`. (see description of `\makenotics`).
Example: `\renewcommand{\axislabelsep}{0.3}` (default is `0.1`).

3.4.4 Position of numerical labels

`\xlabelpos{`⟨*position*⟩`}` Relative position of labels in x axis.
Example: `\xlabelpos{t}` (default is `-90`).

`\ylabelpos{`⟨*position*⟩`}` Relative position of labels in y axis.
Example: `\ylabelpos{tl}` (default is `180`).

3.4.5 Style of cut marks

`\ticssize` half the length of main axes cuts.
Example: `\setlength{\ticssize}{3mm}` (default is `4pt`)

`\secundaryticssize` half the length of secundary axes cuts.
Example: `\setlength{\secunadryticssize}{1mm}` (default is `2pt`)

`\ticsthickness` thickness of the marks on axes.
Example: `\setlength{\ticsthickness}{0.5pt}` (default is `1pt`)

`\ticscolor` User can change the color of tics by redefining the `\ticscolor` declaration.
Example: `\renewcommand{\ticscolor}{lightgray}` (default is `black`)

3.4.6 Grid style

`\gridthickness` thickness of the main grid lines.
Example: `\setlength{\gridthickness}{1pt}` (default is `0.4pt`)

`\secundarygridthickness` thickness of the secundary grid lines.
Example: `\setlength{\gridthickness}{0.25pt}` (default is `0.2pt`)

`\gridcolor` User can change the color of main grid lines by redefining the `\ticscolor` declaration.
Example: `\renewcommand{\gridcolor}{blue}` (default is `gray`)

`\secundarygridcolor` User can change the color of secundary grid lines by redefining the `\ticscolor` declaration. Example: `\renewcommand{\secundarygridcolor}{blue}` (default is `lightgray`)

3.4.7 Removing cut marks, labels and grids

`\maketics` when this declaration is active, divisions of axes are marked (default).

`\makenotics` when this declaration is active, divisions of axes are not marked. In this case, the distance between axes and tags equals `\axislabelsep`.

`\makelabels` when this declaration is active, numerical labels are printed (default).

`\makenolabels` when this declaration is active, numerical labels are not printed.

`\makenogrid` If the `\makenogrid` declaration is active, then `\cartesianaxes` plots only the axes (default).

`\makegrid` If the `\makegrid` declaration is active, then `\cartesianaxes` plots a Cartesian grid. In this case, `\cartesianaxes` is equivalent to `\cartesiangrid`.

3.5 Directly printing cuts and labels

`\plotxtic{`⟨*x-coor*⟩`}` plot a tic for the given x coordinate.

`\plotytic{`⟨*y-coor*⟩`}` plot a tic for the given y coordinate.

`\printxlabel{`⟨*x-coor*⟩`}{`⟨*label*⟩`}` print the required label at the given x coordinate.

`\printylabel{`⟨*y-coor*⟩`}{`⟨*label*⟩`}` print the required label at the given y coordinate.

`\printxticlabel{`⟨*x-coor*⟩`}{`⟨*label*⟩`}` print a tic and the required label at the given x coordinate.

`\printyticlabel{`⟨*y-coor*⟩`}{`⟨*label*⟩`}` print a tic and the required label at the given y coordinate.

`\plotxtics{`⟨*firstcoor*⟩`}{`⟨*incr*⟩`}{`⟨*bound*⟩`}` plot several x tics, from the initial coordinate *firstcoor*; *incr* is the distance between consecutive tics, and the last tic is not in a position greater than *bound*.

`\plotytics{`⟨*firstcoor*⟩`}{`⟨*incr*⟩`}{`⟨*bound*⟩`}` plot several y tics, from the initial coordinate *firstcoor*; *incr* is the distance between consecutive tics, and the last tic is not in a position greater than *bound*.

`\printxlabels[`⟨*digits*⟩`]{`⟨*firstcoor*⟩`}{`⟨*incr*⟩`}{`⟨*bound*⟩`}` print several x labels, from the initial coordinate *firstcoor*; *incr* is the distance between consecutive label positions, and the last position is not greater than *bound*. The optional argument *digits* is the number of decimal digits to be printed (by default, numbers are printed with its natural number of decimals).

`\printylabels[`⟨*digits*⟩`]{`⟨*firstcoor*⟩`}{`⟨*incr*⟩`}{`⟨*bound*⟩`}` print several x labels, from the initial coordinate *firstcoor*; *incr* is the distance between consecutive label positions, and the last position is not greater than *bound*. The optional argument *digits* is the number of decimal digits to be printed (by default, numbers are printed with its natural number of decimals).

`\printxticslabels[`⟨*digits*⟩`]{`⟨*firstcoor*⟩`}{`⟨*incr*⟩`}{`⟨*bound*⟩`}` print x tics and labels simultaneously.

`\printyticslabels[`⟨*digits*⟩`]{`⟨*firstcoor*⟩`}{`⟨*incr*⟩`}{`⟨*bound*⟩`}` print y tics and labels simultaneously.

3.6 `\put` and `\multiput` extensions

`\cPut{`⟨*position*⟩`}(`⟨*x,y*⟩`){`⟨*object*⟩`}`

`\rPut*{`⟨*position*⟩`}(`⟨*x,y*⟩`){`⟨*object*⟩`}`

`\Put*[`⟨*position*⟩`](`⟨*x,y*⟩`){`⟨*object*⟩`}`

draw *object* in point (*x,y*). Argument *position* fixes the precise position of *object* with respect (*x,y*).
In starred versions objects positioned below the reference point are aligned at a fixed vertical distance (normally, by the baseline). User must decide which is that amount (normally the higher object to be positioned), and introduce it as an argument of the `\highestlabel` declaration.
Example: `\Put*[SSE](1,2){\Ellipse{2}{3}}`

Argument *position* supports the following values:

An integer or decimal number, determining the angle (in degrees) where *object* is placed, (with respect to the reference point (*x,y*)).

Letter `c` which places the center of *object* at (*x,y*)).

Letter or letter combinations `N`, `E`, `S`, `W`, `NE`, `SE`, `SW`, `NW`, `NNE`, `ENE`, `ESE`, `SSE`, `SSW`, `WSW`, `WNW`, `NNW`
Abbreviation of *North*, *East*..., *North-East*..., *North-North-East*...

Letter o letter combinations `t`, `r`, `b`, `l`, `tr`, `br`, `bl`, `tl`, `ttr`, `rtr`, `rbr`, `bbr`, `bbl`, `lbl`, `ltl`, `ttl`
Abbreviation of *top*, *right*..., *top-right*..., *top-top-right*...

Without optional argument *position* (in command `\Put`) the reference point of *object* is placed at (*x,y*)) (in a similar way to the `\put` command).

`\Pictlabelsep` determines the distance between the graphical object and the given point. User can redefine this declaration by typing `\renewcommand\Pictlabelsep{`⟨*number*⟩`}`. This number is interpreted as an amount of `\unitlength`.
Example: `\renewcommand{\Pictlabelsep}{1}` (default is `0.1`).
This distance is understood either as the Euclidean (circular) distance, derived from the 2-norm, or as the distance derived from the ∞-norm (rectangular distance), following these rules:

- If argument *position* is a *compass* argument (like `N` or `SSW`), then circular distance is used.
- If argument *position* is like `t`, `bbl`,... then rectangular distance is used. In all other cases, `\cPut` uses circular distance, `\rPut` uses rectangular distance and `\Put` uses distance established by `\defaultPut`.

`\defaultPut{`⟨*position*⟩`}` fixes the default position for `\Put`, `\multiPut` and `\multiPlot` commands. Argument *position* can be `c` or `r`.
Example: `\defaultPut{r}` (default is `c`).

`\highestlabel{`⟨*text*⟩`}` declares the highest label to be equal to height of *text*.
Example: `\highestlabel{\Huge A}` (default is `\normalfont\normalsize$1$`)

`\multicPut{`⟨*position*⟩`}(`⟨*x,y*⟩`)(`⟨Δx,Δy⟩`){`⟨*n*⟩`}{`⟨*object*⟩`}`

`\multirPut*{`⟨*position*⟩`}(`⟨*x,y*⟩`)(`⟨Δx,Δy⟩`){`⟨*n*⟩`}{`⟨*object*⟩`}`

`\multiPut*[`⟨*position*⟩`](`⟨*x,y*⟩`)(`⟨Δx,Δy⟩`){`⟨*n*⟩`}{`⟨*object*⟩`}`
put *n* copies of *object* in *position* at points ($x0, y0$), ($x0 + \Delta x, y0 + \Delta y$), ($x0 + 2\Delta x, y0 + 2\Delta y$), ..., ($x0 + (n-1)\Delta x, y0 + (n-1)\Delta y$).
Example: `\multicPut{c}(1,2)(1,-1){4}{\xVECTOR(0,0)(1,1)}`

`\multicPlot{`⟨*position*⟩`}{`⟨*object*⟩`}(`⟨*x0,y0*⟩`)(`⟨*x1,y1*⟩`)...(`⟨*xn,yn*⟩`)`

`\multirPlot*{`⟨*position*⟩`}{`⟨*object*⟩`}(`⟨*x0,y0*⟩`)(`⟨*x1,y1*⟩`)...(`⟨*xn,yn*⟩`)`

`\multiPlot*[`⟨*position*⟩`]{`⟨*object*⟩`}(`⟨*x0,y0*⟩`)(`⟨*x1,y1*⟩`)...(`⟨*xn,yn*⟩`)`
put $n+1$ copies of *object* at points `(x0,y0)`, `(x1,y1)`,..., `(xn,yn)`
Example: `\multirPlot{c}{\xVECTOR(0,0)(1,1)}(1,2)(2,1)(3,0)(4,-1)`

3.7 Drawing lines, vectors and polylines

3.7.1 Lines and vectors

`\xLINE(`⟨*x0,y0*⟩`)(`⟨*x1,y1*⟩`)` draws a stright line between points `(x0,y0)` and `(x1,y1)`.
Example: `\xLINE(1,-2)(0,3)`.

`\xVECTOR(`⟨*x0,y0*⟩`)(`⟨*x1,y1*⟩`)` draws an arrow from point `(x0,y0)` to point `(x1,y1)`.
Example: `\xVECTOR(1,-2)(0,3)`.

`\xtrivVECTOR(`⟨*x0,y0*⟩`)(`⟨*x1,y1*⟩`)` draws an arrow from point `(x0,y0)` to point `(x1,y1)`. The arrowhead consists of two lines, controled by the `\arrowsize` declaration.
Example: `\xtrivVECTOR(1,-2)(0,3)`.

`\xline(`⟨*x,y*⟩`){`⟨*size*⟩`}`

`\xvector(`⟨*x,y*⟩`){`⟨*size*⟩`}`

`\xtrivvector(`⟨*x,y*⟩`){`⟨*size*⟩`}`
draw lines, vectors and triv vectors with the standard LaTeX syntax, but without any restriction.
Example: `\Put(1,-2){\xline(-1,5){1}}`

`\zerovector(`⟨*x,y*⟩`)`

`\zerotrivvector(`⟨*x,y*⟩`)`
draw a zero-length vector (an arrowhead) in direction `(x,y)`.
Example: `\Put(0,3){\zerovector(-1,5)}`

`\arrowsize{`⟨*xlen*⟩`}{`⟨*ylen*⟩`}` declares dimensions of triv arrowhead: `xlen` pt is its length, and `ylen` pt is half of its aperture.
Example: `\arrowsize{4}{2}` (default is `xlen=5`, `ylen=2`)

3.7.2 Polylines and polygons

`\Polyline(`⟨*x0,y0*⟩`)(`⟨*x1,y1*⟩`)...(`⟨*xn,yn*⟩`)` draws a polyline with vertices `(x0,y0)(x1,y1)...(xn,yn)`.
Example: `\Polyline(1,1)(2,0)(0,-1)`

`\Polygon(`⟨*x0,y0*⟩`)(`⟨*x1,y1*⟩`)...(`⟨*xn,yn*⟩`)` draws a polygon with vertices `(x0,y0)(x1,y1)...(xn,yn)`.
Example: `\Polygon(1,1)(2,0)(0,-1)`

`\regularPolygon[`⟨*angle*⟩`]{`⟨*radius*⟩`}{`⟨*sides*⟩`}` draws a regular polygon with the given *radius* and *sides*. The optional argument (zero, by default) determines the inclination angle of the first vertex, always measured in degrees.
Example: `\regularPolygon[90]{4}{7}`

3.8 Drawing curves

3.8.1 Conic sections and arcs

`\Circle{`⟨*r*⟩`}` draws the circle $x^2 + y^2 = r^2$.

Example: `\Circle{2.5}`

`\Ellipse{`$\langle a\rangle$`}{`$\langle b\rangle$`}` draws the ellipse $\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$.
Example: `\Ellipse{2}{3}`

`\Hyperbola{`$\langle a\rangle$`}{`$\langle b\rangle$`}{`$\langle xmax\rangle$`}{`$\langle ymax\rangle$`}` draws the hyperbola $\frac{x^2}{a^2}-\frac{y^2}{b^2}=1$.
Variables x and y are limited, respectively, to the $[-xmax, xmax]$ and $[-ymax, ymax]$ intervals. This curve is well defined if the parameter $xmax$ is greater than a. Otherwise, `xpicture` returns an error message and does not draw any curve.
Example: `\Hyperbola{2}{3}{5}{5}`

`\rHyperbola{`$\langle a\rangle$`}{`$\langle b\rangle$`}{`$\langle xmax\rangle$`}{`$\langle ymax\rangle$`}` draws the *right* branch of hyperbola $\frac{x^2}{a^2}-\frac{y^2}{b^2}=1$.
(parameters are restricted as in `\Hyperbola`).
Example: `\rHyperbola{2}{3}{5}{5}`

`\lHyperbola{`$\langle a\rangle$`}{`$\langle b\rangle$`}{`$\langle xmax\rangle$`}{`$\langle ymax\rangle$`}` draws the *left* branch of hyperbola $\frac{x^2}{a^2}-\frac{y^2}{b^2}=1$.
(parameters are restricted as in `\Hyperbola`).
Example: `\rHyperbola{2}{3}{5}{5}`

`\Parabola{`$\langle a\rangle$`}{`$\langle xmax\rangle$`}{`$\langle ymax\rangle$`}` draws the parabola $x=ay^2$.
Variable x is limited, respectively, to the $[0, xmax]$ (if a is positive) or $[-xmax, 0]$ (if negative) interval. $[-ymax, ymax]$ intervals.
Example: `\Parabola{2}{5}{5}`

`\circularArc{`$\langle r\rangle$`}{`$\langle angle1\rangle$`}{`$\langle angle2\rangle$`}` draws the arc of circle $x=r\cos t, y=r\sin t,\ t\in[angle1, angle2]$ (the arc of the circle centered at $(0,0)$ with radius r and limited between *angle1* and *angle2*).
Example: `\circularArc{3}{0}{\numberSIXTHPI}`

`\xArc` is an alias for `\circularArc`.
Example: `\xArc{3}{0}{\numberSIXTHPI}`

`\ellipticArc{`$\langle a\rangle$`}{`$\langle b\rangle$`}{`$\langle angle1\rangle$`}{`$\langle angle2\rangle$`}` draws the arc of ellipse $x=a\cos t, y=b\sin t,\ t\in[angle1, angle2]$ (the arc of the ellipse centered at $(0,0)$ with semiaxes a and b and limited between *angle1* and *angle2*).
Example: `\ellipticArc{2}{3}{-\numberSIXTHPI}{\numberSIXTHPI}`

`\rhyperbolicArc{`$\langle a\rangle$`}{`$\langle b\rangle$`}{`$\langle y0\rangle$`}{`$\langle y1\rangle$`}` draws the right arc of hyperbola $\frac{x^2}{a^2}-\frac{y^2}{b^2}=1$ included between $y=y0$ and $y=y1$.
Example: `\rhyperbolicArc{2}{3}{-2}{2}`

`\lhyperbolicArc{`$\langle a\rangle$`}{`$\langle b\rangle$`}{`$\langle y0\rangle$`}{`$\langle y1\rangle$`}` draws the left arc of hyperbola $\frac{x^2}{a^2}-\frac{y^2}{b^2}=1$ included between $y=y0$ and $y=y1$.
Example: `\lhyperbolicArc{2}{3}{-2}{2}`

`\parabolicArc{`$\langle a\rangle$`}{`$\langle y0\rangle$`}{`$\langle y1\rangle$`}` Draw the arc of the parabola $x=ay^2$ included between $y=y0$ and $y=y1$.
Example: `\parabolicArc{2}{-2}{2}`

`\defaultplotdivs{`$\langle divisions\rangle$`}` declares the number of subintervals we divide the domain of curves when plotting conic arcs.
Example: `\defaultplotdivs{16}` (default is `8`).

3.8.2 Real variable functions

`\PlotFunction[`⟨*n*⟩`]{`⟨*\functionname*⟩`}{`⟨*t0*⟩`}{`⟨*t1*⟩`}` draws the graph of function `\functionname`(t), $t \in [t0, t1]$. This interval is partitioned in n subintervals (default for n is 2).
Example: `\PlotFunction[16]{\COSfunction}{-\numberTWOPI}{\numberTWOPI}`

`\PlotPointsOfFunction{`⟨*n*⟩`}{`⟨*\functionname*⟩`}{`⟨*t0*⟩`}{`⟨*t1*⟩`}` draws n+1 points of the graph of function `\functionname`(t), $t \in [t0, t1]$.
Example: `\PlotPointsOfFunction{20}{\SQRTfunction}{0}{4}`

`\pointmarkdiam` is the size of points printed by `\PlotPointsOfFunction`, measured in `\unitlength` units. It may be redefined with a `\renewcommand` declaration.
Example: `\renewcommand{\pointmarkdiam}{0.3}`

`\pointmark` is the symbol printed at every point by `\PlotPointsOfFunction`. It may be redefined with a `\renewcommand` declaration.
Example: `\renewcommand{\pointmark}{$\diamond$}`

3.8.3 Parametrically defined curves

`\PlotParametricFunction[`⟨*n*⟩`]{`⟨*\functionname*⟩`}{`⟨*t0*⟩`}{`⟨*t1*⟩`}` draws the graph of parametric curve `\functionname`(t), $t \in [t0, t1]$. This interval is partitioned in n subintervals (default for n is 2).
Example:
```
\ParametricFunction{\F}{\SQUAREfunction}{CUBEfunction}
\PlotParametricFunction[15]{\F}{-2}{2}
```

3.8.4 Drawing curves from a table of values

`\qCurve(`⟨*x0,y0*⟩`)(`⟨*u0,v0*⟩`)(`⟨*x1,y1*⟩`)(`⟨*u1,v1*⟩`)` draws the quadratic curve between points *x0,y0* and *x1,y1* with tangent vectors *u0,v0* nd *u1,v1*.
Example: `\qCurve(1,2)(1,2)(4,3)(-1,1)`

`\PlotQuadraticCurve(`⟨*x0,y0*⟩`)(`⟨*u0,v0*⟩`)(`⟨*x1,y1*⟩`)(`⟨*u1,v1*⟩`)...(`⟨*xn,yn*⟩`)(`⟨*un,vn*⟩`)`
draws a curve through the points `(`⟨*x0,y0*⟩`)`, `(`⟨*x1,y1*⟩`)`,..., `(`⟨*xn,yn*⟩`)` with tangent vectors `(`⟨*u0,v0*⟩`)`, `(`⟨*u1,v1*⟩`)`,..., `(`⟨*un,vn*⟩`)`.
Example: `\PlotQuadraticCurve(1,0)(1,0)(0,1)(0,1)(-1,0)(-1,0)(0,-1)(0,-1)`

`\PlotQuadraticCurve(`⟨*x0,y0*⟩`){`⟨*angle0*⟩`}(`⟨*x1,y1*⟩`){`⟨*angle1*⟩`}...(`⟨*xn,yn*⟩`){`⟨*anglen*⟩`}`
draws a curve through the points `(`⟨*x0,y0*⟩`)`, `(`⟨*x1,y1*⟩`)`,..., `(`⟨*xn,yn*⟩`)` the inclination angles of which, with respect to the x axis, are *angle0, angle1*..., *anglen* (always measured in degrees).
Example: `\PlotQuadraticCurve(1,0){0}(0,1){90}(-1,0){180}(0,-1){270}`

`\PlotxyDyData(`⟨*x0,y0,Dy0*⟩`)(`⟨*x1,y1,Dy1*⟩`)...(`⟨*xn,yn,Dyn*⟩`)` draws a curve through the points `(`⟨*x0,y0*⟩`)`, `(`⟨*x1,y1*⟩`)`,..., `(`⟨*xn,yn*⟩`)` with derivatives *Dy0*, *Dy1*, ..., *Dyn*.

4 Implementation

```
1 ⟨*xpicture⟩
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{xpicture}[2012/12/17 v.1.2a picture environment extensions]
```

4.1 Package options

If the `draft` option is selected, `Picture` environments are shown as a rectangular frame and `xpicture` commands are ignored (Boolean `draft` controls whether this option has been selected).

```
\newif\ifdraft\draftfalse
\DeclareOption{draft}{\drafttrue}
```

All other options are passed to packages `curve2e` and `xcolor` (Old options `dvi`, `pict2e` and `curve2e` have been removed in version 1.2a).

```
\DeclareOption*{%
   \PassOptionsToPackage{\CurrentOption}{curve2e}
    \PassOptionsToPackage{\CurrentOption}{xcolor}}
\ProcessOptions
```

4.2 Booleans for some command options

Booleans used by several declarations controlling the behavior of some `xpicture` commands.

`\ifpolar` True: polar coordinates. False: Cartesian coordinates.

```
\newif\ifpolar\polarfalse
```

`\ifrputstar` True: `\rPut` starred.

```
\newif\ifrputstar\rputstarfalse
```

`\ifdegrees` True: angles mesured in degrees. False: arcs mesured in radians.

```
\newif\ifdegrees\degreesfalse
```

`\iftics` True: coordinate axes include tic marks.

```
\newif\iftics\ticstrue
```

`\iflabels` True: coordinate axes include numeric labels.

```
\newif\iflabels\labelstrue
```

`\ifgrid` True: Cartesian grids.

```
\newif\ifgrid\gridfalse
```

`\ifticslabelsgrid` True: Tics, labels or grid must be printed.

```
\newif\ifticslabelsgrid\ticslabelsgridfalse
```

`\ifinzeroaxes` True: Representation of axes passes through the origin (internal axes). False: external axes.

```
\newif\ifinzeroaxes\inzeroaxestrue
```

`\ifbg` True: Background will be colored.

```
\newif\ifbg\bgfalse
```

4.3 Required packages

```
\RequirePackage{curve2e}
\RequirePackage{xcolor}
\RequirePackage{calculus}
```

4.4 Error, Warning and Info messages

```
\def\xpct@Warnbadpos{%
      \PackageWarning{xpicture}%
        {Argument in \noexpand\defaultPut command must be either
        'c' or 'r'\MessageBreak
         I will no change the default position for
         \noexpand\Put commands}}
\def\xpct@Infopos#1{%
      \PackageInfo{xpicture}%
        {Default position for \noexpand\Put commands changed to #1}}
\def\xpct@WarnIncSys(#1,#2)(#3,#4){%
      \PackageWarning{xpicture}{%
           Incompatible linear system!\MessageBreak
           Tangent lines are parallel}}
\def\xpct@ErrHypCons{%
       \PackageError{xpicture}{%
        Inconsistent parameters in \noexpand\Hyperbola command}{%
        The first and second parameters in a \noexpand\Hyperbola
        command\MessageBreak
        must be, respectively, lesser than the third and
        the fourth ones.}}
\def\xpct@Infocfg{\PackageInfo{xpicture}{%
          Loading local configuration file xpicture.cfg}}
\def\xpct@Infonocfg{\PackageInfo{xpicture}{%
          Local configuration file xpicture.cfg does not exists}}
```

4.5 Internal counters and lengths and a special number

Counters `xpct@counta` and `xpct@countb` will be used by several internal commands (mainly in `while` clauses). `xpct@step` is used when iterating functions plots, and `multiput` by commands extending the `\multiput` command.

```
\newcounter{xpct@counta}
\newcounter{xpct@countb}
\newcounter{xpct@step}
\newcounter{multiput}
```

`\xpct@bxw` `\xpct@bxh` Width and height of certain boxes.

```
\newdimen\xpct@bxw
\newdimen\xpct@bxh
```

`\xpct@maxnum` The largest TEX number.

```
\def\xpct@maxnum{16383.99998}
```

4.6 Declarations and parameters controlling axes style

\makenotics
\maketics
\makenolabels
\makelabels
\makegrid
\makenogrid
\externalaxes
\internalaxes

Four pairs of alternative declarations, switching booleans \iftics, \iflabels, \ifgrid, and \ifinzeroaxes. Defaults are \maketics, \makelabels, \makenogrid, and \internalaxes.

```
\def\makenotics{\ticsfalse}
\def\maketics{\ticstrue}
\def\makenolabels{\labelsfalse}
\def\makelabels{\labelstrue}
\def\makenogrid{\gridfalse}
\def\makegrid{\gridtrue}
\def\externalaxes{\inzeroaxesfalse}
\def\internalaxes{\inzeroaxestrue}
```

\axesthickness

Thickness of axes (it is a length).

```
\newdimen\axesthickness
```

\xpct@axeslabelattrib

Attributes of labels. It is a private declaration, because you can select attributes (size, color and mathversion) of labels independently.

```
\def\xpct@axeslabelattrib{\axeslabelsize%
        \pictcolor{\axeslabelcolor}%
        \mathversion{\axeslabelmathversion}}
```

\ticsthickness
\ticssize
\secundaryticssize
\gridthickness
\secundarygridthickness

Thickness and size of tics and grid lines.

```
\newdimen\ticsthickness
\newdimen\ticssize
\newdimen\secundaryticssize
\newdimen\gridthickness
\newdimen\secundarygridthickness
```

4.7 Color selection

\pictcolor

Declaration \pictcolor supresses spureus spaces when selecting color.

```
\def\pictcolor{\@killglue\color}
```

4.8 Reference systems

\standardreferencesystem

Declaration to select the standard reference system.

```
\def\standardreferencesystem{\referencesystem(0,0)(1,0)(0,1)}
```

\referencesystem
\changereferencesystem
\xpct@xorigin
\xpct@yorigin
\xpct@xI
\xpct@yI
\xpct@xII
\xpct@yII

\referencesystem changes to the affine reference centered in P(#1,#2) with directions (#3,#4) and (#5,#6). These six numbers are stored in \xpct@xorigin, \xpct@yorigin, \xpct@xI, \xpct@yI, \xpct@xII, and \xpct@yII.

```
\def\referencesystem(#1,#2)(#3,#4)(#5,#6){%
        \COPY{#1}\xpct@xorigin
        \COPY{#2}\xpct@yorigin
        \COPY{#3}\xpct@xI
        \COPY{#4}\xpct@yI
        \COPY{#5}\xpct@xII
        \COPY{#6}\xpct@yII}
```

The \changereferencesystem changes from the active reference system.

```
\def\changereferencesystem(#1)(#2)(#3){%
      \refsysPoint(#1)(\xpct@newx,\xpct@newy)
      \refsysVector(#2)(\xpct@newux,\xpct@newuy)
      \refsysVector(#3)(\xpct@newvx,\xpct@newvy)
      \referencesystem(\xpct@newx,\xpct@newy)(\xpct@newux,\xpct@newuy)%
      (\xpct@newvx,\xpct@newvy)}
```

\translateorigin
\rotateaxes
\symmetrize

Translations and orthogonal changes (rotations and symmetries) of reference system.

```
\def\translateorigin(#1){\changereferencesystem(#1)(1,0)(0,1)}
\def\rotateaxes#1{%
      \ifdegrees\DEGREESCOS{#1}\xpct@cosine\DEGREESSIN{#1}\xpct@sine
      \else\COS{#1}\xpct@cosine\SIN{#1}\xpct@sine\fi
      \changereferencesystem%
         (0,0)(\xpct@cosine,\xpct@sine)(-\xpct@sine,\xpct@cosine)}
\def\symmetrize#1{%
      \MULTIPLY{2}{#1}{\xpct@sym}
      \ifdegrees
         \DEGREESCOS{\xpct@sym}\xpct@cosine\DEGREESSIN{\xpct@sym}\xpct@sine
      \else
         \COS{\xpct@sym}\xpct@cosine\SIN{\xpct@sym}\xpct@sine\fi
      \changereferencesystem%
         (0,0)(\xpct@cosine,\xpct@sine)(\xpct@sine,-\xpct@cosine)}
```

4.9 Coordinates

\refsysxyVector
\refsysxyPoint

Canonical coordinates of a point or vector given in Cartesian coordinates (change from the active r.s. to the standard one).

```
\def\refsysxyVector(#1)(#2,#3){%
      \MATRIXVECTORPRODUCT%
         (\xpct@xI,\xpct@xII;\xpct@yI,\xpct@yII)(#1)(#2,#3)}
\def\refsysxyPoint(#1)(#2,#3){%
      \MATRIXVECTORPRODUCT(\xpct@xI,\xpct@xII;\xpct@yI,\xpct@yII)(#1)(#2,#3)
      \VECTORADD(#2,#3)(\xpct@xorigin,\xpct@yorigin)(#2,#3)}
```

\refsyspVector
\refsyspPoint

Canonical coordinates of a point or vector given in polar coordinates.

```
\def\refsyspVector(#1,#2)(#3,#4){%
     \polarcoor(#1,#2)(\xpct@polarx,\xpct@polary)
     \refsysxyVector(\xpct@polarx,\xpct@polary)(#3,#4)}
\def\refsyspPoint(#1,#2)(#3,#4){%
     \polarcoor(#1,#2)(\xpct@polarx,\xpct@polary)
     \refsysxyPoint(\xpct@polarx,\xpct@polary)(#3,#4)}
```

\cartesianreference
\polarreference

Alternative declarations to switch between Cartesian or polar coordinates.
In fact, they define \refsysVector/\refsysPoint to be \refsysxyVector/\refsysxyPoint or \refsyspVector/\refsyspPoint.

```
\def\cartesianreference{%
      \def\refsysVector{\refsysxyVector}%
      \def\refsysPoint{\refsysxyPoint}\polarfalse}
```

```
\def\polarreference{%
        \def\refsysVector{\refsyspVector}%
        \def\refsysPoint{\refsyspPoint}\polartrue}
```

`\polarcoor` `\polarcoor` changes from polar to rectangular coordinates.

```
\def\polarcoor(#1,#2)(#3,#4){%
        \ifdegrees\DEGREESCOS{#2}{\xpct@Px}\DEGREESSIN{#2}{\xpct@Py}
        \else\COS{#2}{\xpct@Px}\SIN{#2}{\xpct@Py}\fi
        \MULTIPLY{\xpct@Px}{#1}{#3}
        \MULTIPLY{\xpct@Py}{#1}{#4}}
```

`\degreesangles` Switches to measure angles in degrees or radians.
`\radiansangles`

```
\def\degreesangles{\degreestrue}
\def\radiansangles{\degreesfalse}
```

4.10 The `Picture` environment

`Picture` `Picture` is an extension of `picture` to refer points to the active reference system. It can take an optional argument (background color).

```
\def\Picture{\@ifnextchar[{\xpct@@Picture}{\xpct@Picture}}
```

`\xpct@Picture` Compute the surrounding box and call `picture` with the appropriate parameters.

```
\def\xpct@Picture(#1,#2)(#3,#4){%
```

First, we determine the standard coordinates of the four vertices

```
        \refsysxyPoint(#1,#2)(\xpct@xzero,\xpct@yzero)
        \refsysxyPoint(#3,#4)(\xpct@xone,\xpct@yone)
        \refsysxyPoint(#1,#4)(\xpct@xtwo,\xpct@ytwo)
        \refsysxyPoint(#3,#2)(\xpct@xthree,\xpct@ythree)
```

Now we calculate the maximum and minimum `x` and `y` coordinates.

```
        \MIN{\xpct@xzero}{\xpct@xone}{\xpct@xmin}
        \MIN{\xpct@xmin}{\xpct@xtwo}{\xpct@xmin}
        \MIN{\xpct@xmin}{\xpct@xthree}{\xpct@xmin}
        \MIN{\xpct@yzero}{\xpct@yone}{\xpct@ymin}
        \MIN{\xpct@ymin}{\xpct@ytwo}{\xpct@ymin}
        \MIN{\xpct@ymin}{\xpct@ythree}{\xpct@ymin}
        \MAX{\xpct@xzero}{\xpct@xone}{\xpct@xmax}
        \MAX{\xpct@xmax}{\xpct@xtwo}{\xpct@xmax}
        \MAX{\xpct@xmax}{\xpct@xthree}{\xpct@xmax}
        \MAX{\xpct@yzero}{\xpct@yone}{\xpct@ymax}
        \MAX{\xpct@ymax}{\xpct@ytwo}{\xpct@ymax}
        \MAX{\xpct@ymax}{\xpct@ythree}{\xpct@ymax}
```

Width and height (`xmax-xmin` and `ymax-ymin`) of the sorrounding box.

```
        \SUBTRACT{\xpct@xmax}{\xpct@xmin}{\xpct@pictwidth}
        \SUBTRACT{\xpct@ymax}{\xpct@ymin}{\xpct@pictheight}
```

Call `picture`.

```
          \begin{picture}(\xpct@pictwidth,\xpct@pictheight)(%
             \xpct@xmin,\xpct@ymin)
```

Fix highest label to normal 1.

```
          \highestlabel{\normalfont\normalsize$1$}
```

If option `draft` was selected, background is colored, a surrounding rectangle is drawn and a centered label is printed.

```
          \ifdraft
            \colorlet{backgroundcolor}{lightgray}
            \xpct@backgrd
            \put(\xpct@xmin,\xpct@ymin){\line(1,0){\xpct@pictwidth}}
            \put(\xpct@xmin,\xpct@ymin){\line(0,1){\xpct@pictheight}}
            \put(\xpct@xmin,\xpct@ymax){\line(1,0){\xpct@pictwidth}}
            \put(\xpct@xmax,\xpct@ymin){\line(0,1){\xpct@pictheight}}
            \VECTORADD(\xpct@xmax,\xpct@ymax)(\xpct@xmin,\xpct@ymin)(%
            \xpct@xmed,\xpct@ymed)
            \SCALARVECTORPRODUCT{0.5}(\xpct@xmed,\xpct@ymed)(%
            \xpct@xmed,\xpct@ymed)
            \put(\xpct@xmed,\xpct@ymed){\makebox(0,0){\scshape xpicture}}
          \else
```

Finally, if required, we color the background.

```
            \ifbg\xpct@backgrd\fi
          \fi}
```

`\xpct@@Picture` Set background color to `#1`, swich boolean `\ifbg` to true and call `\xpct@Picture`.

```
\def\xpct@@Picture[#1](#2,#3)(#4,#5){%
          \colorlet{backgroundcolor}{#1}%
          \bgtrue\xpct@Picture(#2,#3)(#4,#5)}
```

`\xpct@backgrd` Fill background with `backgroundcolor`. We use `pict2e` path commands.

```
\def\xpct@backgrd{\begingroup
                  \pictcolor{backgroundcolor}
                  \moveto(\xpct@xzero,\xpct@yzero)
                  \lineto(\xpct@xthree,\xpct@ythree)
                  \lineto(\xpct@xone,\xpct@yone)
                  \lineto(\xpct@xtwo,\xpct@ytwo)
                  \closepath\fillpath
                \endgroup}
```

`\endPicture` Close `picture` environment.

```
\def\endPicture{\end{picture}}
```

`xpicture` `xpicture` is an alias for `Picture`.

```
\newenvironment{xpicture}{\begin{Picture}}{\end{Picture}}
```

4.11 \put extensions

User's commands are `\cPut`, `\rPut`, and `\Put`. `\rPut` and `\Put` have starred versions. Related commands are `\highestlabel` and `\defaultPut`.

`\cPut` `\cPut` puts the `#4` object in the `(#2,#3)` point at the `#1` position (circular version).

```
\def\cPut#1(#2,#3)#4{%
```

Select circular trigonometry and call \xpct@PUT.

```
\COPY{0}{\xpct@CorRput}
\xpct@PUT{#1}(#2,#3){#4}}
```

\rPut
\rPut*

\rPut puts the #4 object in the (#2,#3) point at the #1 position (rectangular version). Call \xpct@rPut (ordinary) or \xpct@rPutstar (starred).

```
\def\rPut{\@ifstar
             \xpct@rPutstar%
             \xpct@rPut%
             }
```

\Put
\Put*

\Put is equivalent to \cPut or \rPut, and has a starred form. Call \xpct@Put (ordinary) or \xpct@Putstar (starred).

```
\def\Put{\@ifstar
            \xpct@Putstar%
            \xpct@Put%
       }
```

\defaultPut
\xpct@defaultPut

\defaultPut is a declaration to fix default position (c or r) for the \Put command. It defines \xpct@defaultPut to be \rPut or \cPut.

```
\def\defaultPut#1{\def\xpct@tempa{#1}\def\xpct@tempb{r}
       \ifx\xpct@tempa\xpct@tempb
          \xpct@Infopos#1
          \def\xpct@defaultPut{\rPut}
       \else
          \xpct@Infopos#1
          \def\xpct@tempc{c}
          \ifx\xpct@tempa\xpct@tempc
             \def\xpct@defaultPut{\cPut}
          \else
             \xpct@Warnbadpos
       \fi\fi}
```

\highestlabel

The highest label for the starred \Put and \rPut commands. First we mesure the label, then we convert this length to \unitlength. This number is stored in \xpct@rputmxhg.

```
\def\highestlabel#1{\settoheight{\xpct@bxh}{#1}%
       \LENGTHDIVIDE{\xpct@bxh}{\unitlength}{\xpct@rputmxhg}}
```

Private commands. Main command is \xpct@PUT, all other commands are intended to select appropiate geometry.

\xpct@rPut
\xpct@rPutstar

We give the appropriate value to boolean \rputstar, select rectangular trigonometry and call \xpct@PUT.

```
\def\xpct@rPutstar{\rputstartrue\COPY{1}{\xpct@CorRput}\xpct@PUT}
\def\xpct@rPut{\rputstarfalse\COPY{1}{\xpct@CorRput}\xpct@PUT}
```

\xpct@Putstar
\xpct@Put

\Put can take an optional argument.

```
\def\xpct@Putstar{\@ifnextchar[{\xpct@@Putstar}{\xpct@@Put}}
\def\xpct@Put{\@ifnextchar[{\xpct@@@Put}{\xpct@@Put}}
```

\xpct@@Putstar
\xpct@@Put
\xpct@@@Put

\Put[pos] is \rPut*{pos} (for "bl" like pos), \cPut{pos} (for "SW" like pos) and \defaultPut pos otherwise.

\Put*[pos] is \rPut*{pos} or \cPut{pos} (only for "SW" like pos).

```
\def\xpct@@Put(#1){\refsysPoint(#1)(\xpct@abscoorx,\xpct@abscoory)
      \put(\xpct@abscoorx,\xpct@abscoory)}
\def\xpct@@Putstar[#1](#2)#3{\xpct@convtoang{#1}{\xpct@putpos}{\xpct@CorR}
      \if\xpct@CorR c
         \cPut{#1}(#2){#3}
      \else
         \rPut*{#1}(#2){#3}
      \fi}
\def\xpct@@@Put[#1](#2)#3{\xpct@convtoang{#1}{\xpct@putpos}{\xpct@CorR}
      \if\xpct@CorR c
         \cPut{#1}(#2){#3}
      \else
         \if\xpct@CorR r
            \rPut{#1}(#2){#3}
         \else
            \xpct@defaultPut{#1}(#2){#3}
      \fi\fi}
```

\xpct@PUT

This command puts object #4 in (#2,#3) (active reference), according to #1 position.

```
\def\xpct@PUT#1(#2,#3)#4{%
```

Call \xpct@alphaput to compute (\xpct@xPictsep,\xpct@yPictsep), displacement of \Pictlabelsep units in direction #1. Then, apply \refsysxyVector to get (\xpct@Posx,\xpct@Posy), standard coordinates of vector (\xpct@xPictsep,\xpct@yPictsep).

```
      \xpct@alphaput{#1}{\xpct@CorRput}
      \refsysxyVector(\xpct@xPictsep,\xpct@yPictsep)(\xpct@Posx,\xpct@Posy)
```

Compute (\xpct@posx,\xpct@posy), standard coordinates of point (#2,#3).

```
      \refsysPoint(#2,#3)(\xpct@posx,\xpct@posy)
```

Call \xpct@alphamove to adjust (\xpct@Posx,\xpct@Posy) according to dimensions of #4. Then add (\xpct@posx,\xpct@posy) to (\xpct@Posx,\xpct@Posy).

```
      \xpct@alphamove{#4}{\xpct@CorRput}
      \VECTORADD(\xpct@posx,\xpct@posy)(\xpct@Posx,\xpct@Posy)(%
         \xpct@Posx,\xpct@Posy)
```

Now (\xpct@Posx,\xpct@Posy) is the absolute position where #4 must go.

```
      \put(\xpct@Posx,\xpct@Posy){#4}}
```

\xpct@alphaput

Computes displacement vector required by #1 and stores it in (\xpct@xPictsep},\xpct@yPictsep).

```
\def\xpct@alphaput#1#2{\def\xpct@tempa{#1}\def\xpct@tempb{c}%
      \ifx\xpct@tempa\xpct@tempb
```

If #1=c, no displacement is required: (\xpct@xPictsep},\xpct@yPictsep)=(0,0).

```
         \COPY{0}{\xpct@xPictsep}\COPY{0}{\xpct@yPictsep}
      \else
```

Else, call \xpct@convtoang to translate #1 to a number (of degrees),

```
         \xpct@convtoang{#1}{\xpct@putpos}{\xpct@CorR}
```

and compute (\xpct@xPictsep},\xpct@yPictsep).

```
          \ifnum #2=0
             \DEGREESCOS{\xpct@putpos}{\xpct@cosine}
             \DEGREESSIN{\xpct@putpos}{\xpct@sine}
          \else
             \qCOS{\xpct@putpos}{\xpct@cosine}
             \qSIN{\xpct@putpos}{\xpct@sine}
          \fi
          \MULTIPLY{\Pictlabelsep}{\xpct@cosine}{\xpct@xPictsep}
          \MULTIPLY{\Pictlabelsep}{\xpct@sine}{\xpct@yPictsep}
       \fi}
```

\xpct@alphamove Adjust (\xpct@Posx,\xpct@Posy) to required position, according to #1 dimensions. If #2 equals 0, it uses circular trigonometry, else it uses square trigonometry.

```
\def\xpct@alphamove#1#2{%
```

Computes half of dimensions of #1,

```
       \xpct@halfbox{#1}{\xpct@amplada}{\xpct@altura}
       \ifx\xpct@tempa\xpct@tempb
       \else
```

If required position is not centered, move (\xpct@Posx,\xpct@Posy) (circular or square cases). First, compute a unitary vector in (\xpct@Posx,\xpct@Posy) direction.

```
          \ifnum #2=0
             \UNITVECTOR(\xpct@Posx,\xpct@Posy)(\xpct@xdir,\xpct@ydir)
          \else
             \qUNITVECTOR(\xpct@Posx,\xpct@Posy)(\xpct@xdir,\xpct@ydir)
```

If starred, change height to half \xpct@rputmxhg.

```
             \ifrputstar
                \ifdim\xpct@ydir\p@=-1\p@
                   \DIVIDE{\xpct@rputmxhg}{2}{\xpct@altura}
                \fi
             \fi
             \fi
```

Adjust (\xpct@xdir,\xpct@ydir) to #1 dimensions.

```
             \MULTIPLY{\xpct@ydir}{\xpct@altura}{\xpct@ydir}
             \MULTIPLY{\xpct@xdir}{\xpct@amplada}{\xpct@xdir}
             \VECTORADD(\xpct@Posx,\xpct@Posy)(\xpct@xdir,\xpct@ydir)%
                       (\xpct@Posx,\xpct@Posy)
       \fi
```

Move (\xpct@Posx,\xpct@Posy) according to #1 dimensions.

```
       \VECTORSUB(\xpct@Posx,\xpct@Posy)(\xpct@amplada,\xpct@altura)(%
          \xpct@Posx,\xpct@Posy)}
```

\xpct@convtoang Literal specifiers in \Put-like commands must be converted to angles. c or r (circular or rectangular) distance is also selected.

```
\def\xpct@convtoang#1#2#3{%
     \def\xpct@tempc{#1}
     \def\xpct@tempd{r}\ifx\xpct@tempc\xpct@tempd\COPY{0}{#2}\def#3{r}\else
```

```
\def\xpct@tempd{tr}\ifx\xpct@tempc\xpct@tempd\COPY{45}{#2}\def#3{r}\else
\def\xpct@tempd{t}\ifx\xpct@tempc\xpct@tempd\COPY{90}{#2}\def#3{r}\else
\def\xpct@tempd{tl}\ifx\xpct@tempc\xpct@tempd\COPY{135}{#2}\def#3{r}\else
\def\xpct@tempd{l}\ifx\xpct@tempc\xpct@tempd\COPY{180}{#2}\def#3{r}
\else
\def\xpct@tempd{bl}\ifx\xpct@tempc\xpct@tempd\COPY{-135}{#2}
\def#3{r}\else
\def\xpct@tempd{b}\ifx\xpct@tempc\xpct@tempd\COPY{-90}{#2}
\def#3{r}\else
\def\xpct@tempd{br}\ifx\xpct@tempc\xpct@tempd\COPY{-45}{#2}
\def#3{r}\else
\def\xpct@tempd{rtr}\ifx\xpct@tempc\xpct@tempd\COPY{22.5}{#2}
\def#3{r}\else
\def\xpct@tempd{ttr}\ifx\xpct@tempc\xpct@tempd\COPY{67.5}{#2}
\def#3{r}\else
\def\xpct@tempd{ttl}\ifx\xpct@tempc\xpct@tempd\COPY{112.5}{#2}
\def#3{r}\else
\def\xpct@tempd{ltl}\ifx\xpct@tempc\xpct@tempd\COPY{157.5}{#2}
\def#3{r}\else
\def\xpct@tempd{lbl}\ifx\xpct@tempc\xpct@tempd\COPY{-157.5}{#2}
\def#3{r}\else
\def\xpct@tempd{bbl}\ifx\xpct@tempc\xpct@tempd\COPY{-112.5}{#2}
\def#3{r}\else
\def\xpct@tempd{bbr}\ifx\xpct@tempc\xpct@tempd\COPY{-67.5}{#2}
\def#3{r}\else
\def\xpct@tempd{rbr}\ifx\xpct@tempc\xpct@tempd\COPY{-22.5}{#2}
\def#3{r}\else
\def\xpct@tempd{E}\ifx\xpct@tempc\xpct@tempd\COPY{0}{#2}\def#3{c}\else
\def\xpct@tempd{NE}\ifx\xpct@tempc\xpct@tempd\COPY{45}{#2}\def#3{c}\else
\def\xpct@tempd{N}\ifx\xpct@tempc\xpct@tempd\COPY{90}{#2}\def#3{c}\else
\def\xpct@tempd{NW}\ifx\xpct@tempc\xpct@tempd\COPY{135}{#2}\def#3{c}\else
\def\xpct@tempd{W}\ifx\xpct@tempc\xpct@tempd\COPY{180}{#2}\def#3{c}\else
\def\xpct@tempd{SW}\ifx\xpct@tempc\xpct@tempd\COPY{-135}{#2}
\def#3{c}\else
\def\xpct@tempd{S}\ifx\xpct@tempc\xpct@tempd\COPY{-90}{#2}\def#3{c}\else
\def\xpct@tempd{SE}\ifx\xpct@tempc\xpct@tempd\COPY{-45}{#2}\def#3{c}\else
\def\xpct@tempd{ENE}\ifx\xpct@tempc\xpct@tempd\COPY{22.5}{#2}
\def#3{c}\else
\def\xpct@tempd{NNE}\ifx\xpct@tempc\xpct@tempd\COPY{67.5}{#2}
\def#3{c}\else
\def\xpct@tempd{NNW}\ifx\xpct@tempc\xpct@tempd\COPY{112.5}{#2}
\def#3{c}\else
\def\xpct@tempd{WNW}\ifx\xpct@tempc\xpct@tempd\COPY{157.5}{#2}
\def#3{c}\else
\def\xpct@tempd{WSW}\ifx\xpct@tempc\xpct@tempd\COPY{-157.5}{#2}
\def#3{c}\else
\def\xpct@tempd{SSW}\ifx\xpct@tempc\xpct@tempd\COPY{-112.5}{#2}
\def#3{c}\else
\def\xpct@tempd{SSE}\ifx\xpct@tempc\xpct@tempd\COPY{-67.5}{#2}
\def#3{c}\else
```

```
      \def\xpct@tempd{ESE}\ifx\xpct@tempc\xpct@tempd\COPY{-22.5}{#2}
      \def#3{c}\else
      \def\xpct@tempd{c}\ifx\xpct@tempc\xpct@tempd\COPY{0}{#2}\def#3{c}\else
      \COPY{#1}{#2}\def#3{a}
\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
}
```

`\xpct@halfbox` Half of dimensions of a box.

```
\def\xpct@halfbox#1#2#3{%
        \settowidth\xpct@bxw{#1}%
        \settoheight\xpct@bxh{#1}%
        \LENGTHDIVIDE{\xpct@bxw}{\unitlength}{#2}
        \LENGTHDIVIDE{\xpct@bxh}{\unitlength}{#3}
        \MULTIPLY{0.5}{#2}{#2}
        \MULTIPLY{0.5}{#3}{#3}}
```

`\qCOS` `\qSIN` `\qUNITVECTOR` Square versions of `\DEGREESCOS`, `\DEGREESSIN` and `\UNITVECTOR`.

```
\def\qCOS#1#2{%
    \ifdim #1\p@<-135\p@
         \ADD{360}{#1}{\xpct@angles}\qCOS{\xpct@angles}{#2}
    \else
       \ifdim #1\p@>225\p@ \SUBTRACT{#1}{360}{\xpct@angles}
                           \qCOS{\xpct@angles}{#2}
       \else
          \ifdim #1\p@<-45\p@ \DEGREESCOT{#1}{#2}\MULTIPLY{-1}{#2}{#2}
          \else
             \ifdim #1\p@<45\p@ \COPY{1}{#2}
             \else
                \ifdim #1\p@<135\p@ \DEGREESCOT{#1}{#2}
                \else
                   \COPY{-1}{#2}
                \fi
             \fi
          \fi
       \fi
    \fi
}
\def\qSIN#1#2{%
    \ifdim #1\p@<-135\p@
         \ADD{360}{#1}{\xpct@angles}\qSIN{\xpct@angles}{#2}
    \else
       \ifdim #1\p@>225\p@ \SUBTRACT{#1}{360}{\xpct@angles}
                           \qSIN{\xpct@angles}{#2}
       \else
          \ifdim #1\p@<-45\p@ \COPY{-1}{#2}
          \else
             \ifdim #1\p@<45\p@ \DEGREESTAN{#1}{#2}
             \else
```

```
                \ifdim #1\p@<135\p@ \COPY{1}{#2}
                \else
                   \DEGREESTAN{#1}{#2}\MULTIPLY{-1}{#2}{#2}
                \fi
             \fi
          \fi
       \fi
    \fi
}
\def\qUNITVECTOR(#1,#2)(#3,#4){%
          \VECTORCOPY(#1,#2)(#3,#4)
          \ABSVALUE{#4}{\xpct@Ydir}
          \ifdim \xpct@Ydir\p@ < 0.00005\p@
          \COPY{\xpct@maxnum}{\xpct@tan}
          \else
             \DIVIDE{#3}{#4}{\xpct@tan}
          \fi
          \ifdim #3\p@ > 0\p@
             \ifdim #4\p@ > 0\p@
                \ifdim #3\p@ > #4\p@
                   \COPY{1}{#3}\DIVIDE{#4}{\xpct@tan}{#4}
                \else
                   \COPY{1}{#4}\COPY{\xpct@tan}{#3}
                \fi
                \else
                   \ifdim #3\p@ > -#4\p@
                      \COPY{1}{#3}\DIVIDE{-#4}{\xpct@tan}{#4}
                   \else
                      \COPY{-1}{#4}\MULTIPLY{-1}{\xpct@tan}{#3}
                   \fi
                \fi
             \else
                \ifdim #4\p@ > 0\p@
                   \ifdim -#3\p@ > #4\p@
                      \COPY{-1}{#3}\DIVIDE{-#4}{\xpct@tan}{#4}
                   \else
                      \COPY{1}{#4}\COPY{\xpct@tan}{#3}
                   \fi
                \else
                   \ifdim #3\p@ > #4\p@
                      \COPY{-1}{#4}\COPY{-\xpct@tan}{#3}
                   \else
                      \COPY{-1}{#3}\DIVIDE{#4}{\xpct@tan}{#4}
                   \fi
                \fi
             \fi
}
```

4.12 \multiput extensions

User commands: \multicPut, \multirPut, \multiPut; \multicPlot, \multirPlot, \multiPlot. \multirPut, \multiPut, \multirPlot, and \multiPlot have starred versions.

\multicPut Define \xpct@mPut as \cPut{#1} and call \xpct@@mPut.

```
\def\multicPut#1{\def\xpct@mPut{\cPut{#1}}\xpct@@mPut}
```

\multirPut \multirPut* Call \xpct@multirPut or \xpct@multirPutstar (if starred).

```
\def\multirPut{\@ifstar
        \xpct@multirPutstar%
        \xpct@multirPut%
        }
```

\multiPut \multiPut* Call \xpct@multiPut or \xpct@multiPutstar (if starred).

```
\def\multiPut{\@ifstar
        \xpct@multiPutstar%
        \xpct@multiPut%
        }
```

\xpct@multirPutstar \xpct@multirPut Define \xpct@mPut as \rPut*{#1} or \rPut{#1} and call \xpct@@mPut.

```
\def\xpct@multirPutstar#1{\def\xpct@mPut{\rPut*{#1}}\xpct@@mPut}
\def\xpct@multirPut#1{\def\xpct@mPut{\rPut{#1}}\xpct@@mPut}
```

\xpct@multiPut \xpct@multiPutstar \multiPut can take an optional argument.

```
\def\xpct@multiPut{\@ifnextchar[{\xpct@@@multiPut}{\xpct@@multiPut}}
\def\xpct@multiPutstar{\@ifnextchar[{\xpct@@@multiPutstar}{\xpct@@multiPutstar}}
```

\xpct@@multiPut \xpct@@multiPutstar Define \xpct@mPut as \Put or \Put* and call \xpct@@mPut.

```
\def\xpct@@multiPut{\def\xpct@mPut{\Put}\xpct@@mPut}
\def\xpct@@multiPutstar{\def\xpct@mPut{\Put*}\xpct@@mPut}
```

\xpct@@@multiPut \xpct@@@multiPutstar Define \xpct@mPut as \Put[#1] or \Put*[#1] and call \xpct@@mPut.

```
\def\xpct@@@multiPut[#1]{\def\xpct@mPut{\Put[#1]}\xpct@@mPut}
\def\xpct@@@multiPutstar[#1]{\def\xpct@mPut{\Put*[#1]}\xpct@@mPut}
```

\xpct@@mPut \xpct@mPut \xpct@@mPut is the main macro about \multiPut-like commands. \xpct@mPut is already defined as the appropiate \Put command.

```
\def\xpct@@mPut(#1,#2)(#3,#4)#5#6{%
```

Use counter multiput to count iterations. (\xpct@@abscoorx,\xpct@@abscoory) is the point to be ploted in each iteration.

```
        \COPY{#1}\xpct@@abscoorx\COPY{#2}\xpct@@abscoory
        \setcounter{multiput}{0}%
        \@whilenum\value{multiput}<#5 \do
```

Plot the point, translate it, and update conter.

```
            {\xpct@mPut(\xpct@@abscoorx,\xpct@@abscoory){#6}
             \ADD{#3}\xpct@@abscoorx\xpct@@abscoorx
             \ADD{#4}\xpct@@abscoory\xpct@@abscoory
             \stepcounter{multiput}}}
```

\multicPlot Execute \cPut and iterates itself while next character be (.

```
\def\multicPlot#1#2(#3){\cPut{#1}(#3){#2}\@ifnextchar({\multicPlot{#1}{#2}}{}}
```

\multirPlot \multirPlot can take a starred form. Call \xpct@multirPlot or, if starred, \xpct@multirPlotstar.
\multirPlot*

```
\def\multirPlot{\@ifstar
        \xpct@multirPlotstar%
        \xpct@multirPlot%
        }
```

\multiPlot \multiPlot can take a starred form. Call \xpct@multiPlot or, if starred, \xpct@multiPlotstar.
\multiPlot*

```
\def\multiPlot{\@ifstar
             \xpct@multiPlotstar%
             \xpct@multiPlot%
             }
```

\xpct@multirPlotstar Execute \rPut* or \rPut and iterates itself while next character be (.
\xpct@multirPlot

```
\def\xpct@multirPlotstar#1#2(#3){\rPut*{#1}(#3){#2}
        \@ifnextchar({\xpct@multirPlotstar{#1}{#2}}{}}
\def\xpct@multirPlot#1#2(#3){\rPut{#1}(#3){#2}
        \@ifnextchar({\xpct@multirPlot{#1}{#2}}{}}
```

\xpct@multiPlotstar \multiPlot (and \multiPlot*) can take an optional argument. We have four cases: (starred
\xpct@multiPlot or not) and (optional argument or not).

```
\def\xpct@multiPlotstar{%
   \@ifnextchar[{\xpct@@@multiPlotstar}{\xpct@@multiPlotstar}}
\def\xpct@multiPlot{\@ifnextchar[{\xpct@@@multiPlot}{\xpct@@multiPlot}}
```

\xpct@@multiPlot Execute \Put (or \Put*) and iterates itself while next character be (.
\xpct@@@multiPlot

```
\def\xpct@@multiPlot#1(#2){\Put(#2){#1}\@ifnextchar({\xpct@@multiPlot{#1}}{}}
\def\xpct@@@multiPlot[#1]#2(#3){\Put[#1](#3){#2}
        \@ifnextchar({\xpct@@@multiPlot[#1]{#2}}{}}
\def\xpct@@multiPlotstar#1(#2){\Put*(#2){#1}
        \@ifnextchar({\xpct@@multiPlotstar{#1}}{}}
\def\xpct@@@multiPlotstar[#1]#2(#3){\Put*[#1](#3){#2}
        \@ifnextchar({\xpct@@@multiPlotstar[#1]{#2}}{}}
```

4.13 Strigth lines and vectors

\xLINE Compute standard coordinates of two points and call \xpct@strline to plot a line.
\strline

```
\def\xLINE(#1)(#2){%
         \refsysPoint(#1)(\xpct@xzero,\xpct@yzero)
         \refsysPoint(#2)(\xpct@xone,\xpct@yone)
         \xpct@strline(\xpct@xzero,\xpct@yzero)(\xpct@xone,\xpct@yone)}
\let\strline\xLINE
```

\xpct@strline This command calls the \segment command from curve2e (or \LINE, for old versions of curve2e).

```
\def\xpct@strline{\@killglue\@ifundefined{segment}{\LINE}{\segment}}
```

\xVECTOR Compute standard coordinates of two points and call \VECTOR to plot a vector.

```
\def\xVECTOR(#1)(#2){%
        \refsysPoint(#1)(\xpct@xzero,\xpct@yzero)
        \refsysPoint(#2)(\xpct@xone,\xpct@yone)
        \VECTOR(\xpct@xzero,\xpct@yzero)(\xpct@xone,\xpct@yone)}
```

\xtrivVECTOR Compute standard coordinates of two points and call \xpct@xtrivVECTOR to plot a 'triv' vector.

```
\def\xtrivVECTOR(#1)(#2){%
        \refsysPoint(#1)(\xpct@xzeropoint,\xpct@yzeropoint)
        \refsysPoint(#2)(\xpct@xonepoint,\xpct@yonepoint)
        \xpct@xtrivVECTOR(\xpct@xzeropoint,\xpct@yzeropoint)(%
                          \xpct@xonepoint,\xpct@yonepoint)}
```

\arrowsize Store dimensions of triv arrows. to plot a vector.

```
\def\arrowsize#1#2{\COPY{#1}{\xpct@xarrowlen}
                   \COPY{#2}{\xpct@yarrowlen}}
```

\xpct@xtrivVECTOR Plot a stright line, compute size of arrowhead and call \xpct@arrow to plot it.

```
\def\xpct@xtrivVECTOR(#1)(#2){%
          \xpct@strline(#1)(#2)
          \VECTORSUB(#2)(#1)(\xpct@xarrow,\xpct@yarrow)
          \VECTORNORM(\xpct@xarrow,\xpct@yarrow){\xpct@xarrowunit}
          \DIVIDE{\xpct@xarrow}{\xpct@xarrowunit}{\xpct@xarrow}
          \DIVIDE{\xpct@yarrow}{\xpct@xarrowunit}{\xpct@yarrow}
          \xpct@arrow(#2){\xpct@xarrow}{\xpct@yarrow}}
```

\xpct@arrow Make an arrowhead as a small picture.

```
\def\xpct@arrow(#1)#2#3{\begingroup%
              \referencesystem(#1)(#2,#3)(-#3,#2)
              \Put(0,0){\setlength{\unitlength}{1pt}%
                 \begin{Picture}(0,0)(0,0)\cartesianreference
                          \xLINE(-\xpct@xarrowlen,\xpct@yarrowlen)(0,0)
                          \xLINE(0,0)(-\xpct@xarrowlen,-\xpct@yarrowlen)
                 \end{Picture}}\endgroup}
```

\zerovector To have an arrowhead, draw a very short vector (of 0.01\unitlength).
\zerotrivvector

```
\def\zerovector(#1){%
   \UNITVECTOR(#1)(\xpct@dirx,\xpct@diry)
   \SCALARVECTORPRODUCT{0.01}(\xpct@dirx,\xpct@diry)(\xpct@dirx,\xpct@diry)
    \xVECTOR(0,0)(\xpct@dirx,\xpct@diry)}
\def\zerotrivvector(#1){%
   \UNITVECTOR(#1)(\xpct@dirx,\xpct@diry)
   \SCALARVECTORPRODUCT{0.01}(\xpct@dirx,\xpct@diry)(\xpct@dirx,\xpct@diry)
    \xtrivVECTOR(0,0)(\xpct@dirx,\xpct@diry)}
```

\xline Standard syntax strigth lines and vectors. Call \xpct@xline to compute adequate coordinates
\xvector of line or vector ends. Then call \xLINE, \xVECTOR or \xtrivVECTOR command.
\xtrivvector

```
\def\xline(#1,#2)#3{%
        \xpct@xline(#1,#2){#3}
```

```
   \xLINE(0,0)(\xpct@@xdir,\xpct@@ydir)}

\def\xvector(#1,#2)#3{%
   \ifdim #3 pt = 0 pt \zerovector(#1,#2)
   \else
       \xpct@xline(#1,#2){#3}
   \xVECTOR(0,0)(\xpct@@xdir,\xpct@@ydir)
   \fi}

\def\xtrivvector(#1,#2)#3{%
   \ifdim #3 pt = 0 pt \zerotrivvector(#1,#2)
   \else
       \xpct@xline(#1,#2){#3}
   \xtrivVECTOR(0,0)(\xpct@@xdir,\xpct@@ydir)
   \fi}
```

`\xpct@xline` Calculate the coordinates of the endpoint of `\xline(#1,#2){#3}` and stores them in `(\xpct@@xdir,\xpct@@y`

```
\def\xpct@xline(#1,#2)#3{%
         \ABSVALUE{#1}{\xpct@modx}
         \ifdim \xpct@modx pt < 0.0001 pt
              \COPY{0}{\xpct@@xdir}
              \ifdim #2\p@>\z@ \COPY{#3}{\xpct@@ydir}
              \else \MULTIPLY{-1}{#3}{\xpct@@ydir}
              \fi
         \else
              \DIVIDE{#1}{\xpct@modx}{\xpct@@xdir}
              \DIVIDE{#2}{\xpct@modx}{\xpct@@ydir}
              \SCALARVECTORPRODUCT{#3}(\xpct@@xdir,\xpct@@ydir)(%
                                      \xpct@@xdir,\xpct@@ydir)
        \fi}
```

4.14 Polygons and polylines

`\Polyline` This command plots a line between the two first points and, if next character is `(`, supresses first point and iterates itself.

```
\def\Polyline(#1)(#2){%
         \xLINE(#1)(#2)\@ifnextchar({\Polyline(#2)}{}}
```

`\Polygon` Store the first point in `(\xpct@firstx,\xpct@firsty)` and call `\xpct@Polygon`.

```
\def\Polygon(#1,#2)(#3){%
       \COPY{#1}{\xpct@firstx}\COPY{#2}{\xpct@firsty}
                  \xpct@Polygon(#1,#2)(#3)}
```

`\xpct@Polygon` This command plots a line between the two first points and, if next character is `(`, supresses first point and iterates itself. When finished, adds a closing line to the previously stored first point.

```
\def\xpct@Polygon(#1)(#2){%
       \xLINE(#1)(#2)\@ifnextchar({\xpct@Polygon(#2)}{%
       \xLINE(#2)(\xpct@firstx,\xpct@firsty)}}
```

\regularPolygon `\regularPolygon` can take an optional argument.

```
\def\regularPolygon{%
      \@ifnextchar[{\xpct@regPolygon}{\xpct@@regPolygon}}
```

\xpct@@regPolygon Default for optional argument is `0`.

```
\def\xpct@@regPolygon#1#2{\xpct@regPolygon[0]{#1}{#2}}
```

\xpct@regPolygon `\xpct@regPolygon[#1]{#2}{#3}` uses the `xpct@counta` counter to plot `#3` lines, in polar coordinates with `#2` radius, startint with angle `#1` and using 360/`#3` to incrementing angle in each step.

```
\def\xpct@regPolygon[#1]#2#3{\begingroup%
      \polarreference\degreesangles
      \setcounter{xpct@counta}{0}%
      \setcounter{xpct@countb}{#3}%
      \DIVIDE{360}{#3}{\xpct@angles}
      \COPY{#1}{\xpct@anglea}
      \@whilenum\value{xpct@counta}<\value{xpct@countb} \do {%
         \ADD{\xpct@anglea}{\xpct@angles}{\xpct@angleb}
         \xLINE(#2,\xpct@anglea)(#2,\xpct@angleb)
         \COPY{\xpct@angleb}{\xpct@anglea}\stepcounter{xpct@counta}}
      \endgroup}
```

4.15 Quadratic curves

\xpct@ctrlpoint The main command in this section is `\xpct@ctrlpoint`. It computes the control point in a quadratic Bezier curve from the coordinates and direction vectors of ending points.

```
\def\xpct@ctrlpoint(#1,#2)(#3,#4)(#5,#6)(#7,#8){%
      \DETERMINANT(#3,#4;#7,#8)\xpct@detA
      \DETERMINANT(#1,#2;#3,#4)\xpct@detB
      \DETERMINANT(#5,#6;#7,#8)\xpct@detC
      \DETERMINANT(#3,#7;\xpct@detB,\xpct@detC)\xpct@detD
      \DETERMINANT(#4,#8;\xpct@detB,\xpct@detC)\xpct@detE
      \ABSVALUE{\xpct@detA}{\xpct@@detA}
      \ABSVALUE{\xpct@detD}{\xpct@@detD}
      \ABSVALUE{\xpct@detE}{\xpct@@detE}
      \ifdim \xpct@@detA pt<0.00005 pt
```

If `\xpct@detA` approaches zero, matrix is singular or close to singular. Then tangent lines may be parallel or coincide.

```
         \ifdim \xpct@@detD pt<0.00005 pt %\xpct@detD pt=0 pt
            \ifdim \xpct@@detE pt<0.00005 pt %\xpct@detE pt=0 pt
```

Indeterminate system. The curve is a straight line. We take (as reference point) middle point between end points.

```
               \ADD{#1}{#5}{\xpct@solx}\DIVIDE{\xpct@solx}{2}{\xpct@solx}
               \ADD{#2}{#6}{\xpct@soly}\DIVIDE{\xpct@soly}{2}{\xpct@soly}
            \fi\else
```

Inconsistent case. Return a warning and undefine control point.

```
            \xpct@WarnIncSys(#1,#2)(#5,#6)
```

```
            \let\xpct@solx\undefined\let\xpct@soly\undefined
         \fi
      \else
```

This is the regular case.

```
         \DIVIDE{\xpct@detD}{\xpct@detA}{\xpct@solx}
         \DIVIDE{\xpct@detE}{\xpct@detA}{\xpct@soly}
      \fi}
```

`\qCurve` This macro accepts two alternative syntax (directions given by a vector or by an angle).

```
\def\qCurve(#1){\@ifnextchar({\xpct@@qCurve(#1)}{\xpct@@@qCurve(#1)}}
```

`\xpct@@qCurve` Compute standard coordinates of points and vectors and call `\xpct@qCurve`.

```
\def\xpct@@qCurve(#1)(#2)(#3)(#4){%
      \refsysPoint(#1)(\xpct@@xzero,\xpct@@yzero)
      \refsysPoint(#3)(\xpct@@xone,\xpct@@yone)
      \refsysVector(#2)(\xpct@@dxzero,\xpct@@dyzero)
      \refsysVector(#4)(\xpct@@dxone,\xpct@@dyone)
      \xpct@qCurve(\xpct@@xzero,\xpct@@yzero)(\xpct@@dxzero,\xpct@@dyzero)(%
                  \xpct@@xone,\xpct@@yone)%
         (\xpct@@dxone,\xpct@@dyone)}
```

`\xpct@@@qCurve` Translate direction angles to vectors and call `\qCurve`.

```
\def\xpct@@@qCurve(#1)#2(#3)#4{%
            \ifpolar
                \qCurve(#1)(1,#2)(#3)(1,#4)
            \else
            \DEGREESCOS{#2}{\xpct@angxz}
            \DEGREESSIN{#2}{\xpct@angyz}
            \DEGREESCOS{#4}{\xpct@angxo}
            \DEGREESSIN{#4}{\xpct@angyo}
            \qCurve(#1)(\xpct@angxz,\xpct@angyz)(#3)%
               (\xpct@angxo,\xpct@angyo)\fi}
```

`\xpct@qCurve` Call `\xpct@ctrlpoint` to compute control point; then, use `\qbezier` to plot the curve. If the control point is undefined, nothing is drawn.

```
\def\xpct@qCurve(#1)(#2)(#3)(#4){%
      \xpct@ctrlpoint(#1)(#2)(#3)(#4)
      \ifx\xpct@solx\undefined
      \else
         \qbezier(#1)(\xpct@solx,\xpct@soly)(#3)\fi\ignorespaces}
```

`\PlotQuadraticCurve` Try between the two alternative sintax.

```
\def\PlotQuadraticCurve(#1){%
      \@ifnextchar({\xpct@PlotQuadraticCurve(#1)}{%
                     \xpct@@PlotQuadraticCurve(#1)}}
```

`\xpct@PlotQuadraticCurve` `\xpct@@PlotQuadraticCurve` Call `\qCurve` and iterate `\PlotQuadraticCurve`.

```
\def\xpct@PlotQuadraticCurve(#1)(#2)(#3)(#4){%
      \qCurve(#1)(#2)(#3)(#4)
```

```
        \@ifnextchar({\PlotQuadraticCurve(#3)(#4)}{}}
\def\xpct@@PlotQuadraticCurve(#1)#2(#3)#4{%
        \qCurve(#1){#2}(#3){#4}
        \@ifnextchar({\PlotQuadraticCurve(#3){#4}}{}}
```

4.16 Conic sections and arcs

\xpct@circulararc
\xpct@hyperbolicarc
\xpct@parabolicarc

Parametric equations of circular, hyperbolic and parabolic arcs defined as vector functions.

```
\newvectorfunction{\xpct@circulararc}{%
```

Unit circle equation $x^2 + y^2 = 1$ can be parameterized as $f(t) = (\cos t, \sin t)$. If the angles are measured in degrees, the derivative is not correct. Should be multiplied by $\pi/180$, but because what we want is the direction of the derivative, we will not do.

```
    \ifdegrees
       \DEGREESCOS{\t}{\x}
       \DEGREESSIN{\t}{\y}
       \COPY{\x}{\Dy}
       \MULTIPLY{-1}{\y}{\Dx}
    \else
       \COS{\t}{\x}
       \SIN{\t}{\y}
       \COPY{\x}{\Dy}
       \MULTIPLY{-1}{\y}{\Dx}
    \fi}
\newvectorfunction{\xpct@hyperbolicarc}{%
```

Hyperbola $x^2 - y^2 = 1$, parameterized as $f(t) = \frac{1}{2}(t+1/t, t-1/t)$. This derivative is not correct. We should divide it by t, but that did not change direction.

```
       \DIVIDE{1}{\t}{\xpct@invt}
       \ADD{\t}{\xpct@invt}{\x}
       \SUBTRACT{\t}{\xpct@invt}{\y}
       \MULTIPLY{0.5}{\x}{\x}
       \MULTIPLY{0.5}{\y}{\y}
       \COPY{\x}{\Dy}
       \COPY{\y}{\Dx}}
```

Parabola $x = y^2$ (or $f(t) = (t^2, t)$).

```
\newvectorfunction{\xpct@parabolicarc}{%
        \COPY{\t}{\y}
        \COPY{1}{\Dy}
        \SQUARE{\t}{\x}
        \MULTIPLY{2}{\t}{\Dx}}
```

\circularArc
\xArc

A circular arc is an elliptic arc with equal semiaxes.

```
\def\circularArc#1#2#3{\ellipticArc{#1}{#1}{#2}{#3}}
\let\xArc\circularArc
```

\ellipticArc

To draw an arc of ellipse of semiaxes `#1` and `#2`, scale the axes and draw a circular arc. `\defaultplotdivs` is the number of subintervals we divide `[#3,#4]`.

```
\def\ellipticArc#1#2#3#4{%
```

```
    \begingroup
      \cartesianreference
       \changereferencesystem(0,0)(#1,0)(0,#2)
       \PlotParametricFunction[\defaultplotdivs]{\xpct@circulararc}{#3}{#4}
    \endgroup\ignorespaces}
```

`\Circle` A circle (or ellipse) is a circular (elliptic) arc of amplitude 2π.

`\Ellipse`

```
\def\Circle#1{\begingroup\radiansangles
                      \circularArc{#1}{0}{\numberTWOPI}\endgroup\ignorespaces}
\def\Ellipse#1#2{\begingroup\radiansangles
                          \ellipticArc{#1}{#2}{0}{\numberTWOPI}
                          \endgroup\ignorespaces}
```

`\lhyperbolicArc` Change x-axis to $-x$, then draw a right hyperbolic arc.

```
\def\lhyperbolicArc#1#2#3#4{%
            \begingroup
               \changereferencesystem(0,0)(-1,0)(0,1)
               \rhyperbolicArc{#1}{#2}{#3}{#4}
           \endgroup}
```

`\rhyperbolicArc` Call `\xpct@hypluy` to compute extreme variables, then draw a normalized arc of hyperbola.

```
\def\rhyperbolicArc#1#2#3#4{%
              \xpct@hypluy{#2}{#3}{\xpct@uone}
              \xpct@hypluy{#2}{#4}{\xpct@utwo}
              \xpct@hyperbolicArc{#1}{#2}{\xpct@uone}{\xpct@utwo}}
```

`\xpct@hyperbolicArc` To draw an arc of (right branch of) hyperbola of semiaxes `#1` and `#2`, scale the axes and draw a normalized arc of hyperbola. `\defaultplotdivs` is the number of subintervals we divide `[#3,#4]`.

```
\def\xpct@hyperbolicArc#1#2#3#4{%
    \begingroup
      \cartesianreference
      \changereferencesystem(0,0)(#1,0)(0,#2)
       \PlotParametricFunction[\defaultplotdivs]{\xpct@hyperbolicarc}{#3}{#4}
    \endgroup}
```

`\lHyperbola` Change x-axis to $-x$, then draw a right hyperbola branch.

```
\def\lHyperbola#1#2#3#4{%
          \begingroup
          \changereferencesystem(0,0)(-1,0)(0,1)
          \rHyperbola{#1}{#2}{#3}{#4}
          \endgroup}
```

`\rHyperbola` Use `\xpct@hypconsist` to ensure parameters consistency, call `\xpct@hyperbolalastu` to compute extreme variable, then plot the right hyperbola branch. Divide the curve into two arcs to ensure that it includes point `(#1,0)`.

```
\def\rHyperbola#1#2#3#4{%
       \def\xpct@hycons{}\xpct@hypconsist{#1}{#3}%
          \ifx\xpct@hycons\undefined
```

```
          \else
              \xpct@hyperbolalastu{#1}{#2}{#3}{#4}
              \DIVIDE{1}{\xpct@umax}{\xpct@umin}
              \xpct@hyperbolicArc{#1}{#2}{\xpct@umin}{1}
               \xpct@hyperbolicArc{#1}{#2}{1}{\xpct@umax}
          \fi}
```

`\Hyperbola` Use `\xpct@hypconsist` to ensure parameters consistency, call `\xpct@hyperbolalastu` to compute extreme variable, then plot the two branches.

```
\def\Hyperbola#1#2#3#4{%
       \begingroup
       \def\xpct@hycons{}\xpct@hypconsist{#1}{#3}%
          \ifx\xpct@hycons\undefined
          \else
              \xpct@hyperbolalastu{#1}{#2}{#3}{#4}
              \DIVIDE{1}{\xpct@umax}{\xpct@umin}
               \xpct@hyperbolicArc{#1}{#2}{\xpct@umin}{1}
               \xpct@hyperbolicArc{#1}{#2}{1}{\xpct@umax}
             \changereferencesystem(0,0)(-1,0)(0,1)
              \xpct@hyperbolicArc{#1}{#2}{\xpct@umin}{1}
               \xpct@hyperbolicArc{#1}{#2}{1}{\xpct@umax}
          \fi\endgroup}
```

`\xpct@hypconsist` Ensures consistency of parameters in `\Hyperbola`-like commands. This curve is not defined for $x < a$ values.

```
\def\xpct@hypconsist#1#2{%
       \ifnum #1<#2\else\xpct@ErrHypCons
                           \let\xpct@hycons\undefined\fi}
```

`\xpct@hyperbolalastu` Compute the max value of parameter ensuring restrictions `x<#3` and `y<#4`.

```
\def\xpct@hyperbolalastu#1#2#3#4{%
      \xpct@hyplux{#1}{#3}{\xpct@umaxx}
      \xpct@hypluy{#2}{#4}{\xpct@umaxy}
       \MIN{\xpct@umaxx}{\xpct@umaxy}{\xpct@umax}}
```

`\xpct@hyplux` To compute the max value of parameter ensuring restriction `x<=#2`, solve equation `#2=(1/2)#1(u+1/u)` (`u=#3`).

```
\def \xpct@hyplux#1#2#3{%
       \DIVIDE{#2}{#1}{\xpct@xa}
       \SQUARE{\xpct@xa}{#3}
       \SUBTRACT{#3}{1}{#3}
       \SQUAREROOT{#3}{\xpct@@umaxx}
       \ADD{\xpct@xa}{\xpct@@umaxx}{#3}}
```

`\xpct@hypluy` To compute the max value of parameter ensuring restriction `y<#2`, solve equation `#2=(1/2)#1(u-1/u)` (`u=#3`).

```
\def \xpct@hypluy#1#2#3{%
       \DIVIDE{#2}{#1}{\xpct@xa}
       \SQUARE{\xpct@xa}{#3}
```

```
        \ADD{#3}{1}{#3}
        \SQUAREROOT{#3}{\xpct@@umaxx}
        \ADD{\xpct@xa}{\xpct@@umaxx}{#3}}
```

`\parabolicArc` To draw an arc of parabola scale the x-axis and draw a normalized arc of parabola. `\defaultplotdivs` is the number of subintervals we divide `[#2,#3]`.

```
\def\parabolicArc#1#2#3{%
    \begingroup
       \changereferencesystem(0,0)(#1,0)(0,1)
       \PlotParametricFunction[\defaultplotdivs]{\xpct@parabolicarc}{#2}{#3}
    \endgroup}
```

`\Parabola` Call `\xpct@parabolalasty` to compute extreme variable, then plot the parabola. Divide the curve into two arcs to ensure that it includes point `(0,0)`.

```
\def\Parabola#1#2#3{%
              \xpct@parabolalasty{#1}{#2}{#3}
              \parabolicArc{#1}{-\xpct@maxy}{0}
              \parabolicArc{#1}{0}{\xpct@maxy}}
```

`\xpct@parabolalasty` Ensure restrictions `x<=#2`, `y<=#3`: solve equation `#2=#1 y^2`. Then, `\xpct@maxy=min(y,#3)`.

```
\def\xpct@parabolalasty#1#2#3{%
       \ABSVALUE{#1}{\xpct@@maxy}
       \DIVIDE{#2}{\xpct@@maxy}{\xpct@@maxy}
       \SQUAREROOT{\xpct@@maxy}{\xpct@maxy}
       \MIN{\xpct@maxy}{#3}{\xpct@maxy}}
```

4.17 Graphing functions

`\PlotFunction` This command can take an optional argument.

```
\def\PlotFunction{%
       \@ifnextchar[{\xpct@iterateplotfunction}{\xpct@plotfunction}}
```

`\xpct@iterateplotfunction` Compute `\xpct@step` as `(#4-#3)/#1` and iterate `\xpct@plotfunction #1` times.

```
\def\xpct@iterateplotfunction[#1]#2#3#4{%
\setcounter{xpct@step}{0}%
\COPY{#3}{\xpct@oldt}
\SUBTRACT{#4}{#3}{\xpct@step}
\DIVIDE{\xpct@step}{#1}{\xpct@step}
\@whilenum \value{xpct@step}<#1 \do
   {\ADD{\xpct@oldt}{\xpct@step}{\xpct@newt}
   \xpct@plotfunction{#2}{\xpct@oldt}{\xpct@newt}
   \stepcounter{xpct@step}%
   \COPY\xpct@newt\xpct@oldt
}}
```

`\xpct@plotfunction` Draw graph of `#1` function between `#2` and `#3`.

```
\def\xpct@plotfunction#1#2#3{\@killglue%
```

Compute f and f' in #2 and #3, and apply \PlotxyDyData.

```
        #1{#2}{\yzero}{\Dyzero}%
        #1{#3}{\yone}{\Dyone}%
        \PlotxyDyData(#2,\yzero,\Dyzero)(#3,\yone,\Dyone)
        \ifx\xpct@solx\undefined
```

If tangent vectors are parallel, divide the interval into two halves and recall \xpct@plotfunction.

```
        \ADD{#2}{#3}{\xpct@middt}
        \MULTIPLY{0.5}{\xpct@middt}{\xpct@middt}
        \xpct@plotfunction{#1}{#2}{\xpct@middt}
        \xpct@plotfunction{#1}{\xpct@middt}{#3}
        \fi}
```

\PlotPointsOfFunction The \PlotPointsOfFunction command is essentialy equal to \xpct@iterateplotfunction, but instead of a curve between two adjacent points, plots a \pointmark (user can redefine \pointmark).

```
\def\PlotPointsOfFunction#1#2#3#4{%
   \setcounter{xpct@step}{0}%
\COPY{#3}{\xpct@oldt}
\SUBTRACT{#4}{#3}{\xpct@step}
\DIVIDE{\xpct@step}{#1}{\xpct@step}
\ADD{#1}{1}{\xpct@lastt}
\@whilenum \value{xpct@step}<\xpct@lastt \do
   {\ADD{\xpct@oldt}{\xpct@step}{\xpct@newt}
   #2{\xpct@oldt}{\xpct@oldy}{\xpct@oldDy}
   \Put[c](\xpct@oldt,\xpct@oldy){\pointmark}
   \stepcounter{xpct@step}%
   \COPY\xpct@newt\xpct@oldt
}}
```

\PlotxyDyData \PlotxyDyData(x0,y0,y0')(x1,y1,y1')(x2,y2,y2')... uses \qCurve to draw a curve between (x0,y0) and (x1,y1) with tangent vectors (1,y0') and (1,y1'), then iterates itself.

```
\def\PlotxyDyData(#1,#2,#3)(#4,#5,#6){%
        \qCurve(#1,#2)(1,#3)(#4,#5)(1,#6)
        \@ifnextchar({\PlotxyDyData(#4,#5,#6)}{}}
```

4.18 Graphing parametric curves

\PlotParametricFunction Plot vectorial function #2 between the parameter values #3 and #4. It can take an optional argument #1.

```
\def\PlotParametricFunction{%
    \@ifnextchar[{\xpct@iterateplotpfunction}{\xpct@plotpfunction}}
```

\xpct@iterateplotpfunction Divide [#3,#4] in #1 pieces, then iterate \xpct@plotpfunction #1 times.

```
\def\xpct@iterateplotpfunction[#1]#2#3#4{%
\setcounter{xpct@step}{0}%
\COPY{#3}{\xpct@oldt}
\SUBTRACT{#4}{#3}{\xpct@step}
\DIVIDE{\xpct@step}{#1}{\xpct@step}
```

```
\@whilenum \value{xpct@step}<#1 \do
   {\ADD{\xpct@oldt}{\xpct@step}{\xpct@newt}
   \xpct@plotpfunction{#2}{\xpct@oldt}{\xpct@newt}
   \stepcounter{xpct@step}%
   \COPY\xpct@newt\xpct@oldt}\ignorespaces}
```

\xpct@plotpfunction Compute function (and derivative of) #1 in #2 and #3, then call \qCurve.

```
\def\xpct@plotpfunction#1#2#3{%
    \begingroup
       #1{#2}\xzero\Dxzero\yzero\Dyzero
       #1{#3}\xone\Dxone\yone\Dyone
       \cartesianreference
       \qCurve(\xzero,\yzero)(\Dxzero,\Dyzero)(\xone,\yone)(\Dxone,\Dyone)
    \endgroup\ignorespaces}
```

4.19 Cartesian axes and grids

Main commands: \cartesianaxes and \cartesiangrid.

\cartesiangrid Put \ifgrid to true, then call \cartesianaxes.

```
\def\cartesiangrid(#1,#2)(#3,#4){%
       \begingroup\gridtrue\cartesianaxes(#1,#2)(#3,#4)\endgroup}
```

\cartesianaxes \xpct@XZero \xpct@XOne \xpct@YZero \xpct@YOne \cartesianaxes makes axes and, optionally, grid, tics, and/or labels. Cartesian rectangle limits are stored in \xpct@XZero, \xpct@XOne, \xpct@YZero, and \xpct@YOne.

```
\def\cartesianaxes(#1,#2)(#3,#4){%
```

In this command, coordinates are Cartesian.

```
       \begingroup\cartesianreference
          \GLOBALCOPY{#1}{\xpct@XZero}\GLOBALCOPY{#2}{\xpct@YZero}
          \GLOBALCOPY{#3}{\xpct@XOne}\GLOBALCOPY{#4}{\xpct@YOne}
```

There shall be cuts, labels or grid?

```
          \iftics
             \ticslabelsgridtrue
          \else
             \iflabels
                \ticslabelsgridtrue
             \else
                \ifgrid
                   \ticslabelsgridtrue
          \fi\fi\fi
          \ifticslabelsgrid
             \xpct@plotticslabels
          \fi
```

Call \xpct@plotaxes to plot axes.

```
          \xpct@plotaxes\endgroup}
```

\plotxtic \plotytic Put \iftics boolean to true, adjust tics lengths and position, and call \xpct@printtic.

```
\def\plotxtic#1{%
```

```
          \maketics
          \xpct@adjticssize
          \xpct@adjxorytics{#1}{0}
          \xpct@printtic}
\def\plotytic#1{%
          \maketics
          \xpct@adjticssize
          \xpct@adjxorytics{#1}{1}
          \xpct@printtic}
```

`\printxlabel` `\printylabel` Adjust tics lengths and position, and call `\xpct@printlabel`.

```
\def\printxlabel#1#2{%
          \xpct@adjticssize
          \xpct@adjxorytics{#1}{0}
          \xpct@printlabel{0}{#2}}
\def\printylabel#1#2{%
          \xpct@adjticssize
          \xpct@adjxorytics{#1}{1}
          \xpct@printlabel{1}{#2}}
```

`\printxticlabel` `\printyticlabel` Print tic and label.

```
\def\printxticlabel#1#2{\plotxtic{#1}\printxlabel{#1}{#2}}
\def\printyticlabel#1#2{\plotytic{#1}\printylabel{#1}{#2}}
```

`\plotxtics` `\plotytics` Call `\xpct@plottics{0}}` or `\xpct@plottics{1}`.

```
\def\plotxtics{\xpct@plottics{0}}
\def\plotytics{\xpct@plottics{1}}
```

`\printxlabels` `\printylabels` Call `\xpct@printlabels{0}` or `\xpct@printlabels{1}`. By default, optional argument must be `-1`.

```
\def\printxlabels{%
   \@ifnextchar[{\xpct@printlabels{0}}{\xpct@printlabels{0}[-1]}}
\def\printylabels{%
   \@ifnextchar[{\xpct@printlabels{1}}{\xpct@printlabels{1}[-1]}}
```

`\printxticslabels` `\printyticslabels` Call `\xpct@printxticslabels` or `\xpct@printyticslabels`. By default, optional argument must be `-1`.

```
\def\printxticslabels{%
   \@ifnextchar[{\xpct@printxticslabels}{\xpct@printxticslabels[-1]}}
\def\printyticslabels{%
   \@ifnextchar[{\xpct@printyticslabels}{\xpct@printyticslabels[-1]}}
```

`\xpct@plotaxes` Axes are simple lines, but its position depends on boolean `\inzeroaxes`.

```
\def\xpct@plotaxes{\linethickness{\axesthickness}%
                \pictcolor{\axescolor}
                \ifinzeroaxes
                   \xLINE(\xpct@XZero,0)(\xpct@XOne,0)
                   \xLINE(0,\xpct@YZero)(0,\xpct@YOne)
                \else
```

```
                \xLINE(\xpct@XZero,\xpct@YZero)(\xpct@XOne,\xpct@YZero)
                \xLINE(\xpct@XZero,\xpct@YZero)(\xpct@XZero,\xpct@YOne)
            \fi}
```

`\xpct@plotticslabels` Adjust tics sizes to axes lengths and call `\xpct@plotxticslabels` and `\xpct@plotyticslabels`.

```
\def\xpct@plotticslabels{%
    \xpct@adjticssize
    \xpct@plotxticslabels\xpct@plotyticslabels}
```

`\xpct@plotxticslabels` Grid, tics and labels on the `x` axis. If secundary divisions are required, this command iterates itself.

```
\def\xpct@plotxticslabels{%
```

If required, plot grid (in both directions, `x` and `y`).

```
    \ifgrid\xpct@plotgrid\fi
    \begingroup
      \ifnum\xunitdivisions=1
```

Call `\xpct@ticsinterval` to compute integer interval extremes and number of tics; then plot `x` tics.

```
          \xpct@ticsinterval{\xpct@XZero}{\xpct@XOne}
          \xpct@plotxtics
      \else
```

Secundary tics.

```
          \begingroup
```

Secundary tics. Change the reference system to the small unities, and ajust tics sizes, thickness and colors.

```
            \xpct@adjstics
            \MULTIPLY{\secundaryyticssize}{\yunitdivisions}{\yticssize}
```

At secundary level one must not print labels.

```
            \makenolabels
```

Print secundary tics.

```
            \def\xunitdivisions{1}
            \xpct@plotxticslabels
          \endgroup
```

Print primary tics and (perhaps) labels.

```
          \def\xunitdivisions{1}
          \xpct@plotxticslabels
      \fi
    \endgroup}
```

`\xpct@plotyticslabels` Tics and labels on the `y` axis. If secundary divisions are required, this command iterates itself.

```
\def\xpct@plotyticslabels{%
    \begingroup
      \ifnum\yunitdivisions=1
```

Call `\xpct@ticsinterval` to compute integer interval extremes and number of tics; then plot `y` tics.

```
            \xpct@ticsinterval{\xpct@YZero}{\xpct@YOne}
            \xpct@plotytics
        \else
```

Secundary tics.

```
            \begingroup
```

Secundary tics. Change the reference system to the small unities, and ajust tics sizes, thickness and colors.

```
                \xpct@adjstics
                \MULTIPLY{\secundaryxticssize}{\xunitdivisions}{\xticssize}
```

At secundary level one must not print labels.

```
                \makenolabels
```

Print secundary tics.

```
                \def\yunitdivisions{1}
                \xpct@plotyticslabels
            \endgroup
```

Print primary tics and (perhaps) labels.

```
            \def\yunitdivisions{1}
            \xpct@plotyticslabels
        \fi
    \endgroup}
```

`\xpct@adjstics` Adjust length, color and thickness for secundary tics.

```
\def\xpct@adjstics{%
                \MULTIPLY{\xpct@XZero}{\xunitdivisions}{\xpct@XZero}
                \MULTIPLY{\xpct@YZero}{\yunitdivisions}{\xpct@YZero}
                \MULTIPLY{\xpct@XOne}{\xunitdivisions}{\xpct@XOne}
                \MULTIPLY{\xpct@YOne}{\yunitdivisions}{\xpct@YOne}
                \DIVIDE{1}{\xunitdivisions}{\xpct@xunit}
                \DIVIDE{1}{\yunitdivisions}{\xpct@yunit}
                \changereferencesystem(0,0)(\xpct@xunit,0)(0,\xpct@yunit)
                \def\gridthickness{\secundarygridthickness}
                \def\gridcolor{\secundarygridcolor}}
```

`\xpct@plotxtics` `\xpct@plotytics` Call `\xpct@maketics` to make tics and/or labels (on `x` and `y` axes).

```
\def\xpct@plotxtics{\xpct@maketics{\xpct@firstint}{\xpct@numtics}{0}}
\def\xpct@plotytics{\xpct@maketics{\xpct@firstint}{\xpct@numtics}{1}}
```

`\xpct@maketics` Makes tics and/or labels (`#2` points, begining in `#1`; `#3=0` means `x` axis, `#3=1` means `y` axis).

```
\def\xpct@maketics#1#2#3{%
```

Call `\xpct@adjxorytics` to compute coordinates of extreme points of first tic and translation vector from one tic to the next one.

```
    \xpct@adjxorytics{#1}{#3}
```

Use counter `xpct@counta` for tics and `xpct@countb` for labels (number to print in each label).

```
      \setcounter{xpct@counta}{0}%
      \iflabels\setcounter{xpct@countb}{#1}\fi
```

Main loop: `#2` steps, begining in `#1`.

```
      \@whilenum \value{xpct@counta}<#2 \do {%
         \iftics
```

If required, print tic.

```
           \xpct@printtic
           \fi
           \iflabels
```

If labels are to be printed, adjust `\Pictlabelsep`; then print label and step label (`xpct@countb` counter).

```
              \highestlabel{\xpct@axeslabelattrib%
                            $\axeslabelmathalphabet{1}$}%
              \xpct@printlabel{#3}{\thexpct@countb}
              \stepcounter{xpct@countb}%
           \fi
```

Step tics counter and move coordinates to next point.

```
           \stepcounter{xpct@counta}%
           \VECTORADD(\xpct@@xzero,\xpct@@yzero)(\xpct@@xincr,\xpct@@yincr)%
           (\xpct@@xzero,\xpct@@yzero)
           \VECTORADD(\xpct@@xone,\xpct@@yone)(\xpct@@xincr,\xpct@@yincr)%
           (\xpct@@xone,\xpct@@yone)
      }}
```

`\xpct@adjxorytics` Compute coordinates of extreme points of first tic and translation vector from one tic to the next one. There are four cases: `x` or `y` axis, and external or internal axes.

```
\def\xpct@adjxorytics#1#2{%
     \ifnum #2=0
               \COPY{#1}{\xpct@@xzero}
               \COPY{-\yticssize}{\xpct@@yzero}
               \COPY{#1}{\xpct@@xone}
               \COPY{\yticssize}{\xpct@@yone}
               \COPY{1}{\xpct@@xincr}
               \COPY{0}{\xpct@@yincr}
               \ifinzeroaxes\else
                  \ADD{\xpct@YZero}{\xpct@@yzero}{\xpct@@yzero}
                  \ADD{\xpct@YZero}{\xpct@@yone}{\xpct@@yone}
               \fi
          \else
               \COPY{#1}{\xpct@@yzero}
               \COPY{-\xticssize}{\xpct@@xzero}
               \COPY{#1}{\xpct@@yone}
               \COPY{\xticssize}{\xpct@@xone}
               \COPY{1}{\xpct@@yincr}
               \COPY{0}{\xpct@@xincr}
               \ifinzeroaxes\else
```

```
                  \ADD{\xpct@XZero}{\xpct@@xzero}{\xpct@@xzero}
                  \ADD{\xpct@XZero}{\xpct@@xone}{\xpct@@xone}
               \fi
      \fi}
```

`\xpct@printtic` Plot a tic.

```
\def\xpct@printtic{\pictcolor{\ticscolor}
            \linethickness{\ticsthickness}
            \xLINE(\xpct@@xzero,\xpct@@yzero)(\xpct@@xone,\xpct@@yone)}
```

`\xpct@adjticssize` Adjust size of tics according to axes length.

```
\def\xpct@adjticssize{%
```

First, convert absolute lenghts `\ticssize` and `\secundaryticssize` to the `\unitlength` unity.

```
      \LENGTHDIVIDE{\ticssize}{\unitlength}{\xpct@ticssize}
      \LENGTHDIVIDE{\secundaryticssize}{\unitlength}{\xpct@sticssize}
```

Calculate the size of vector `(1,0)`, converting it to standard coordinates and computing its norm. Then we adjust `\xticssize` and `\secundaryxticssize`; this ensures the desired sizes.

```
      \refsysxyVector(1,0)(\xpct@a,\xpct@b)
         \VECTORNORM(\xpct@a,\xpct@b){\xpct@norm}
         \DIVIDE{\xpct@ticssize}{\xpct@norm}{\xticssize}
         \DIVIDE{\xpct@sticssize}{\xpct@norm}{\secundaryxticssize}
         \DIVIDE{\axislabelsep}{\xpct@norm}{\xpct@xaxislabelsep}
```

Repeat calculations for vector $(0, 1)$, adjusting `\yticssize` and `\secundaryyticssize`.

```
      \refsysxyVector(0,1)(\xpct@a,\xpct@b)
         \VECTORNORM(\xpct@a,\xpct@b){\xpct@norm}
         \DIVIDE{\xpct@ticssize}{\xpct@norm}{\yticssize}
         \DIVIDE{\xpct@sticssize}{\xpct@norm}{\secundaryyticssize}
         \DIVIDE{\axislabelsep}{\xpct@norm}{\xpct@yaxislabelsep}}
```

`\xpct@printlabel` Adjust highest label (for horizontal labels); then print value of `#2`. Four cases: `x` or `y`, external or internal.

```
\def\xpct@printlabel#1#2{%
      \iftics
         \ifnum #1=0
            \ADD{\yticssize}{\xpct@yaxislabelsep}{\Pictlabelsep}
         \else
            \ADD{\xticssize}{\xpct@xaxislabelsep}{\Pictlabelsep}
         \fi
      \else
         \ifnum #1=0
            \COPY{\xpct@yaxislabelsep}{\Pictlabelsep}
         \else
            \COPY{\xpct@xaxislabelsep}{\Pictlabelsep}
         \fi
      \fi
   \ifinzeroaxes
      \ifnum\thexpct@countb=0
      \else
```

```
961          \ifnum #1=0
962             \rPut*{\xpct@xlblpos}(\xpct@@xzero,0){%
963                \xpct@axeslabelattrib%
964                \ensuremath{\axeslabelmathalphabet{#2}}}
965          \else
966             \rPut*{\xpct@ylblpos}(0,\xpct@@yzero){%
967                \xpct@axeslabelattrib%
968                \ensuremath{\axeslabelmathalphabet{#2}}}
969          \fi
970       \fi
971    \else
972       \ifnum #1=0
973          \rPut*{\xpct@xlblpos}(\xpct@@xzero,\xpct@YZero){%
974             \xpct@axeslabelattrib%
975             \ensuremath{\axeslabelmathalphabet{#2}}}
976       \else
977          \rPut*{\xpct@ylblpos}(\xpct@XZero,\xpct@@yzero){%
978             \xpct@axeslabelattrib%
979             \ensuremath{\axeslabelmathalphabet{#2}}}
980       \fi
981    \fi}
```

`\xlabelpos` `\ylabelpos` Default position for labels on axes. Call `\xpct@convtoang` to define `\xpct@xlblpos` or `\xpct@ylblpos`.

```
982 \def\xlabelpos#1{\xpct@convtoang{#1}{\xpct@xlblpos}{\xpct@CorR}}
983 \def\ylabelpos#1{\xpct@convtoang{#1}{\xpct@ylblpos}{\xpct@CorR}}
```

`\xpct@printxticslabels` `\xpct@printyticslabels` Print tics and labels.

```
984 \def\xpct@printxticslabels[#1]#2#3#4{%
985       \plotxtics{#2}{#3}{#4}\printxlabels[#1]{#2}{#3}{#4}}
986 \def\xpct@printyticslabels[#1]#2#3#4{%
987       \plotytics{#2}{#3}{#4}\printylabels[#1]{#2}{#3}{#4}}
```

`\xpct@plottics` Plot `x` or `y` tics (if `#1` equals `0` or `1`), starting an `#2`. Distance between two consecutive tics is `#3`, and position of last tic is not greather than `#4`.

```
988 \def\xpct@plottics#1#2#3#4{%
989      \COPY{#2}{\xpct@ticcoor}
```

`\xpct@ticcoor` is the position of next tic.

```
990      \@whiledim\xpct@ticcoor\p@<#4\p@ \do {%
```

Make a tic while `\xpct@ticcoor<#4`

```
991      \ifnum #1=0
992         \plotxtic{\xpct@ticcoor}
993      \else
994         \plotytic{\xpct@ticcoor}
995      \fi
996      \ADD{#3}{\xpct@ticcoor}{\xpct@ticcoor}
997      }
```

If \xpct@ticcoor=#4 then this is the last tic position.

```
      \ifdim\xpct@ticcoor\p@>#4\p@
      \else
         \ifnum #1=0
            \plotxtic{\xpct@ticcoor}
         \else
            \plotytic{\xpct@ticcoor}
         \fi
      \fi}
```

\xpct@printlabels Print x or y labels (if #1 equals 0 or 1), starting an #3. Distance between two consecutive tics is #4, and position of last tic is not greather than #5. #2 is the number of decimal digits to be printed (default is #2=-1, meaning no control of digits in number printing).

```
\def\xpct@printlabels#1[#2]#3#4#5{%
      \COPY{#3}{\xpct@ticcoor}
```

\xpct@ticcoor is the position of next label.

```
      \@whiledim\xpct@ticcoor\p@<#5\p@ \do {%
```

Print a label while \xpct@ticcoor<#5 \xpct@Ticcoor is the label with adjusted number of digits.

```
         \ifnum #2=-1
            \COPY{\xpct@ticcoor}{\xpct@Ticcoor}
         \else
            \ROUND[#2]{\xpct@ticcoor}{\xpct@Ticcoor}
         \fi
         \xpct@prtlbl{#1}
         \ADD{#4}{\xpct@ticcoor}{\xpct@ticcoor}}
      \ifdim\xpct@ticcoor\p@>#5\p@
```

If \xpct@ticcoor=#5 then this is the last label position.

```
      \else
         \ifnum #2=-1
            \COPY{\xpct@ticcoor}{\xpct@Ticcoor}
         \else
            \ROUND[#2]{\xpct@ticcoor}{\xpct@Ticcoor}
         \fi
         \xpct@prtlbl{#1}
      \fi}
```

\xpct@prtlbl Print the x or y label (for #1=0 or 1) \xpct@Ticcoor at \xpct@ticcoor. When \ifinzeroaxes is true label at 0 position is not printed.

```
\def\xpct@prtlbl#1{%
        \ifinzeroaxes
        \ifdim \xpct@ticcoor\p@=\z@\else
            \xpct@adjticssize
            \xpct@adjxorytics{\xpct@ticcoor}{#1}
            \xpct@printlabel{#1}{\xpct@Ticcoor}
         \fi
         \else
            \xpct@adjticssize
```

```
            \xpct@adjxorytics{\xpct@ticcoor}{#1}
            \xpct@printlabel{#1}{\xpct@Ticcoor}\fi}
```

`\xpct@plotgrid` Plot a grid in a Cartesian rectangle.

```
\def\xpct@plotgrid{%
```

Call `\xpct@ticsinterval` to compute integer interval extremes and number of tics; then plot grid lines (for `x` axis).

```
      \xpct@ticsinterval{\xpct@XZero}{\xpct@XOne}
      \begingroup\setcounter{xpct@counta}{0}%
         \pictcolor{\gridcolor}\linethickness{\gridthickness}
         \COPY{\xpct@firstint}{\xpct@grid}
         \@whilenum\value{xpct@counta}<\xpct@numtics\do{
            \xLINE(\xpct@grid,\xpct@YZero)(\xpct@grid,\xpct@YOne)
            \ADD{1}{\xpct@grid}{\xpct@grid}
            \stepcounter{xpct@counta}}\endgroup
```

Call `\xpct@ticsinterval` to compute integer interval extremes and number of tics; then plot grid lines (for `y` axis).

```
      \xpct@ticsinterval{\xpct@YZero}{\xpct@YOne}
      \begingroup\setcounter{xpct@counta}{0}%
         \pictcolor{\gridcolor}\linethickness{\gridthickness}
         \COPY{\xpct@firstint}{\xpct@grid}
         \@whilenum\value{xpct@counta}<\xpct@numtics\do{
            \xLINE(\xpct@XZero,\xpct@grid)(\xpct@XOne,\xpct@grid)
            \ADD{1}{\xpct@grid}{\xpct@grid}
            \stepcounter{xpct@counta}}\endgroup}
```

`\xpct@ticsinterval` Truncate extremes to integers, then compute the number of tics (`\xpct@firstint-\xpct@lastint+1`).

```
\def\xpct@ticsinterval#1#2{\TRUNCATE[0]{#1}{\xpct@firstint}
                          \TRUNCATE[0]{#2}{\xpct@lastint}
                          \SUBTRACT{\xpct@lastint}{\xpct@firstint}{\xpct@numtics}
                          \ADD{\xpct@numtics}{1}{\xpct@numtics}}
```

4.20 Polar grids

`\polargrid` Plot a polar grid of radius `#1` and `#2` divisions of circle.

```
\def\polargrid#1#2{%
      \begingroup
      \polarreference
```

Compute integer part of radius, number of circles and distance between circles.

```
      \FLOOR{#1}{\xpct@rint}
      \MULTIPLY{\xpct@rint}{\runitdivisions}{\xpct@rdivs}
      \DIVIDE{1}{\runitdivisions}{\rincr}
```

Use counter `xpct@counta` to control the number of printed circles and `\xpct@radius` as radius of the current circle.

```
      \COPY{0}{\xpct@radius}
      \setcounter{xpct@counta}{1}%
```

Plot `\xpct@rdivs` circles.

```
      \begingroup
         \pictcolor{\gridcolor}
         \linethickness{\gridthickness}
         \@whilenum \value{xpct@counta}<\xpct@rdivs\do {%
            \ADD{\rincr}{\xpct@radius}{\xpct@radius}
            \Ellipse{\xpct@radius}{\xpct@radius}
            \stepcounter{xpct@counta}}%
```

Plot external circle.

```
         \pictcolor{\axescolor}
         \linethickness{\axesthickness}
         \Ellipse{\xpct@rint}{\xpct@rint}
      \endgroup
```

Use counter `xpct@counta` to control the number of printed lines and `\xpct@angle` as arc (in radians) of the current line. `\xpct@angincr` is the gap between two adjacent lines.

```
      \COPY{0}{\xpct@angle}
      \DIVIDE{\numberTWOPI}{#2}{\xpct@angincr}
      \setcounter{xpct@counta}{0}%
```

Plot `#2` lines.

```
      \pictcolor{\gridcolor}
      \linethickness{\gridthickness}
      \@whilenum \value{xpct@counta}<#2 \do {%
         \xLINE(0,0)(#1,\xpct@angle)
```

If required, print angular label: evaluate the number `\xpct@arc` such that angle is `(\xpct@arc/#2) pi` and call `\xpct@polarlabel`.

```
         \iflabels
            \COPY{\axislabelsep}{\Pictlabelsep}
            \MULTIPLY{2}{\thexpct@counta}{\xpct@arc}
            \xpct@polarlabel{#1}{\xpct@arc}{#2}\fi
         \ADD{\xpct@angincr}{\xpct@angle}{\xpct@angle}
         \stepcounter{xpct@counta}}%
```

Plot the polar line.

```
       \pictcolor{\axescolor}
       \linethickness{\axesthickness}
       \xLINE(0,0)(#1,0)
```

If required, print radial labels.

```
      \iflabels
         \highestlabel{\xpct@axeslabelattrib$\axeslabelmathalphabet{1}$}
         \multiPut*[\xpct@rlblpos](1,0)(1,0){\xpct@rint}{%
            \ADD{\value{multiput}}{1}{\xpct@lbl}
            \xpct@axeslabelattrib%
            \ensuremath{\axeslabelmathalphabet{\xpct@lbl}}}%
      \fi
      \endgroup}
```

`\rlabelpos` Default position for labels on polar axis. Call `\xpct@convtoang` to define `\xpct@rlblpos`.

```
\def\rlabelpos#1{\xpct@convtoang{#1}{\xpct@rlblpos}{\xpct@CorR}}
```

\degreespolarlabels Define \xpct@polarlabel to be \xpct@degreeslabel or \xpct@radianslabel (print polar
\radianspolarlabels label as degrees or radians).

```
\def\degreespolarlabels{\def\xpct@polarlabel{\xpct@degreeslabel}}
\def\radianspolarlabels{\def\xpct@polarlabel{\xpct@radianslabel}}
```

\xpct@degreeslabel Print the angle label (#2/#3) pi converted to degrees.

```
\def\xpct@degreeslabel#1#2#3{%
```

Adjust label position.

Simplify #2/#3. Then convert (#2/#3) pi to degrees (evaluate (#2 180)/#3).

```
    \FRACTIONSIMPLIFY{#2}{#3}\xpct@num\xpct@den
    \MULTIPLY{\xpct@num}{180}{\xpct@degangle}
    \DIVIDE{\xpct@degangle}{\xpct@den}{\xpct@degangle}
```

Print label.

```
    \cPut{\xpct@degangle}(#1,\xpct@angle){%
       \xpct@axeslabelattrib%
       \ensuremath{\axeslabelmathalphabet{\xpct@degangle^\mathrm{o}}}}}
```

\xpct@radianslabel Print the angle label (#2/#3) pi.

```
\def\xpct@radianslabel#1#2#3{%
```

Adjust label position and call \xpct@prtfracrad.

```
        \RADtoDEG{\xpct@angle}{\xpct@angles}
        \cPut{\xpct@angles}(#1,\xpct@angle){%
           \xpct@axeslabelattrib%
           \ensuremath{\axeslabelmathalphabet
               {\xpct@prtfracrad{#2}{#3}}}}}
```

\xpct@prtfracrad Pretty print (#1/#2)pi

```
\def\xpct@prtfracrad#1#2{%
    \FRACTIONSIMPLIFY{#1}{#2}\xpct@num\xpct@den
       \ifnum \xpct@num = 0 0
       \else
          \ifnum \xpct@num = 1
             \ifnum \xpct@den = 1 \pi
             \else \pi/\xpct@den
             \fi
          \else \xpct@num\pi/\xpct@den
          \fi
       \fi}
```

4.21 Configurable parameters

These are the parameters the user can customize. Default values are written to xpicture.sty and xpicture.cfgxmpl.

⟨/xpicture⟩
⟨*defaults⟩
⟨+cfg⟩%%

```
⟨+cfg⟩%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
⟨+cfg⟩% xpicture configurable parameters %
⟨+cfg⟩%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
⟨+cfg⟩%%
⟨+cfg⟩%%%%% Cartesian and polar axes
⟨+cfg⟩    % Thickness and color of axes
\axesthickness=1pt
\def\axescolor{black}
⟨+cfg⟩    % Color, size, mathversion and mathalphabet of numeric labels
\def\axeslabelcolor{\axescolor}
\def\axeslabelsize{\small}
\def\axeslabelmathversion{normal}
\def\axeslabelmathalphabet{\mathrm}
⟨+cfg⟩    % Relative position of numeric labels on x- y- and r- axes
\xlabelpos{-90}
\ylabelpos{180}
\rlabelpos{bbr}
⟨+cfg⟩    % Distance between tags and cut marks,
⟨+cfg⟩    % is is a number (not a lenght) of \unitlength units
\def\axislabelsep{0.1}
⟨+cfg⟩    % Color, thickness and size of tics
\def\ticscolor{\axescolor}
\ticsthickness=1pt
\ticssize=4pt
⟨+cfg⟩    % Size of secundary tics
\secundaryticssize=2pt
⟨+cfg⟩    % Thickness and color of Cartesian or polar grid
\gridthickness=0.4pt
\def\gridcolor{gray}
⟨+cfg⟩    % Thickness and color of Cartesian or polar secundary grid
\secundarygridthickness=0.2pt
\def\secundarygridcolor{lightgray}
⟨+cfg⟩    % Number of divisions of unity in x- y- and r-axis
\def\xunitdivisions{1}
\def\yunitdivisions{1}
\def\runitdivisions{1}
⟨+cfg⟩    % Arc labels in radians (\xpct@radianslabel)
⟨+cfg⟩    %            or degrees (\xpct@degreeslabel)
\def\xpct@polarlabel{\xpct@radianslabel}
⟨+cfg⟩%%%%% \put and \multiput extensions
⟨+cfg⟩    % Distance from label to reference point,
⟨+cfg⟩    % is is a number (not a lenght) of \unitlength units
\def\Pictlabelsep{0.1}
⟨+cfg⟩    % Default layout for distance (\defaultPut{c} or \defaultPut{r})
\defaultPut{c}
⟨+cfg⟩%%%%% Reference systems
⟨+cfg⟩    % Default reference system
\referencesystem(0,0)(1,0)(0,1)
⟨+cfg⟩    % Cartesian or polar reference
\cartesianreference
```

```
⟨+cfg⟩%%%%% Arrow size in \xtrivVECTOR
\arrowsize{5}{2}
⟨+cfg⟩%%%%% Default interval divisions
⟨+cfg⟩    % (used when plotting conic sections and arcs)
\def\defaultplotdivs{8}
⟨+cfg⟩%%%%% Size to be used by \pointmark
\def\pointmarkdiam{0.1}
⟨+cfg⟩%%%%% Point mark used by \PlotPointsOfFunction
\def\pointmark{\circle*{\pointmarkdiam}}
⟨/defaults⟩
⟨*xpicture⟩
```

4.22 Commands to be ignored if draft option or \draftPicture declaration is active

\draftPictures This declaration allow user to locally disable Picture drawns.

```
\def\draftPictures{%
      \drafttrue
      \def\cPut##1(##2,##3)##4{}
      \def\xpct@@Put(##1)##2{}
      \def\xpct@@Putstar[##1](##2)##3{}
      \def\xpct@@@Put[##1](##2)##3{}
      \def\defaultPut##1{\def\xpct@defaultPut{\cPut}}
      \def\xpct@@mPut(##1,##2)(##3,##4)##5##6{}
      \def\xpct@PUT##1(##2,##3)##4{}
      \def\xLINE(##1)(##2){}
      \def\xtrivVECTOR(##1)(##2){}
      \def\xVECTOR(##1)(##2){}
      \def\zerovector(##1){}
      \def\zerotrivvector(##1){}
      \def\xline(##1,##2)##3{}
      \def\xvector(##1,##2)##3{}
      \def\xtrivvector(##1,##2)##3{}
      \def\xpct@regPolygon[##1]##2##3{}
      \def\xpct@@qCurve(##1)(##2)(##3)(##4){}
      \def\xpct@PlotQuadraticCurve(##1)(##2)(##3)(##4){%
          \@ifnextchar({\PlotQuadraticCurve(##3)(##4)}{}}
      \def\xpct@@PlotQuadraticCurve(##1)##2(##3)##4{%
          \@ifnextchar({\PlotQuadraticCurve(##3){##4}}{}}
      \def\circularArc##1##2##3{}
      \def\ellipticArc##1##2##3##4{}
      \def\Ellipse##1##2{}
      \def\Circle##1{}
      \def\xpct@hyperbolicArc##1##2##3##4{}
      \def\lHyperbola##1##2##3##4{}
      \def\rHyperbola##1##2##3##4{}
      \def\Hyperbola##1##2##3##4{}
      \def\rhyperbolicArc##1##2##3##4{}
      \def\lhyperbolicArc##1##2##3##4{}
      \def\parabolicArc##1##2##3{}
```

```
        \def\Parabola##1##2##3{}
        \def\PlotPointsOfFunction##1##2##3##4{}
        \def\xpct@iterateplotfunction[##1]##2##3##4{}
        \def\xpct@plotfunction##1##2##3{}
        \def\xpct@iterateplotpfunction[##1]##2##3##4{}
        \def\xpct@plotpfunction##1##2##3{}
        \def\cartesianaxes(##1,##2)(##3,##4){}
        \def\cartesiangrid(##1,##2)(##3,##4){}
        \def\plotxtic##1{}
        \def\plotytic##1{}
        \def\printxlabel##1##2{}
        \def\printylabel##1##2{}
        \def\printxticlabel##1##2{}
        \def\printyticlabel##1##2{}
        \def\plotxtics##1##2##3{}
        \def\plotytics##1##2##3{}
        \def\xpct@printlabels##1[##2]##3##4##5{}
        \def\polargrid##1##2{}
}
```

`\ifdraft` If `draft` option is active `\draftPictures` is executed. Then all `Picture` commands are disabled.

```
\ifdraft
    \draftPictures
\fi
```

Input local defaults (file `xpicture.cfg`).

```
\InputIfFileExists{xpicture.cfg}{\xpct@Infocfg}{\xpct@Infonocfg}
```

⟨/xpicture⟩

5 Change history

v1.2a (2012/11/17)

Documented source.

Many internal c.s. renamed and/or rewrited.

dvi/pict2e/curve2e options supressed.

draft option added.

Background color added to Picture environment.

`\Pictlabelsep` is set to `\normalfont\normalsize$1$` when a `Picture` environment starts.

New commands: `\draftPictures`,`\symmetrize`, `\xlabelpos`, `\ylabelpos`, `\plotxtic`, `\plotytic`, `\plotxtics`, `\plotytics`, `\printxlabel`, `\printylabel`, `\printxlabels`, `\printylabels`, `\printxticlabel`, `\printyticlabel`, `\printxticslabels`, `\printyticslabels`, `\makegrid`, `\makenogrid`, `\PlotPointsOfFunction`, `\pointmark`, `\pointmarkdiam`.

v1.2 (2012/04/25)

First public version.

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

A

\arrowsize *9*, 474, 1181
\axescolor . *5*, 826, 1072, 1089, 1137, 1139, 1151
\axeslabelcolor . *6*, 63, 1139
\axeslabelmathalphabet *6*, 894, 964, 968, 975, 979, 1093, 1097, 1109, 1114, 1142
\axeslabelmathversion *6*, 64, 1141
\axeslabelsize .. *6*, 62, 1140
\axesthickness *5*, 61, 825, 1073, 1090, 1136
\axislabelsep *6*, 938, 943, 1084, 1149

B

\bgfalse 18
\bgtrue 164

C

\cartesianaxes *5*, 777, 778, 1231
\cartesiangrid . *5*, 776, 1232
\cartesianreference *4*, 111, 486, 634, 654, 773, 779, 1179
\changereferencesystem *4*, 72, 85, 89, 97, 635, 645, 655, 660, 681, 706, 879
\Circle *9*, 638, 1217
\circularArc *10*, 630, 639, 1214
\cPut *7*, 175, 194, 208, 214, 411, 436, 1107, 1112, 1193, 1197

D

\defaultplotdivs *10*, 636, 656, 707, 1184
\defaultPut *8*, 24, 186, 1173, 1174, 1197
\degreesangles .. *4*, 122, 540
\degreesfalse 12, 123
\degreespolarlabels *6*, 1101
\degreestrue 122
draft (package option) *3*
\draftfalse 4
\draftPictures *5*, 1191, 1245
\drafttrue 5, 1192
\Dxone 772, 774
\Dxzero 771, 774
\Dyone 733, 734, 772, 774
\Dyzero ... 732, 734, 771, 774

E

\Ellipse *10*, 638, 1070, 1074, 1216
\ellipticArc *10*, 630, 632, 641, 1215
\endPicture 173
environments:
  Picture *5*, 124
  xpicture *5*, 174
\externalaxes *5*, 53

G

\gridcolor . *6*, 881, 1039, 1047, 1066, 1079, 1158
\gridfalse 15, 57
\gridthickness *6*, 65, 880, 1039, 1047, 1067, 1080, 1157
\gridtrue 58, 777

H

\highestlabel *8*, 146, 198, 893, 1093
\Hyperbola *10*, 37, 38, 672, 1221

I

\ifbg 18, 160
\ifdegrees 12, 87, 93, 118, 606
\ifdraft 4, 147, 1244
\ifgrid 15, 788, 838
\ifinzeroaxes 17, 827, 912, 923, 958, 1026
\iflabels 14, 785, 887, 892, 1083, 1092
\ifpolar 10, 582
\ifrputstar 11, 252
\iftics ... 13, 782, 889, 945
\ifticslabelsgrid . 16, 791
\internalaxes *5*, 53
\inzeroaxesfalse 59
\inzeroaxestrue 17, 60

L

\labelsfalse 55
\labelstrue 14, 56
\lHyperbola ... *10*, 658, 1219
\lhyperbolicArc *10*, 643, 1223

M

\makegrid *7*, 53
\makelabels *7*, 53
\makenogrid *7*, 53
\makenolabels *7*, 53, 847, 864
\makenotics *7*, 53
\maketics ... *7*, 53, 796, 801
\multicPlot *8*, 436
\multicPut *8*, 411
\multiPlot *9*, 441
\multiPlot* 441
\multiPut *8*, 416, 1094
\multiPut* 416
\multirPlot *8*, 437
\multirPlot* 437
\multirPut *8*, 412
\multirPut* 412

O

options:
  draft *3*
  pstarrows *4*

P

\Parabola *10*, 709, 1225
\parabolicArc *10*, 704, 711, 712, 1224

\pictcolor *4*, 63, 70, 166, 826, 928, 1039, 1047, 1066, 1072, 1079, 1089
\Pictlabelsep ... *8*, 241, 242, 947, 949, 953, 955, 1084, 1172
Picture (environment) *5*, 124
\PlotFunction ... *11*, 718
\PlotParametricFunction . *11*, 636, 656, 707, 757
\PlotPointsOfFunction *11*, 741, 1187, 1226
\PlotQuadraticCurve . *11*, 596, 601, 604, 1211, 1213
\plotxtic ... *7*, 795, 813, 992, 1001, 1233
\plotxtics *7*, 815, 985, 1239
\PlotxyDyData .. *11*, 734, 754
\plotytic ... *7*, 795, 814, 994, 1003, 1234
\plotytics *7*, 815, 987, 1240
\pointmark *11*, 750, 1185, 1188
\pointmarkdiam *11*, 1186, 1188
\polarcoor . *4*, 106, 109, 117
\polarfalse ... 10, 113
\polargrid ... *5*, 1057, 1242
\polarreference *4*, 111, 540, 1059
\polartrue ... 116
\Polygon ... *9*, 530
\Polyline ... *9*, 528
\printxlabel *7*, 805, 813, 1235
\printxlabels ... *7*, 817, 985
\printxticlabel *7*, 813, 1237
\printxticslabels .. *7*, 821
\printylabel *7*, 805, 814, 1236
\printylabels ... *7*, 817, 987
\printyticlabel *7*, 813, 1238
\printyticslabels .. *7*, 821
pstarrows (package option) . *4*
\Put ... *7*, 27, 30, 182, 424–427, 452, 453, 455, 457, 485, 750
\Put* ... 182

Q

\qCOS ... 238, 333
\qCurve *11*, 572, 583, 589, 600, 603, 755, 774
\qSIN ... 239, 333
\qUNITVECTOR ... 251, 333

R

\radiansangles *4*, 122, 638, 640
\radianspolarlabels *6*, 1101
\referencesystem *4*, 71, 72, 484, 1177
\refsysPoint 80, 113, 116, 204, 224, 460, 461, 466, 467, 470, 471, 574, 575
\refsyspPoint ... 105, 116
\refsyspVector 105, 115
\refsysVector ... 81, 82, 112, 115, 576, 577
\refsysxyPoint 99, 110, 113, 126–129
\refsysxyVector 99, 107, 112, 223, 934, 939
\regularPolygon *9*, 536
\rHyperbola *10*, 661, 663, 1220
\rhyperbolicArc *10*, 646, 648, 1222
\rincr ... 1062, 1069
\rlabelpos ... 1100, 1146
\rotateaxes ... *4*, 85
\rPut ... *7*, 178, 189, 210, 217, 420, 421, 445, 447, 962, 966, 973, 977
\rPut* ... 178
\rputstarfalse 11, 201
\rputstartrue ... 200
\runitdivisions *5*, 1061, 1062, 1165

S

\secundarygridcolor *7*, 881, 1161
\secundarygridthickness ... *6*, 65, 880, 1160
\secundaryticssize *6*, 65, 933, 1155
\secundaryxticssize 863, 937
\secundaryyticssize 846, 942
\standardreferencesystem 71
\strline ... 459
\symmetrize ... *4*, 85

T

\thexpct@counta ... 1085
\thexpct@countb ... 895, 959
\ticscolor *6*, 928, 1151
\ticsfalse ... 53
\ticslabelsgridfalse ... 16
\ticslabelsgridtrue 783, 786, 789
\ticssize .. *6*, 65, 932, 1153
\ticsthickness *6*, 65, 929, 1152
\ticstrue ... 13, 54
\translateorigin *4*, 85

X

\xArc ... *10*, 630
\xlabelpos *6*, 982, 1144
\xLINE ... *9*, 459, 487, 488, 500, 529, 534, 535, 547, 828, 829, 831, 832, 930, 1042, 1050, 1082, 1091, 1200
\xline ... *9*, 498, 1205
\xone ... 772, 774
\xpct@@@multiPlot . 451, 452
\xpct@@@multiPlotstar 450, 452
\xpct@@@multiPut .. 422, 426
\xpct@@@multiPutstar 423, 426
\xpct@@@Put .. 203, 204, 1196
\xpct@@@qCurve 572, 581
\xpct@@abscoorx 429, 432, 433
\xpct@@abscoory 429, 432, 434
\xpct@@detA ... 556, 559
\xpct@@detD ... 557, 560
\xpct@@detE ... 558, 561
\xpct@@dxone ... 577, 580
\xpct@@dxzero ... 576, 578
\xpct@@dyone ... 577, 580
\xpct@@dyzero ... 576, 578
\xpct@@maxy ... 714–716
\xpct@@mPut ... 411, 420, 421, 424–427, 428, 1198
\xpct@@multiPlot .. 451, 452
\xpct@@multiPlotstar 450, 452
\xpct@@multiPut ... 422, 424
\xpct@@multiPutstar 423, 424
\xpct@@Picture 124, 162
\xpct@@PlotQuadraticCurve ... 598, 599, 1212
\xpct@@Put 202, 203, 204, 1194
\xpct@@Putstar 202, 204, 1195
\xpct@@qCurve 572, 573, 1209
\xpct@@regPolygon . 537, 538

\xpct@@umaxx 696, 697, 702, 703
\xpct@@xdir . . . 500, 506, 513, 518, 523, 525, 526
\xpct@@xincr 899, 901, 910, 922
\xpct@@xone 575, 579, 901, 902, 908, 920, 925, 930
\xpct@@xzero 574, 578, 899, 900, 906, 918, 924, 930, 962, 973
\xpct@@ydir . . . 500, 506, 513, 519, 520, 524–526
\xpct@@yincr 899, 901, 911, 921
\xpct@@yone 575, 579, 901, 902, 909, 914, 919, 930
\xpct@@yzero 574, 578, 899, 900, 907, 913, 917, 930, 966, 977
\xpct@a . . . 934, 935, 939, 940
\xpct@abscoorx 204, 205
\xpct@abscoory 204, 205
\xpct@adjstics 845, 862, 872
\xpct@adjticssize 797, 802, 806, 810, 835, 931, 1028, 1033
\xpct@adjxorytics 798, 803, 807, 811, 885, 904, 1029, 1034
\xpct@alphamove . . . 225, 244
\xpct@alphaput 222, 229
\xpct@altura 245, 254, 258, 263
\xpct@amplada . 245, 259, 263
\xpct@angincr . . . 1077, 1087
\xpct@angle . . 1076, 1082, 1087, 1107, 1111, 1112
\xpct@anglea . . 544, 546–548
\xpct@angleb 546–548
\xpct@angles 335, 337, 338, 355, 357, 358, 543, 546, 1111, 1112
\xpct@angxo 587, 590
\xpct@angxz 585, 589
\xpct@angyo 588, 590
\xpct@angyz 586, 589
\xpct@arc 1085, 1086
\xpct@arrow 482, 483
\xpct@axeslabelattrib 62, 893, 963, 967, 974, 978, 1093, 1096, 1108, 1113
\xpct@b . . . 934, 935, 939, 940
\xpct@backgrd . 149, 160, 165
\xpct@bxh 50, 198, 199, 328, 330
\xpct@bxw 50, 327, 329
\xpct@circulararc . 605, 636
\xpct@convtoang 206, 212, 233, 265, 982, 983, 1100
\xpct@CorR 206, 207, 212, 213, 216, 233, 982, 983, 1100
\xpct@CorRput 176, 200, 201, 222, 225
\xpct@cosine . 87, 88, 90, 94, 96, 98, 235, 238, 241
\xpct@ctrlpoint . . . 550, 592
\xpct@defaultPut 186, 219, 1197
\xpct@degangle 1105–1107, 1109
\xpct@degreeslabel 1101, 1103, 1167
\xpct@den . . . 1104, 1106, 1117, 1121, 1122, 1124
\xpct@detA 551, 556, 569, 570
\xpct@detB . . . 552, 554, 555
\xpct@detC 553–555
\xpct@detD 554, 557, 560, 569
\xpct@detE 555, 558, 561, 570
\xpct@dirx 491–493, 495–497
\xpct@diry 491–493, 495–497
\xpct@ErrHypCons . . 35, 686
\xpct@firstint 882, 883, 1040, 1048, 1053, 1055
\xpct@firstx 531, 535
\xpct@firsty 531, 535
\xpct@grid . . 1040, 1042, 1043, 1048, 1050, 1051
\xpct@halfbox 245, 326
\xpct@hycons 664, 665, 674, 675, 687
\xpct@hypconsist 664, 674, 685
\xpct@hyperbolalastu 667, 677, 688
\xpct@hyperbolicArc 651, 652, 669, 670, 679, 680, 682, 683, 1218
\xpct@hyperbolicarc 605, 656
\xpct@hyplux 689, 692
\xpct@hypluy 649, 650, 690, 698
\xpct@Infocfg 42, 1247
\xpct@Infonocfg . . . 44, 1247
\xpct@Infopos . . 28, 188, 191
\xpct@invt 618–620
\xpct@iterateplotfunction 719, 720, 1227
\xpct@iterateplotpfunction 758, 759, 1229
\xpct@lastint . . . 1054, 1055
\xpct@lastt 746, 747
\xpct@lbl 1095, 1097
\xpct@maketics 882, 883, 884
\xpct@maxnum 52, 377
\xpct@maxy 711, 712, 716, 717
\xpct@middt 736–739
\xpct@modx 516, 517, 523, 524
\xpct@mPut 411, 420, 421, 424–427, 428
\xpct@multiPlot . . . 443, 449
\xpct@multiPlotstar 442, 449
\xpct@multiPut 418, 422
\xpct@multiPutstar . 417, 422
\xpct@multirPlot . . 439, 445
\xpct@multirPlotstar 438, 445
\xpct@multirPut . . . 414, 420
\xpct@multirPutstar 413, 420
\xpct@newt 726, 727, 729, 748, 752, 765, 766, 768
\xpct@newux 81, 83
\xpct@newuy 81, 83
\xpct@newvx 82, 84
\xpct@newvy 82, 84
\xpct@newx 80, 83
\xpct@newy 80, 83
\xpct@norm 935–938, 940–943
\xpct@num . . . 1104, 1105, 1117, 1118, 1120, 1124
\xpct@numtics . 882, 883, 1041, 1049, 1055, 1056
\xpct@oldDy 749
\xpct@oldt 722, 726, 727, 729, 743, 748–750, 752, 761, 765, 766, 768
\xpct@oldy 749, 750
\xpct@parabolalasty 710, 713
\xpct@parabolicarc . 605, 707
\xpct@pictheight 143, 144, 151, 153
\xpct@Picture . 124, 125, 164
\xpct@pictwidth 142, 144, 150, 152
\xpct@plotaxes 794, 825

\xpct@plotfunction 719, 727, 731, 1228
\xpct@plotgrid . . . 838, 1036
\xpct@plotpfunction 758, 766, 769, 1230
\xpct@PlotQuadraticCurve 597, 599, 1210
\xpct@plottics 815, 816, 988
\xpct@plotticslabels 792, 834
\xpct@plotxtics . . . 842, 882
\xpct@plotxticslabels 836, 837
\xpct@plotytics . . . 859, 882
\xpct@plotyticslabels 836, 855
\xpct@polarlabel 1086, 1101, 1102, 1168
\xpct@polarx 106, 107, 109, 110
\xpct@polary 106, 107, 109, 110
\xpct@Polygon 532, 533
\xpct@Posx 223, 226–228, 249, 251, 260, 261, 263, 264
\xpct@posx 224, 226
\xpct@Posy 223, 226–228, 249, 251, 260, 261, 263, 264
\xpct@posy 224, 226
\xpct@printlabel . . 808, 812, 895, 944, 1030, 1035
\xpct@printlabels 818, 820, 1006, 1241
\xpct@printtic 799, 804, 890, 928
\xpct@printxticslabels 822, 984
\xpct@printyticslabels 824, 984
\xpct@prtfracrad 1115, 1116
\xpct@prtlbl 1014, 1023, 1025
\xpct@PUT 177, 200, 201, 221, 1199
\xpct@Put 184, 202
\xpct@putpos . . 206, 212, 233, 235, 236, 238, 239
\xpct@Putstar 183, 202
\xpct@Px 118–120
\xpct@Py 118, 119, 121
\xpct@qCurve 578, 591
\xpct@radianslabel 1102, 1110, 1166, 1168
\xpct@radius 1063, 1069, 1070
\xpct@rdivs 1061, 1068
\xpct@regPolygon 537, 538, 539, 1208
\xpct@rint 1060, 1061, 1074, 1094
\xpct@rlblpos . . . 1094, 1100
\xpct@rPut 180, 200
\xpct@rputmxhg 199, 254
\xpct@rPutstar 179, 200
\xpct@sine . . 87, 88, 90, 94, 96, 98, 236, 239, 242
\xpct@solx 562, 566, 569, 593, 595, 735
\xpct@soly 563, 566, 570, 595
\xpct@step 723, 724, 726, 744, 745, 748, 762, 763, 765
\xpct@sticssize 933, 937, 942
\xpct@strline . 462, 464, 477
\xpct@sym 92, 94, 96
\xpct@tan 377, 379, 384, 386, 390, 392, 398, 400, 404, 406
\xpct@tempa 186, 187, 193, 229, 230, 246
\xpct@tempb 186, 187, 229, 230, 246
\xpct@tempc . . . 192, 193, 266–271, 273, 275, 277, 279, 281, 283, 285, 287, 289, 291, 293, 295–300, 302–304, 306, 308, 310, 312, 314, 316, 318, 320
\xpct@tempd 267–271, 273, 275, 277, 279, 281, 283, 285, 287, 289, 291, 293, 295–300, 302–304, 306, 308, 310, 312, 314, 316, 318, 320
\xpct@Ticcoor 1010, 1012, 1019, 1021, 1030, 1035
\xpct@ticcoor . 989, 990, 992, 994, 996, 998, 1001, 1003, 1007, 1008, 1010, 1012, 1015, 1016, 1019, 1021, 1027, 1029, 1034
\xpct@ticsinterval . 841, 858, 1037, 1045, 1053
\xpct@ticssize 932, 936, 941
\xpct@umax 668, 670, 678, 680, 683, 691
\xpct@umaxx 689, 691
\xpct@umaxy 690, 691
\xpct@umin 668, 669, 678, 679, 682
\xpct@uone 649, 651
\xpct@utwo 650, 651
\xpct@Warnbadpos . . 22, 196
\xpct@WarnIncSys . . 31, 565
\xpct@xa 693, 694, 697, 699, 700, 703
\xpct@xarrow . . 478–480, 482
\xpct@xarrowlen 474, 487, 488
\xpct@xarrowunit . . 479–481
\xpct@xaxislabelsep 938, 949, 955
\xpct@xdir 249, 251, 259, 260
\xpct@xI 72, 101, 103
\xpct@xII 72, 101, 103
\xpct@xlblpos . 962, 973, 982
\xpct@xline 499, 505, 512, 515
\xpct@xmax 136–138, 142, 153, 154
\xpct@xmed 155–158
\xpct@xmin 130–132, 142, 145, 150–152, 154
\xpct@XOne . . . 778, 828, 831, 841, 875, 1037, 1050
\xpct@xone 127, 130, 136, 169, 461, 462, 467, 468
\xpct@xonepoint . . . 471, 473
\xpct@xorigin 72, 104
\xpct@xPictsep 223, 231, 241
\xpct@xthree 129, 132, 138, 168
\xpct@xtrivVECTOR . 472, 476
\xpct@xtwo 128, 131, 137, 170
\xpct@xunit 877, 879
\xpct@XZero . . . 778, 828, 831, 832, 841, 873, 924, 925, 977, 1037, 1050
\xpct@xzero 126, 130, 136, 167, 460, 462, 466, 468
\xpct@xzeropoint . . 470, 472
\xpct@yarrow 478, 479, 481, 482

\xpct@yarrowlen 475, 487, 488
\xpct@yaxislabelsep 943, 947, 953
\xpct@Ydir 375, 376
\xpct@ydir 249, 251, 253, 258, 260
\xpct@yI 72, 101, 103
\xpct@yII 72, 101, 103
\xpct@ylblpos . 966, 977, 983
\xpct@ymax 139–141, 143, 152, 154
\xpct@ymed 155–158
\xpct@ymin 133–135, 143, 145, 150, 151, 153, 154
\xpct@YOne . . . 778, 829, 832, 858, 876, 1042, 1045
\xpct@yone 127, 133, 139, 169, 461, 462, 467, 468
\xpct@yonepoint . . . 471, 473
\xpct@yorigin 72, 104
\xpct@yPictsep 223, 231, 242
\xpct@ythree 129, 135, 141, 168
\xpct@ytwo 128, 134, 140, 170
\xpct@yunit 878, 879
\xpct@YZero . . . 778, 829, 831, 832, 858, 874, 913, 914, 973, 1042, 1045
\xpct@yzero 126, 133, 139, 167, 460, 462, 466, 468
\xpct@yzeropoint . . 470, 472
xpicture (environment) *5*, 174
\xticssize 863, 918, 920, 936, 949
\xtrivVECTOR *9*, 469, 497, 513, 1180, 1201
\xtrivvector . . . *9*, 498, 1207
\xunitdivisions *5*, 840, 848, 851, 863, 873, 875, 877, 1163
\xVECTOR *9*, 465, 493, 506, 1202
\xvector *9*, 498, 1206
\xzero 771, 774

Y

\ylabelpos *6*, 982, 1145
\yone 733, 734, 772, 774
\yticssize 846, 907, 909, 941, 947
\yunitdivisions *5*, 846, 857, 865, 868, 874, 876, 878, 1164
\yzero 732, 734, 771, 774

Z

\zerotrivvector *9*, 490, 510, 1204
\zerovector *9*, 490, 503, 1203