The xypdf package

Daniel Müllner

Version 1.7, dated 2011/03/20

Abstract

The xypdf package improves the output quality of the XY-pic package when PDF documents are generated. It produces generic PDF code for graphical elements like lines, curves and circles instead of approximating these elements with glyphs in special fonts as the original XY-pic package does. The xypdf package works with both TEX and LATEX in the occurrences of pdfTEX, XETEX and ε-TEX with dvipdfm(x) to generate PDF files. xypdf is being integrated and distributed together with XY-pic, starting with XY-pic version 3.8.

Contents

1 Introduction

The XY-pic package is a utility for typesetting diagrams in TeX and LaTeX documents. The authors of the XY-pic package put much effort into the feature that most graphical elements are coded within the limited possibilities of the device independent file format (DVI). The diagrams can thus be generated with even the most basic TeX systems and displayed universally by all device drivers. For example, diagonal lines are composed of short dashes, which are glyphs in a special font. Since there are dashes in 127 discrete directions in the font `xydash10`, diagonal lines which do not match one of these slopes look slightly rugged when they are magnified.

For a better output quality in Postscript files, the authors of the XY-pic package provided a Postscript backend for DVI-to-Postscript drivers. These extensions draw lines and curves by generic Postscript commands, thus trading a much better output quality against universality of the produced DVI files.

The development of the present package was based on XY-pic version 3.7, from 1999, where there is no support for pdfTeX. In order to produce PDF files with high-quality XY-pic diagrams, users had to use so far the Postscript file format as an intermediate step or embed the diagrams as external graphics. However, since many users directly generate PDF files from the TeX or DVI files (with bookmarks, hyperlinks and other PDF features), it is highly desirable to also have the possibility of directly generating XY-pic diagrams with high-quality PDF graphics elements.

The PDF driver provided by the xypdf package adapts the output routines of the XY-pic package to generate high-quality graphics for PDF output. It works with both pdfTeX and the two-step compilation TeX $\rightarrow$ dvipdfm(x) with an intermediate DVI file. Thus, it also works with XeTeX since this program internally uses a modification of dvipdfmx. Note that some version of ε-TeX is needed (which is anyway used by default in modern TeX installations). Figure 1 compares the output quality of a small XY-pic diagram.

The xypdf package is very similar to the Postscript drivers for XY-pic. It does not have (yet) all of their features (see Section 4) but is much more powerful in other respects, e. g. when drawing multiple curves. In general, it greatly improves graphics quality. Currently, the following features are implemented:

- Both straight lines and curves (solid, dashed, dotted and squiggled) are drawn by generic PDF commands.
- XY-pic automatically draws the symbols of which lines and curves are composed at the very beginning and end of a segment. It then distributes the inner symbols evenly

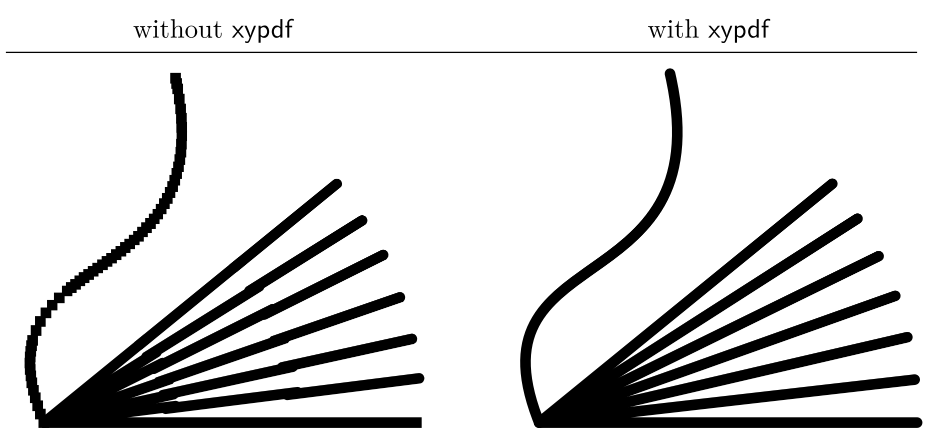
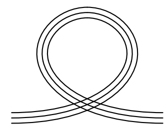


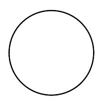
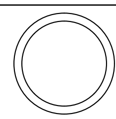
Figure 1: Comparison of XY-pic output, magnified 10 times.

across the segment. Since the arc length of a Bézier curve is normally not proportional to its parameter, this is a nontrivial task in the case of curves. The xypdf package handles this better than the original code. Compare the output in Figure 2.

- As a highlight, xypdf features a Bézier curve offset algorithm, producing high-quality curves with two or three parallel strokes.



- The `\cir` object draws circles of arbitrary radius.

without xypdf	with xypdf	code
		`\xy *\cir<16pt>{} *\cir<19pt>{} \endxy`

- xypdf supports the “rotate” extension of XY-pic.

with xypdf	code
Test text	`\xy *[@!15]\hbox{Test text} \endxy`
Test text	`\xy *[*1.5]\hbox{Test text} \endxy`

- xypdf supports the “frame” extension of XY-pic. The only intended difference to the Postscript drivers is the appearance of the `{**}` fill style. The postscript drivers fill the frame and stroke it with a line of thickness 0. The PDF driver strokes the region instead with a black line which is half as thick as the normal lines. Tip: If you want a colored boundary, overlay two frames.

with xypdf	code
Test text	`\xy *++[F**:lightgray]\hbox{Test text} \endxy`
Test text	`\xy *++[F-:red][F*:lightgray]\hbox{Test text} \endxy`

- xypdf supports the “color” extension of XY-pic. As described in the XY-pic Reference Manual, colors can be defined by the `\newcycolor` command, e. g.

```
\newxycolor{mygreen}{.5 0 1 .5 cmyk}
```

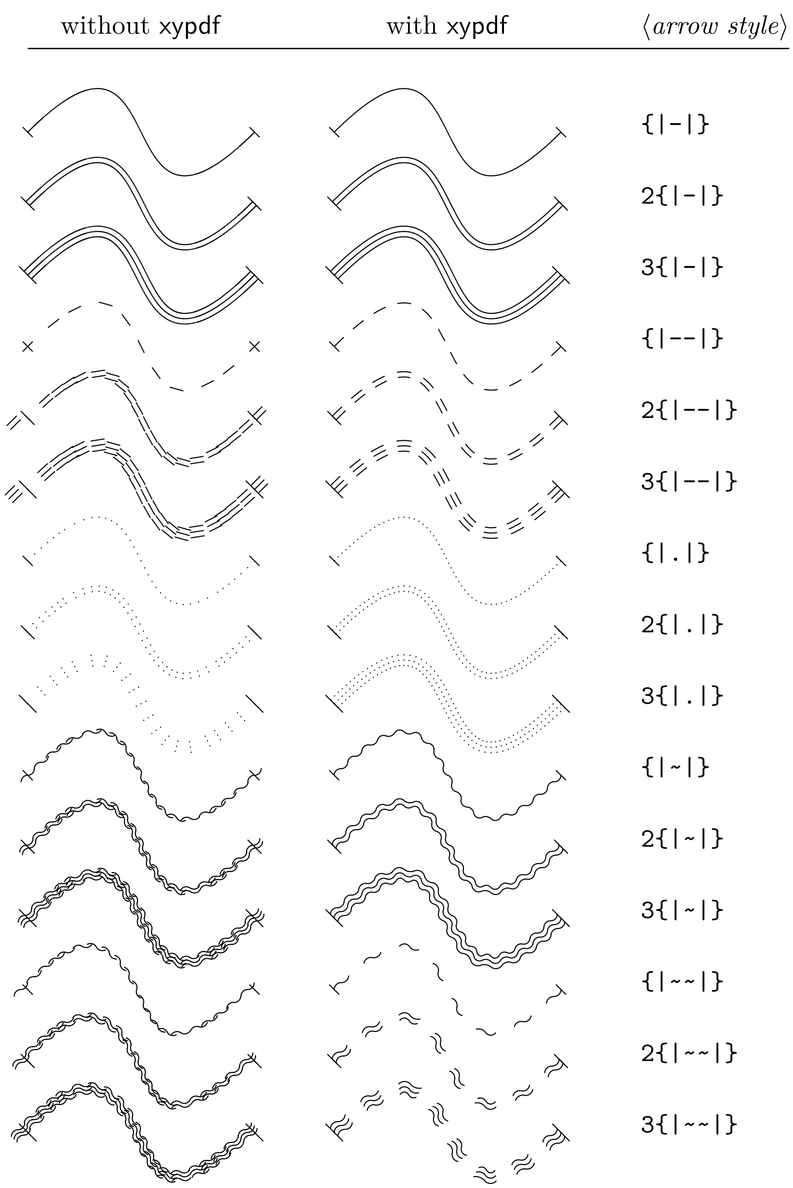
In addition, if the command `\color` is defined, e. g. if the color[1] or xcolor[2] package has been loaded, xypdf recognizes the named colors from these packages and uses the mechanisms from these packages to set colors in the output DVI or PDF file.

An example:

Orange   Green

This was generated by `\usepackage{xcolor}` in the document preamble and the following code:

[1] http://www.ctan.org/tex-archive/macros/latex/required/graphics/
[2] http://www.ctan.org/tex-archive/macros/latex/contrib/xcolor/



Code: \xy (0,0) \ar @⟨*arrow style*⟩@‘{(20,20),(10,-20)} (30,0) \endxy

Figure 2: Comparison of XY-pic output for curves with various line styles.

GreenYellow:	Yellow:	Goldenrod:
Dandelion:	Apricot:	Peach:
Melon:	YellowOrange:	Orange:
BurntOrange:	Bittersweet:	RedOrange:
Mahogany:	Maroon:	BrickRed:
Red:	OrangeRed:	RubineRed:
WildStrawberry:	Salmon:	CarnationPink:
Magenta:	VioletRed:	Rhodamine:
Mulberry:	RedViolet:	Fuchsia:
Lavender:	Thistle:	Orchid:
DarkOrchid:	Purple:	Plum:
Violet:	RoyalPurple:	BlueViolet:
Periwinkle:	CadetBlue:	CornflowerBlue:
MidnightBlue:	NavyBlue:	RoyalBlue:
Blue:	Cerulean:	Cyan:
ProcessBlue:	SkyBlue:	Turquoise:
TealBlue:	Aquamarine:	BlueGreen:
Emerald:	JungleGreen:	SeaGreen:
Green:	ForestGreen:	PineGreen:
LimeGreen:	YellowGreen:	SpringGreen:
OliveGreen:	RawSienna:	Sepia:
Brown:	Tan:	Gray:
Black:	White:	

Figure 3: Additional color names provided by `\UseCrayolaColors`.

```
\definecolor{mygreen}{cmyk}{.5 0 1 .5}
\xymatrix{*+[orange][F-:blue]\hbox{Orange}
  & *[mygreen]\hbox{Green}}
```

When a named color has been defined by both `\newxycolor` and by a color package command like `\definecolor`, the XY-pic definition overrides the general one.

The XY-pic command `\UseCrayolaColors` defines a set of color names, as explained in the XY-pic Reference Manual. Figure 3 lists these colors.

If you notice any unwanted behavior, please generate a minimal example and e-mail it to the author of this package. Current contact details are available at http://math.stanford.edu/~muellner. Please report situations where the algorithms produce arithmetic overflows. Also, the code is not really optimized for speed but for accuracy, so feel free to report a significant slowdown of the compiling process for your thesis/paper/book.

2 Usage

Use `pdf` as an option to the XY-pic package, as in

```
\usepackage[pdf]{xy}
```

or

```
\usepackage{xy}
\xyoption{pdf}
```

for LaTeX and

```
\input xy
\xyoption{pdf}
```

for plain TEX. Do not use one of the other driver options to XY-pic like `dvips`, since combining two drivers will most likely result in mutilated diagrams.

The `xypdf` functionality can be switched off and on within the document by `\xypdfoff` and `\xypdfon`.

When you use plain TEX, make sure that `xypdf` is loaded after any global changes to the math fonts.

3 Acknowledgements

Since the `xypdf` package extends XY-pic, some ideas are adopted from this package and its Postscript backend, and the author gratefully acknowledges the service which Kristoffer H. Rose and Ross Moore did to the mathematical community with their original package.

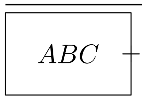
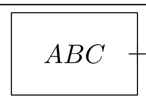
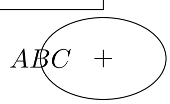
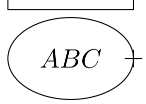
4 To do

- Support for the “line” extension.

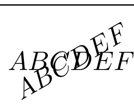
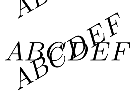
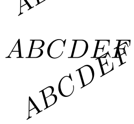
5 Differences between **xypdf** and the `dvips` backend

Most users of `xypdf` will have used the `dvips` backend before. In this section, we list the the intentional differences between `xypdf`’s behavior and the `dvips` driver. If you see further differences which are not mentioned here, this could be a bug, and please notify the author, Daniel Müllner.

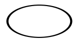
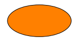
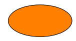
- `xypdf` draw frames always around the current point c.

with `dvips`	with `xypdf`	code
		`\xy *{+} *+<2pc>!R[F-]{ABC}\endxy`
		`\xy *{+} *+<2pc>!R[Fe]{ABC}\endxy`

- The center of rotation and scaling is always the current point c.

with `dvips`	with `xypdf`	code
		`\xy *[@!30]{ABCDEF} *{ABCDEF} \endxy`
		`\xy *!R[@!30]{ABCDEF} *!R{ABCDEF} \endxy`

- Line frames which are drawn as `\frm` objects are always black with the `dvips` backend, as opposed to filled frames. `xypdf` changes the color for both line and filled frames.

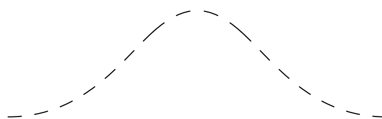
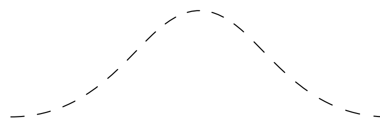
with `dvips`	with `xypdf`	code
		`\xy *+<2pc,1pc>[o]{} *[orange]\frm{e} \endxy`
		`\xy *+<2pc,1pc>[o]{} *[orange]\frm{**} \endxy`

The `[F...]` object modifiers give the same colors to frames with both drivers.

6 The fine print: curves with multiple segments

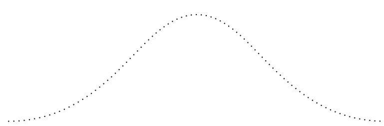
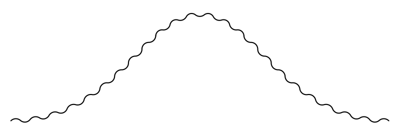
Since the dashes in Bézier segments are aligned to the boundary points, this would result in dashes of double length when a curve is composed of several Bézier segments, as shown in the upper left diagram in Figure 4. To avoid this, `xypdf` records the end point of each segment and adapts the dash pattern whenever the starting point of a segment coincides with the end point of the previous one (see the upper right diagram). Analogous improvements apply to the "dotted" and "squiggled" line styles.

Since this mechanism does not exist in the original XY-pic, it can be switched on and off by `\xypdfcontpatternon` and `\xypdfcontpatternoff`. By default, it is switched on.

`\xypdfcontpatternoff`	`\xypdfcontpatternon` (default)
	

code:

```
\xy (0,0);(50,0)
  **\crv{~**\dir{--} (10,0)&(20,15)&(30,15)&(40,0)}
\endxy
```

similar improvements: dotted curve	squiggled curve
	

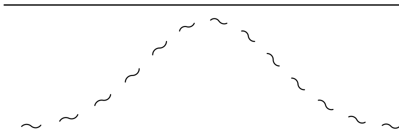
broken squiggled curve


Figure 4: Seamless patterns for curves with multiple segments.

7 Troubleshooting

- I get the error message `pdfTeX version 1.40.0 or higher is needed for the xypdf package with PDF output`

  You seem the use an old version of pdfTEX. If you cannot update your TEX system for some reason, you may still use the `xypdf` package in DVI mode and produce a PDF file via dvipdfm(x). The pathway TEX → dvipdfm(x) is preferable in many cases anyway since it usually produces much smaller PDF files.

- I get the error message `eTeX is needed for the xypdf package`.

  In your TeX installation, the ε-TeX features are not enabled, although they most certainly can be in any reasonably modern TeX installation. You must probably rebuild the TeX and LaTeX format files with ε-TeX enabled. Please consult the documentation of your TeX distribution on how to rebuild the format files.

8 Copyright, license and disclaimer

9 Implementation

This is the code for the file xypdf.tex. From version 1.4 on, it is loaded as the option `pdf` to XY-pic.

`\xypdfversion` `\xypdfdate`

`\xyprovide` defines `\xypdfversion`.

```
⟨*basic⟩
\ifx\xyloaded\undefined\input xy \fi
\xyprovide{pdf}{PDF driver}{1.7}%
  {Daniel M\"ullner\newline}%
  {\url{http://math.stanford.edu/~muellner}}{}
\ifx\makeatletter\undefined\input miniltx \fi
\newcommand*\xypdfdate{2011/03/20}
\newdriver{%
  \xyaddsupport{pdf}\xP@pdf@on
  \xyaddsupport{color}\xP@color@on
  \xyaddsupport{curve}\xP@curve@on
  \xyaddsupport{frame}\xP@frame@on
%  \xyaddsupport{line}\xP@line@on
  \xyaddsupport{rotate}\xP@rotate@on
}
\xyaddunsupport{pdf}\xP@pdf@off
\xyaddunsupport{color}\xP@color@off
\xyaddunsupport{curve}\xP@curve@off
\xyaddunsupport{frame}\xP@frame@off
%\xyaddunsupport{line}\xP@line@off
\xyaddunsupport{rotate}\xP@rotate@off
```

See the end of the file for the code that loads the other files xypdf-*.tex.

`\xypdfon` `\xypdfoff`

Commands for switching the driver on and off.

```
\newcommand*\xypdfon{%
  \xP@pdf@on
  \xP@color@on
  \xP@curve@on
  \xP@frame@on
  \xP@line@on
  \xP@rotate@on
}
\newcommand*\xypdfoff{%
  \xP@pdf@off
  \xP@color@off
  \xP@curve@off
  \xP@frame@off
  \xP@line@off
  \xP@rotate@off
}
```

Test for ε-TEX

```
\ifx\unexpanded\@undefined
  \PackageError{xypdf}{eTeX is needed for the xypdf package}{}
\fi
```

`\xP@testpdfsave`

Test for `\pdfsave`, which was introduced in pdfTEX version 1.40.0.

```
\newcommand*\xP@testpdfsave{%
  \ifpdf
    \ifx\pdfsave\@undefined
```

```
    \PackageError{xypdf}{pdfTeX version 1.40.0 or higher is needed for the %
      xypdf^^J%
      package with PDF output}{}%
  \fi
\fi
\let\xP@testpdfsave\@undefined
}
```

\xP@warning

Check for \PackageWarning.

```
\ifx\PackageWarning\@undefined
  \newcommand*\xP@warning[2]{{%
    \newlinechar`\^^J%
    \@warning{Package #1 Warning: #2\@empty.}%
  }}
\else
  \newcommand*\xP@warning{\PackageWarning}
\fi
```

\xP@pdf@on
\xP@pdf@off

The following initializations are necessary for each supported extension, otherwise the \xP@hook commands will not work.

```
\newcommand*\xP@pdf@on{}
\newcommand*\xP@pdf@off{}
```

\xP@color@on
\xP@color@off

```
\newcommand*\xP@color@on{}
\newcommand*\xP@color@off{}
```

\xP@curve@on
\xP@curve@off

```
\newcommand*\xP@curve@on{}
\newcommand*\xP@curve@off{}
```

\xP@frame@on
\xP@frame@off

```
\newcommand*\xP@frame@on{}
\newcommand*\xP@frame@off{}
```

\xP@line@on
\xP@line@off

```
\newcommand*\xP@line@on{}
\newcommand*\xP@line@off{}
```

\xP@rotate@on
\xP@rotate@off

```
\newcommand*\xP@rotate@on{}
\newcommand*\xP@rotate@off{}
```

\xP@hook

Commands for switching the driver on and off.

```
\newcommand*\xP@hook[2]{%
  \edef\next@{%
    \let\expandafter\noexpand\csname xP@old@#2\endcsname
      \expandafter\noexpand\csname#2\endcsname}%
  \next@
  \expandafter\edef\csname xP@#1@on\endcsname{%
    \unexpanded\expandafter\expandafter\expandafter{\csname xP@#1@on\endcsname}%
    \let\expandafter\noexpand\csname#2\endcsname
      \expandafter\noexpand\csname xP@#2\endcsname
  }%
  \expandafter\edef\csname xP@#1@off\endcsname{%
```

```
  \unexpanded\expandafter\expandafter\expandafter{\csname xP@#1@off\endcsname}%
   \let\expandafter\noexpand\csname#2\endcsname
     \expandafter\noexpand\csname xP@old@#2\endcsname
  }%
}
```

\xP@defpdfliteral \xP@literal \xP@cm \xP@setcolor \xP@resetcolor

Two possibilities to insert literal PDF commands, one for pdftex and one for dvipdfm(x). The command `\xP@cm` changes the current transformation matrix.

```
\newcommand*\xP@defpdfliteral{%
\ifpdf
  \newcommand*\xP@literal[1]{\pdfsave\pdfliteral{##1}\pdfrestore}
  \newcommand*\xP@cm[5]{%
    \pdfsave
    \pdfsetmatrix{##1 ##2 ##3 ##4}%
    ##5%
    \pdfrestore
  }
```

Mimick pdftex.def from the graphicx package.

```
  \@ifundefined{@pdfcolorstack}{%
    \def\@pdfcolorstack{\z@}%
  }{}%
  \newcommand*\xP@setcolor[3]{%
    \pdfcolorstack\@pdfcolorstack push{##1 ##2 ##1 ##3}}
  \newcommand*\xP@resetcolor{\pdfcolorstack\@pdfcolorstack pop\relax}%
\else
  \newcommand*\xP@literal{%
    \xP@warning{xypdf}{%
    The produced DVI file is NOT PORTABLE. Convert it with^^J%
    dvipdfm(x) to the PDF format but do not expect the DVI file itself to be^^J%
    displayed correctly\@gobble}%
    \global\let\xP@literal\xP@literal@
    \xP@literal
  }
  \newcommand*\xP@literal@[1]{\special{pdf:content ##1}}
  \newcommand*\xP@cm[5]{%
    \special{pdf:btrans matrix ##1 ##2 ##3 ##4 0 0}%
    ##5%
    \special{pdf:etrans}%
  }
  \newcommand*\xP@setcolor[3]{\special{pdf:bcolor[##1]}}
  \newcommand*\xP@resetcolor{\special{pdf:ecolor}}%
\fi
\let\xP@defpdfliteral\@undefined
}
```

Rely on the ifpdf package to test for PDF output. The `\AtEndOfPackage` is necessary if xypdf is loaded as an option in `\usepackage[`⟨*options*⟩`]{xy}`. If it is called as a plain TeX package, the commands below can be executed immediately.

```
\DN@{\@firstofone}
\DNii@{xy}
\ifx\@currname\nextii@
  \ifx\AtEndOfPackage\@undefined
  \else
    \DN@{\AtEndOfPackage}%
  \fi
```

```
\fi
\next@
 {\RequirePackage{ifpdf}%
  \xP@testpdfsave
  \xP@defpdfliteral}
```

`\xP@digits` Set the precision for dimension output according to pdfTEX's `\pdfdecimaldigits`. If this number is not defined, use dvipdfm's default precision, which is two decimals.

```
\ifx\pdfdecimaldigits\@undefined
  \newcommand*\xP@digits{2}
\else
  \@ifdefinable\xP@digits\relax
  \xdef\xP@digits{\the\pdfdecimaldigits}
  \ifnum\pdfdecimaldigits<2
    \xP@warning{xypdf}{%
      The precision in \string\pdfdecimaldigits\space is only \xP@digits\space
      decimals.^^J%
      It is recommended to set \string\pdfdecimaldigits\space to 2 or 3 for %
      best output quality\@gobble}
  \fi
\fi
```

`\xP@dim` Conversion between TEX points (pt) and PDF/Postscript points (bp)

```
\newcommand*\xP@dim[1]{%
  \expandafter\xP@removePT\the\dimexpr(#1)*800/803\relax\space}
```

`\xP@precdim` Precise conversion between TEX points (pt) and PDF/Postscript points (bp). No truncation.

```
\newcommand*\xP@precdim[1]{\xP@EARPT\dimexpr(#1)*800/803\relax\space}
```

`\xP@EARPT`

```
\newcommand*\xP@EARPT{\expandafter\removePT@\the}
```

`\xP@coor` Coordinates: two dimensions

```
\newcommand*\xP@coor[1]{\xP@dim{#1}\xP@dim}
```

`\xP@removePT` The following two macros round and truncate a dimension to the desired number of decimal digits.

```
\@ifdefinable\xP@removePT\relax
{\@makeother\p\@makeother\t\gdef\xP@removePT#1pt{\xP@removePT@#10000@}}
```

`\xP@removePT@`

```
\@ifdefinable\xP@removePT@\relax
\ifcase\xP@digits
```

0 decimals

```
  \def\xP@removePT@#1.#2#3@{%
    \ifnum#2<5
      #1%
    \else
      \the\numexpr-\if-#1-\else-#1+\fi\@ne\relax
    \fi
  }
\or
```

1 decimal

```
  \def\xP@removePT@#1#2.#3#4#5@{%
    \ifnum#4<5
      #1#2%
      \if#30%
      \else
        .#3%
      \fi
    \else
      \expandafter\xP@removePT
      \the\dimexpr#1#2.#3pt+\if#1--\fi.12pt\relax
    \fi
  }
\or
```

2 decimals

```
  \def\xP@removePT@#1#2.#3#4#5#6@{%
    \ifnum#5<5
      #1#2%
      \if#40%
        \if#30%
        \else
          .#3%
        \fi
      \else
        .#3#4%
      \fi
    \else
      \expandafter\xP@removePT
      \the\dimexpr#1#2.#3#4pt+\if#1--\fi786sp\relax
    \fi
  }
\or
```

3 decimals

```
  \def\xP@removePT@#1#2.#3#4#5#6#7@{%
    \ifnum#6<5
      #1#2%
      \if#50%
        \if#40%
          \if#30%
          \else
            .#3%
          \fi
        \else
          .#3#4%
        \fi
      \else
        .#3#4#5%
      \fi
    \else
      \expandafter\xP@removePT
      \the\dimexpr#1#2.#3#4#5pt+\if#1--\fi79sp\relax
    \fi
  }
\or
```

4 decimals

```
\def\xP@removePT@#1#2.#3#4#5#6#7#8@{%
  \ifnum#7<5
    #1#2%
    \if#60%
      \if#50%
        \if#40%
          \if#30%
          \else
            .#3%
          \fi
        \else
          .#3#4%
        \fi
      \else
        .#3#4#5%
      \fi
    \else
      .#3#4#5#6%
    \fi
  \else
    \expandafter\xP@removePT
    \the\dimexpr#1#2.#3#4#5#6pt+\if#1--\fi8sp\relax
  \fi
}
\else
```

5 or more decimals: no truncation

```
  \let\xP@dim\xP@precdim
\fi
```

`\xP@lw` `\xP@preclw` Find out the default line width in the math fonts. This is done at the beginning of the document, when hopefully all potential changes to math fonts have taken place.

```
\AtBeginDocument{%
```

Initialize math fonts

```
  {\setboxz@h{$ $}}%
  \@ifdefinable\xP@lw\relax
  \@ifdefinable\xP@preclw\relax
  \edef\xP@preclw{\the\fontdimen8\textfont3}%
  \edef\xP@lw{\xP@dim\xP@preclw}%
  \PackageInfo{xypdf}{Line width: \xP@preclw}%
}
```

9.1 Straight lines

`\line@` Also change the code for `\dir{-}` as an object. Now these dashes are not drawn from the dash font any more but by generic PDF line commands.

```
\xP@hook{pdf}{line@}
\newcommand*\xP@line@{%
  \setboxz@h{%
    \xP@setsolidpat
    \xP@stroke{0 0 m \xP@coor{\cosDirection\xydashl@}{\sinDirection\xydashl@}l}%
  }%
  \U@c\sinDirection\xydashl@
  \D@c\z@
  \ifdim\U@c<\z@
    \multiply\U@c\m@ne
```

```
    \xP@swapdim\U@c\D@c
  \fi
  \ht\z@\U@c
  \dp\z@\D@c
  \R@c\cosDirection\xydashl@
  \L@c\z@
  \ifdim\R@c<\z@
    \multiply\R@c\m@ne
    \xP@swapdim\L@c\R@c
  \fi
  \hskip\L@c\boxz@\hskip\R@c
  \edef\tmp@{\egroup\U@c\the\U@c\D@c\the\D@c\L@c\the\L@c\R@c\the\R@c}%
  \tmp@
  \Edge@c={\rectangleEdge}%
  \edef\Upness@{\ifdim\z@<\U@c1\else0\fi}%
  \edef\Leftness@{\ifdim\z@<\L@c1\else0\fi}%
  \def\Drop@@{\styledboxz@}%
  \def\Connect@@{\solid@}%
}
```

`\solid@` `\xP@solid@` This is the hook for solid straight lines. Derived from `\xyPSsolid@` in `xyps.tex`.

```
\xP@hook{pdf}{solid@}
\newcommand*\xP@solid@{\straight@\xP@solidSpread}
```

`\xP@solidSpread`

```
\@ifdefinable\xP@solidSpread\relax
\def\xP@solidSpread#1\repeat@{{%
```

Neglect zero-length lines.

```
  \@tempswatrue
  \ifdim\X@p=\X@c
  \ifdim\Y@p=\Y@c
    \@tempswafalse
  \fi
  \fi
  \if@tempswa
    \xP@setsolidpat
    \xP@stroke{\xP@coor\X@p\Y@p m \xP@coor\X@c\Y@c l}%
  \fi
}}
```

`\xP@pattern`

```
\newcommand*\xP@pattern{}
```

`\xP@setsolidpat` Pattern for solid lines

```
\newcommand*\xP@setsolidpat{%
  \def\xP@pattern{1 J 1 j []0 d}%
  \global\let\xP@lastpattern\xP@solidmacro
}
```

`\xP@stroke`

```
\newcommand*\xP@stroke[1]{\xP@literal{\xP@lw w \xP@pattern\space#1 S}}
```

`\dash@` `\xP@dash@` This is the hook for dashed straight lines. Derived from `\xyPSdashed@` in `xyps.tex`.

```
\xP@hook{pdf}{dash@}
\newcommand\xP@dash@{\line@\def\Connect@@{\straight@\xP@dashedSpread}}
```

\xP@dashedSpread

```
\@ifdefinable\xP@dashedSpread\relax
\def\xP@dashedSpread#1\repeat@{{%
  \xP@veclen
```

Neglect zero-length lines.

```
  \ifdim\@tempdimb>\z@
    \xP@setdashpat
    \xP@savec
    \xP@stroke{\xP@coor\X@p\Y@p m \xP@coor\X@c\Y@c l}%
  \fi
}}
```

\xP@setdashpat The formula for the dash length is the same as in the `dashed` operator in `xypsdict.tex`:

$$(\text{dash length}) = \frac{l}{2 \cdot \operatorname{round}\left(\frac{l+d}{2d}\right) - 1},$$

where l is the length of the line and d is the minimal dash length.

The length l must be in `\@tempdimb`. The dash length is returned in `\@tempdima`.

```
\newcommand*\xP@setdashpat{%
  \xP@testcont\xP@dashmacro
  \ifxP@splinecont
```

Special pattern in case this line continues another dashed segment.

```
    {\count@\numexpr2*((\@tempdimb-\xydashl@/3)/(2*\xydashl@))\relax
    \global\dimen@i
      \ifnum\count@>\z@
        \dimexpr\@tempdimb/\count@\relax
      \else
        \z@
      \fi
    }%
    \@tempdima\dimen@i
    \edef\xP@pattern{1 J 1 j [%
      \ifdim\@tempdima>\z@
        \xP@precdim\@tempdima]\xP@precdim\@tempdima
      \else
        ]0 %
      \fi
    d}%
  \else
    \@tempdima
      \ifdim\@tempdimb>\xydashl@
        \dimexpr\@tempdimb/(2*((\@tempdimb+\xydashl@)/(2*\xydashl@))-1)\relax
      \else
        \z@
      \fi
    \edef\xP@pattern{1 J 1 j [%
      \ifdim\@tempdima>\z@\xP@precdim\@tempdima\fi
    ]0 d}%
  \fi
  \global\let\xP@lastpattern\xP@dashmacro
}
```

\xP@setcldashpat Dash pattern for closed paths. Offset is half of the dash length to avoid artifacts.

```
\newcommand*\xP@setcldashpat{%
  {\count@\numexpr2*((\@tempdimb-\xydashl@/3)/(2*\xydashl@))\relax
  \xdef\@gtempa{1 J 1 j [\ifnum\count@>\z@\xP@precdim{\@tempdimb/\count@}\fi]%
    \ifnum\count@>\z@\xP@precdim{\@tempdimb*3/2/\count@}\else0 \fi d}%
  }%
  \edef\xP@pattern{\@gtempa}%
}
```

`\point@` `\xP@point@` This is the hook for points. Derived from `\xyPSpoint@` in `xyps.tex`.

```
\xP@hook{pdf}{point@}
\newcommand*\xP@point@{\xP@zerodot\egroup\Invisible@false
  \Hidden@false\def\Leftness@{.5}\def\U@pness@{.5}\ctipEdge@
  \def\Drop@@{\styledboxz@}%
  \def\Connect@@{\straight@\xP@dottedSpread}%
}
```

`\xP@zerodot`

```
\newcommand*\xP@zerodot{%
  \hb@xt@\z@{\hss
    \vbox to\z@{\vss\hrule\@width\xP@preclw\@height\xP@preclw\vss}%
  \hss}%
}
```

`\xP@dottedSpread`

```
\@ifdefinable\xP@dottedSpread\relax
\def\xP@dottedSpread#1\repeat@{{%
  \xP@veclen
  \ifdim\@tempdimb>\z@
    \xP@setdottedpat
    \xP@savec
    \xP@stroke{\xP@coor\X@p\Y@p m \xP@coor\X@c\Y@c l}%
  \fi
}}
```

`\xP@setdottedpat` The formula for the distance between dots is the same as in the `dotted` operator in `xypsdict.tex`:

$$\text{(dot distance)} = \frac{l}{\text{round}\left(\frac{l}{2\text{pt}}\right) + 1},$$

where l is the length of the line.

The length l must be in `\@tempdimb`.

```
\newcommand*\xP@setdottedpat{%
  \xP@testcont\xP@dotmacro
  \ifxP@splinecont
    \@tempdima\dimexpr\@tempdimb/(\@tempdimb/131072+1)-\xP@preclw\relax
    \edef\xP@pattern{%
      0 J [%
```

Produce a dot pattern only when the segment is long enough.

```
      \ifdim\@tempdima>\z@
        \xP@precdim\xP@preclw\xP@precdim\@tempdima
      \fi
```

Advance the offset very slightly by 1sp to really hide the first dot in the viewer. (This improves the display at least in the author's PDF-Xchange viewer.)

```
      ]\xP@precdim{\xP@preclw+1sp}d}%
```

```
\else
  \advance\@tempdimb-\xP@preclw
  \ifdim\@tempdimb<\z@\@tempdimb\z@\fi
  \@tempdima\dimexpr\@tempdimb/(\@tempdimb/131072+1)-\xP@preclw\relax
  \edef\xP@pattern{%
    0 J [%
```

Produce a dot pattern only when the segment is long enough.

```
    \ifdim\@tempdima>\z@
      \xP@precdim\xP@preclw\xP@precdim\@tempdima
    \fi
    ]0 d}%
\fi
\global\let\xP@lastpattern\xP@dotmacro
}
```

`\xP@setcldottedpat` Dotted pattern for closed paths. Offset is half of the dot distance to avoid artifacts.

```
\newcommand*\xP@setcldottedpat{%
  \@tempdima\dimexpr\@tempdimb/(\@tempdimb/131072+1)-\xP@preclw\relax
  \edef\xP@pattern{%
    0 J [%
    \ifdim\@tempdima>\z@
      \xP@precdim\xP@preclw\xP@precdim\@tempdima
    \fi
    ]\xP@precdim{\dimexpr\xP@preclw+\@tempdima/2\relax}d}%
}
```

In contrast to the Postscript drivers for XY-pic, where some computations are left to the Postscript code, all arithmetic for the PDF output must be done by TEX itself. With TEX's rudimentary fixed-point arithmetic, it is still a pain to compute even the length of a line segment, but things have become considerably easier with ε-TEX.

`\xP@abs` Absolute value

```
\newcommand*\xP@abs[1]{\ifdim#1<\z@\multiply#1\m@ne\fi}
```

`\xP@ifabsless`

```
\newcommand*\xP@ifabsless[2]{\ifpdfabsdim#1<#2}
\ifx\ifpdfabsdim\@undefined
  \renewcommand*\xP@ifabsless[2]{\ifdim\ifdim#1<\z@-\fi#1<\ifdim#2<\z@-\fi#2}
  \@gobble\fi
\fi
```

`\xP@swapdim` Works unless parameter `#2` is `\@tempdima`.

```
\newcommand*\xP@swapdim[2]{\@tempdima#1#1#2#2\@tempdima}
```

`\xP@swapnum` Works unless parameter `#2` is `\@tempcnta`.

```
\newcommand*\xP@swapnum[2]{\@tempcnta#1#1#2#2\@tempcnta}
```

`\xP@min` Maximum of two lengths

```
\newcommand*\xP@min[2]{\ifdim#1<#2#1\else#2\fi}
```

`\xP@max` Maximum of two lengths

```
\newcommand*\xP@max[2]{\ifdim#1>#2#1\else#2\fi}
```

`\xP@Max` Assigns `#1` the maximum of `#1` and the absolute value of `#2`.

```
\newcommand*\xP@Max[2]{#1\ifdim#2<\z@\xP@max#1{-#2}\else\xP@max#1#2\fi}
```

`\xP@sqrt` Square root algorithm. The argument is in `\@tempdima`, and the start value for the iteration in `\@tempdimc`. The result goes into `\@tempdimb`.

```
\newcommand*\xP@sqrt{%
  \loop
    \@tempdimb\dimexpr(\@tempdimc+(\@tempdima*\p@/\@tempdimc))/2\relax
```

ε-TeX's `\unless` instead of `\else` since the plain TeX `\loop` cannot deal with `\else`.

```
  \unless\ifdim\@tempdimc=\@tempdimb
```

iterate: (old approx.) := (new approx.)

```
    \@tempdimc\@tempdimb\relax
  \repeat
}
```

`\xP@veclen` Absolute length of the vector (`\d@X`, `\d@Y`). The result goes into the register `\@tempdimb`. Several LaTeX registers are used as temporary registers, so this function is called safely within a group.

(Maybe it is not necessary to scale the coordinates so much as it is done here, and a simpler code would be fine as well.)

```
\newcommand*\xP@veclen{{%
  \xP@veclen@
  \global\dimen@i\@tempdimb
  }\@tempdimb\dimen@i
}
```

`\xP@veclen@`

```
\newcommand*\xP@veclen@{%
  \xP@abs\d@Y
```

1) Strictly vertical vector

```
  \ifdim\d@X=\z@
    \@tempdimb\d@Y
  \else
    \xP@abs\d@X
```

2) Strictly horizontal vector

```
    \ifdim\d@Y=\z@
      \@tempdimb\d@X
    \else
```

3) Diagonal vector. $5931642\mathrm{sp} = \sqrt{\texttt{\textbackslash maxdimen}/2}$. Test whether the components are small enough so that their sum of squares does not generate an arithmetic overflow.

```
      \@tempswatrue
      \ifdim\d@X>5931641sp\relax\@tempswafalse\fi
      \ifdim\d@Y>5931641sp\relax\@tempswafalse\fi
      \if@tempswa
```

3a) Small vector. `\count@` contains a scaling factor for a precise fixed-point arithmetic.

```
        \count@\@ne
        \loop
          \@tempdima\dimexpr\d@X*\d@X/\p@+\d@Y*\d@Y/\p@\relax
```

If the coordinates are small enough, scale them up to improve precision.

```
        \ifdim\@tempdima<4096pt
          \@tempcnta\ifdim\@tempdima<1024pt\ifdim\@tempdima<256pt8\else4\fi%
            \else\tw@\fi
          \multiply\d@X\@tempcnta
          \multiply\d@Y\@tempcnta
```

```
          \multiply\count@\@tempcnta
        \repeat
```

Starting value for the square root algorithm

```
        \@tempdimc\dimexpr(\d@X+\d@Y)*3/4\relax
        \xP@sqrt
```

Rescale

```
        \@tempdimb\dimexpr\@tempdimb/\count@\relax
      \else
        \ifdim\d@X>83042982sp\relax\@tempswatrue\fi
        \ifdim\d@Y>83042982sp\relax\@tempswatrue\fi
        \if@tempswa
```

3b) Large vector. Scale the coordinates down to avoid an overflow. 11927552sp = 182pt

```
          \@tempdima\dimexpr\d@X/182*\d@X/11927552+\d@Y/182*\d@Y/11927552\relax
          \@tempdimc\dimexpr(\d@X+\d@Y)*3/728\relax
          \xP@sqrt
          \multiply\@tempdimb182\relax
        \else
```

3c) Medium vector. Also scale the coordinates down. 12845056sp = 196pt = 14^2pt

```
          \@tempdima\dimexpr\d@X*\d@X/12845056+\d@Y*\d@Y/12845056\relax
          \@tempdimc\dimexpr(\d@X+\d@Y)*3/56\relax
          \xP@sqrt
          \multiply\@tempdimb14\relax
        \fi
      \fi
    \fi
  \fi
}
```

9.2 Squiggled lines

\squiggledSpread@
\xP@squiggledSpread@

This is the hook for squiggled straight lines.

```
\xP@hook{pdf}{squiggledSpread@}
\@ifdefinable\xP@squiggledSpread@\relax
\def\xP@squiggledSpread@#1\repeat@{{%
  \xP@veclen
```

Neglect zero-length lines.

```
  \ifdim\@tempdimb>\z@
    \edef\@tempa{\xP@coor\X@p\Y@p m }%
    \toks@\expandafter{\@tempa}%
```

`\@tempcnta` = number of squiggles

```
    \@tempcnta\numexpr\@tempdimb/\xybsqll@\relax
    \ifnum\@tempcnta<\tw@\@tempcnta\tw@\fi
    \@tempdima\dimexpr\d@X/\@tempcnta\relax
    \@tempdimc\dimexpr\d@Y/\@tempcnta\relax
```

Reverse the direction of the little arcs, if the last squiggle from the previous segment makes it necessary.

```
    \xP@testcont\xP@oddsquigglemacro
    \ifxP@splinecont
      \def\xP@squigsign{-}%
    \else
      \let\xP@squigsign\@empty
    \fi
```

```
    \count@\z@
    \loop
```

The fraction is the continuous fraction approximation for the best spline approximation to a quarter circle ($147546029/534618434 \approx \frac{1}{2} \cdot 0.55196760761152504532$).

```
    \xP@append\toks@{%
      \xP@coor{\X@p+\d@X*\count@/\@tempcnta+(\@tempdima
        -\xP@squigsign\ifodd\count@-\fi\@tempdimc)*147546029/534618434}%
      {\Y@p+\d@Y*\count@/\@tempcnta+(\@tempdimc
        +\xP@squigsign\ifodd\count@-\fi\@tempdima)*147546029/534618434}%
    }%
    \advance\count@\@ne
    \xP@append\toks@{%
      \xP@coor{\X@p+\d@X*\count@/\@tempcnta-(\@tempdima
        -\xP@squigsign\ifodd\count@-\fi\@tempdimc)*147546029/534618434}%
      {\Y@p+\d@Y*\count@/\@tempcnta-(\@tempdimc
        +\xP@squigsign\ifodd\count@-\fi\@tempdima)*147546029/534618434}%
      \xP@coor{\X@p+\d@X*\count@/\@tempcnta}%
      {\Y@p+\d@Y*\count@/\@tempcnta}%
      c }%
    \ifnum\count@<\@tempcnta
    \repeat
    \xP@setsolidpat
```

Record the direction of the last squiggle.

```
    \global\expandafter\let\expandafter\xP@lastpattern
    \ifodd\numexpr\count@\if\xP@squigsign-+1\fi\relax
      \xP@oddsquigglemacro
    \else
      \xP@evensquigglemacro
    \fi
    \xP@savec
    \xP@stroke{\the\toks@}%
  \fi
}}
```

\xP@squigsign

```
\newcommand*\xP@squigsign{}
```

\xP@append

```
\newcommand*\xP@append[2]{{%
  \edef\@tempa{#1{\the#1#2}}%
  \expandafter}\@tempa
}
```

9.3 Circles

\circhar@@
\xP@circhar@@

Replacement macro for the circle chars.

```
\xP@hook{pdf}{circhar@@}
\newcommand*\xP@circhar@@[1]{%
  \expandafter\xP@circhar@@@\ifcase#1 %
```

Bézier segments for 1/8 circle. Let

$$a := \sqrt{1/2} \approx .707106781,$$
$$b := \tfrac{8}{3}\sqrt{2}\cos(\pi/8)\,(1-\cos(\pi/8)) \approx .2652164898,$$

$$c := \tfrac{1}{3}\left(-3 + 8\cos(\pi/8) - 2\cos^2(\pi/8)\right) \approx .8946431596,$$
$$d := \tfrac{1}{2}b(2 + 3\cos(\pi/8) - \cos^2(\pi/8)) \approx .5195704027.$$

(We have $\cos(\pi/8) = \frac{1}{2}\sqrt{2+\sqrt{2}}$.)

The fractions below are best possible rational approximations (obtained by continued fractions) to the following coordinates:

$(0,0)$, $(0,-b)$, $(1-c,-d)$, $(1-a,-a)$

```
    00%
    0{-173517671/654249180}%
    {65307479/619869377}{-34221476/65864945}%
    {225058681/768398401}{-543339720/768398401}%
\or
```

$(0,-a)$, $(a-d,-c)$, $(a-b,-1)$, $(a,-1)$

```
    0{-543339720/768398401}%
    {181455824/967576667}{-554561898/619869377}%
    {826676217/1870772527}{-1}%
    {543339720/768398401}{-1}%
\or
```

$(0,-1)$, $(b,-1)$, $(d,-c)$, $(a,-a)$

```
    0{-1}%
    {173517671/654249180}{-1}%
    {34221476/65864945}{-554561898/619869377}%
    {543339720/768398401}{-543339720/768398401}%
\or
```

$(0,-a)$, $(c-a,-d)$, $(1-a,-b)$, $(1-a,0)$

```
    0{-543339720/768398401}%
    {181455824/967576667}{-34221476/65864945}%
    {225058681/768398401}{-173517671/654249180}%
    {225058681/768398401}0%
\or
```

$(0,a)$, $(c-a,d)$, $(1-a,b)$, $(1-a,0)$

```
    0{543339720/768398401}%
    {181455824/967576667}{34221476/65864945}%
    {225058681/768398401}{173517671/654249180}%
    {225058681/768398401}0%
\or
```

$(0,1)$, $(b,1)$, (d,c), (a,a)

```
    01%
    {173517671/654249180}1%
    {34221476/65864945}{554561898/619869377}%
    {543339720/768398401}{543339720/768398401}%
\or
```

$(0,a)$, $(a-d,c)$, $(a-b,1)$, $(a,1)$

```
    0{543339720/768398401}%
    {181455824/967576667}{554561898/619869377}%
    {826676217/1870772527}1%
    {543339720/768398401}1%
\or
```

$(0,0)$, $(0,b)$, $(1-c,d)$, $(1-a,a)$

```
    00%
    0{173517671/654249180}%
```

```
    {65307479/619869377}{34221476/65864945}%
    {225058681/768398401}{543339720/768398401}%
  \fi}
```

`\xP@circhar@@@` Draw the arc of 1/8 circle and use the same space as the chars from the circle font do.

```
\newcommand\xP@circhar@@@[8]{%
  \xP@setsolidpat
  \xP@stroke{\xP@coor{\R@*#1}{\R@*#2}m
  \xP@coor{\R@*#3}{\R@*#4}\xP@coor{\R@*#5}{\R@*#6}%
  \xP@coor{\R@*#7}{\R@*#8}c}%
  \vrule width\z@ height\R@ depth\R@
  \kern\dimexpr\R@*#7\relax
}
```

`\cirrestrict@@` `\xP@cirrestrict@@` Basically, `\cirrestrict@@` is turned into a no-op and does not change the radius.

```
\xP@hook{pdf}{cirrestrict@@}
\newcommand*\xP@cirrestrict@@{\count@\z@\relax}
⟨/basic⟩
```

9.4 Rotation and scaling

```
⟨*rotate⟩
\xycatcodes
```

`\xypdf-ro@loaded`

```
\expandafter\let\csname xypdf-ro@loaded\endcsname\@empty
```

`\xyscale@@` `\xP@xyscale@@` Scale the box 0 to the factors in `#1` and `#2`.

```
\xP@hook{rotate}{xyscale@@}
\newcommand*\xP@xyscale@@[2]{%
  \setboxz@h{%
    \hskip\L@c
    \hskip-\R@p
    \lower\U@p\hbox{\xP@cm{#1}00{#2}%
      {\raise\U@p\hb@xt@\z@{\hskip-\L@c\hskip\R@p\boxz@\hss}}%
    }%
  }%
  \global\let\xP@lastpattern\@empty
}
```

`\xyRotate@@` `\xP@xyRotate@@` Rotation in the direction `#1`.

```
\xP@hook{rotate}{xyRotate@@}
\newcommand\xP@xyRotate@@{\xP@rotate@\xP@trigfromdir}
```

`\doSpecialRotate@@` `\xP@doSpecialRotate@@` Rotation by the angle in `#1`.

```
\xP@hook{rotate}{doSpecialRotate@@}
\@ifdefinable\xP@doSpecialRotate@@\relax
\def\xP@doSpecialRotate@@#1@@{\xP@rotate@\xP@trig{#1}}
```

`\xP@rotate@` Common code for both rotations: rotate the box 0.

```
\newcommand*\xP@rotate@[2]{%
  \setboxz@h{%
    #1{#2}%
    \hskip\L@c
    \hskip-\R@p
    \lower\U@p\hbox{\xP@cm\cosDirection\sinDirection
```

```
      {\if-\sinDirection\else-\sinDirection\fi}\cosDirection
      {\raise\U@p\hb@xt@\z@{\hskip-\L@c\hskip\R@p\boxz@\hss}}%
    }%
  }%
  \global\let\xP@lastpattern\@empty
}
```

`\xP@trig` Calculate sine and cosine from the angle in `#1`.

```
\newcommand*\xP@trig[1]{%
  \@tempdima\dimexpr#1pt\relax
```

Translate the argument into the interval $[0\mathrm{pt}, 360\mathrm{pt}]$.

```
  \@tempdimb\@tempdima
```

$23592960 = 360 \cdot 65536$

```
  \divide\@tempdimb23592960
  \advance\@tempdima-23592960\@tempdimb
  \ifdim\@tempdima<\z@\advance\@tempdima360pt\fi
  \@tempdimb\@tempdima
```

$5898240 = 90 \cdot 65536$

```
  \divide\@tempdimb5898240
```

It's enough to know sin between 0° and 90°. The cos and the values in the other quadrants can be derived from that.

```
  \ifcase\@tempdimb
    \xP@sinpoly
    \edef\sinDirection{\xP@EARPT\@tempdimb}%
    \@tempdima\dimexpr90pt-\@tempdima\relax
    \xP@sinpoly
    \edef\cosDirection{\xP@EARPT\@tempdimb}%
  \or
    \@tempdima\dimexpr180pt-\@tempdima\relax
    \xP@sinpoly
    \edef\sinDirection{\xP@EARPT\@tempdimb}%
    \@tempdima\dimexpr90pt-\@tempdima\relax
    \xP@sinpoly
    \edef\cosDirection{\xP@EARPT\dimexpr-\@tempdimb\relax}%
  \or
    \@tempdima\dimexpr\@tempdima-180pt\relax
    \xP@sinpoly
    \edef\sinDirection{\xP@EARPT\dimexpr-\@tempdimb\relax}%
    \@tempdima\dimexpr90pt-\@tempdima\relax
    \xP@sinpoly
    \edef\cosDirection{\xP@EARPT\dimexpr-\@tempdimb\relax}%
  \or
    \@tempdima\dimexpr360pt-\@tempdima\relax
    \xP@sinpoly
    \edef\sinDirection{\xP@EARPT\dimexpr-\@tempdimb\relax}%
    \@tempdima\dimexpr90pt-\@tempdima\relax
    \xP@sinpoly
    \edef\cosDirection{\xP@EARPT\@tempdimb}%
  \else
    \PackageError{xypdf}{Unexpected case in sin/cos calculation}%
      {Feel free to contact the author of the xypdf package with a minimal %
      example.}%
  \fi
}
```

`\xP@sinpoly` Polynomial approximation to the sine in the interval [0pt, 90pt]. The deviation should be ±1sp maximal (but no guarantee). (3rd order, 4 subintervals, exact values for 0pt and 90pt)

```
\newcommand*\xP@sinpoly{{%
  \ifdim\@tempdima<49pt
    \ifdim\@tempdima<27pt
      \@tempdimb\dimexpr((\@tempdima*-529771058/16039085-1384933sp)%
        *\@tempdima/268756075+10714164sp)*\@tempdima/613777813\relax
    \else
      \advance\@tempdima-27pt
      \@tempdimb\dimexpr(((\@tempdima*-743101305/20672414-238989613sp)%
        *\@tempdima/80975565+42661556sp)*\@tempdima/622461739+2\p@)%
        *157520747/693945047\relax
    \fi
  \else
    \ifdim\@tempdima<70pt
      \advance\@tempdima-49pt
      \@tempdimb\dimexpr(((\@tempdima*-348406699/107952940-55079229sp)%
        *\@tempdima/866635628+408805sp)*\@tempdima/26926757+\p@)%
        *135751711/179873976\relax
    \else
      \advance\@tempdima-70pt
      \@tempdimb\dimexpr(((\@tempdima*-1015850353/137849442-460519207sp)%
        *\@tempdima/8742349+142263941sp)*\@tempdima/972432199+23\p@)%
        *31253604/764969669\relax
    \fi
  \fi
  \global\dimen@i\@tempdimb
  }\@tempdimb\dimen@i
}
```

End of the section for XY-pic's "rotate" option. The macro `\xP@trigfromdir` below is also used for the `{-}` directional.

```
\xyendinput
⟨/rotate⟩
⟨*basic⟩
```

`\xP@trigfromdir` Calculate sine and cosine from the direction number in `#1`.

```
\newcommand*\xP@trigfromdir[1]{{%
  \Direction#1\relax
```

`\Direction` mod 2048

```
  \count@-\Direction
  \advance\count@4096
  \divide\count@2048
```

Assign the slope in the right way.

```
  \ifcase\count@
    \d@X\K@\p@
    \d@Y\numexpr\Direction-3*\K@\relax\p@
  \or
    \d@X\numexpr\Direction-\K@\relax\p@
    \d@Y-\K@\p@
  \or
    \d@X-\K@\p@
    \d@Y\numexpr-\Direction-\K@\relax\p@
  \or
    \d@X\numexpr-\Direction-3*\K@\relax\p@
```

```
    \d@Y\K@\p@
  \else
    \PackageError{xypdf}{Unexpected case in direction calculation}%
      {Feel free to contact the author of the xypdf package with a minimal %
      example.}%
  \fi
```

Bring the pair (`\d@X`, `\d@Y`) to norm 1.

```
  \xP@veclen
  \xdef\@gtempa{%
    \def\noexpand\cosDirection{\xP@EARPT\dimexpr\d@X*\p@/\@tempdimb\relax}%
    \def\noexpand\sinDirection{\xP@EARPT\dimexpr\d@Y*\p@/\@tempdimb\relax}%
  }%
  }\@gtempa
}
```

9.5 Temporary registers

`\xP@newdimen` Remove the `\outer` from `\newdimen`. (This applies for plain TEX.)

```
\outer\def\@tempa{\alloc@1\dimen\dimendef\insc@unt}
\let\xP@newdimen\newdimen
\ifx\newdimen\@tempa
  \def\xP@newdimen{\alloc@1\dimen\dimendef\insc@unt}
\fi
\outer\def\@tempa#1{\count@=\escapechar\escapechar=-1
    \expandafter\expandafter\expandafter
    \def\@if#1{true}{\let#1=\iftrue}%
    \expandafter\expandafter\expandafter
    \def\@if#1{false}{\let#1=\iffalse}%
    \@if#1{false}\escapechar=\count@}
\let\@tempa\relax
```

The next section is for the “curve” extension!

```
⟨/basic⟩
⟨*curve⟩
\xycatcodes
```

`\xypdf-cu@loaded`

```
\expandafter\let\csname xypdf-cu@loaded\endcsname\@empty
```

`\xP@tempvar` In order to save registers, xypdf shares XY-pic’s and LATEX dimension and counter registers but uses different, more descriptive names. Every macro that uses these temporary variables must be safely encapsulated in a group so that the registers are not changed from the outside scope!

The xypdf package uses several sets of temporary variable names for different modules. Since it is important that these assignments do not overlap and that the variables are only used encapsulated within groups, the macros which use temporary variables are marked by colored bullets •1, •2, •3, •4, •5, •6, •7 with one color for each set of variables.

The table in Figure 5 lists all variable assignments in these sets. It can be seen from the table which sets of variables can be used together. For example, set •1 consisting of `\xP@bigdim` can be used together with all other temporary variables, while •2 and •4 must never be used together.

```
\newcommand*\xP@tempvar[2]{%
  \@ifdefinable#1\relax
  \let#1#2%
}
```

XY-pic/LaTeX variable	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7
\quotPTK@	\xP@bigdim						
\L@p		\xP@parA	\xP@A		(\L@p)		(\L@p)
\U@p		\xP@velA	\xP@B		(\U@p)		(\U@p)
\R@p		\xP@parB	\xP@C		(\R@p)		(\R@p)
\D@p		\xP@velB	\xP@D		(\D@p)		(\D@p)
\X@origin		\xP@parC	\xP@E				\xP@temppar
\Y@origin		\xP@velC	\xP@F				\xP@tempvel
\X@xbase		\xP@parD	\xP@G				\xP@posX
\Y@xbase		\xP@velD	\xP@H				\xP@posY
\X@ybase		\xP@parE	\xP@I = \xP@a		\xP@a		\xP@oldpar
\Y@ybase		\xP@velE	\xP@J = \xP@b		\xP@b		\xP@lastpar
\X@min		\xP@lenA	\xP@K		\xP@c		\xP@tempvel@
\Y@min		\xP@lenB	\xP@L		\xP@valA		\xP@parinc
\X@max		\xP@partlen	\xP@fa		\xP@valB		
\Y@max		\xP@oldpartlen	\xP@fd		\xP@devA		
\almostz@		\xP@tolerance	\xP@tm		\xP@devB		\xP@squiglen
\K@dXdY			\xP@xm		\xP@ti		
\K@dYdX			\xP@ym		\xP@tip		
\splineval@			\xP@off		(\xP@off)		
\splinedepth@			\xP@ta				
\splinetol@			\xP@tb				
\splinelength@			\xP@tc				
\L@			\xP@M				
\normallineskiplimit			\xP@oldobj				
\@savsk				\xP@Tax		\xP@sa	
\lower@bound				\xP@Tay		\xP@sb	
\upper@bound				\xP@Tdx		\xP@sc	
\leftmargin				\xP@Tdy		\xP@Ab	
\rightmargin				\xP@Tmx		\xP@AAb	
\listparindent				\xP@Tmy		\xP@Aba	
\itemindent				\xP@xa	(\xP@xa)	\xP@Abb	
\labelwidth				\xP@ya	(\xP@ya)	\xP@Abc	
\labelsep				\xP@xb	(\xP@xb)	\xP@AAba	
\linewidth				\xP@yb	(\xP@yb)	\xP@AAbb	
\@totalleftmargin				\xP@xc	(\xP@xc)	\xP@AAbc	
\leftmargini				\xP@yc	(\xP@yc)	\xP@dta	
\leftmarginii				\xP@xd	(\xP@xd)	\xP@dtb	
\leftmarginiii				\xP@yd	(\xP@yd)	\xP@dtc	

Figure 5: Temporary dimension registers in xypdf.

\xP@oldornewdimen

Either reuse an existing dimension register if the name is defined or allocate a new dimension register. This makes it possible to selectively reuse LaTeX dimension registers or allocate new registers in plain TeX mode.

```
\newcommand*\xP@oldornewdimen[2]{%
  \@ifundefined{#2}%
    {\xP@newdimen#1}%
    {\expandafter\xP@tempvar\expandafter#1\csname#2\endcsname}%
}
```

\xP@bigdim

•1 A big constant less than $\frac{1}{3}$`\maxdimen` $\approx$ 5461pt and having many small prime factors.

```
\xP@tempvar\xP@bigdim\quotPTK@
```

\xP@parA \xP@velA \xP@parB \xP@velB \xP@parC \xP@velC \xP@parD \xP@velD \xP@parE \xP@velE \xP@lenA \xP@lenB \xP@partlen \xP@oldpartlen \xP@tolerance

•2 Second set of temporary variables: for the arc length algorithm.

```
\xP@tempvar\xP@parA\L@p
\xP@tempvar\xP@velA\U@p
\xP@tempvar\xP@parB\R@p
\xP@tempvar\xP@velB\D@p
\xP@tempvar\xP@parC\X@origin
\xP@tempvar\xP@velC\Y@origin
\xP@tempvar\xP@parD\X@xbase
\xP@tempvar\xP@velD\Y@xbase
\xP@tempvar\xP@parE\X@ybase
\xP@tempvar\xP@velE\Y@ybase
\xP@tempvar\xP@lenA\X@min
\xP@tempvar\xP@lenB\Y@min
\xP@tempvar\xP@partlen\X@max
\xP@tempvar\xP@oldpartlen\Y@max
\xP@tempvar\xP@tolerance\almostz@
```

\xP@A \xP@B \xP@C \xP@D \xP@E \xP@F \xP@G \xP@H \xP@I \xP@J \xP@K \xP@L \xP@fa \xP@fd \xP@tm \xP@xm \xP@ym \xP@off \xP@ta \xP@tb \xP@tc \xP@M

•3 Third set of temporary registers: Bézier offset algorithm ans solving linear equations.

```
\xP@tempvar\xP@A\L@p
\xP@tempvar\xP@B\U@p
\xP@tempvar\xP@C\R@p
\xP@tempvar\xP@D\D@p
\xP@tempvar\xP@E\X@origin
\xP@tempvar\xP@F\Y@origin
\xP@tempvar\xP@G\X@xbase
\xP@tempvar\xP@H\Y@xbase
\xP@tempvar\xP@I\X@ybase
\xP@tempvar\xP@J\Y@ybase
\xP@tempvar\xP@K\X@min
\xP@tempvar\xP@L\Y@min
\xP@tempvar\xP@fa\X@max
\xP@tempvar\xP@fd\Y@max
\xP@tempvar\xP@tm\almostz@
\xP@tempvar\xP@xm\K@dXdY
\xP@tempvar\xP@ym\K@dYdX
\xP@tempvar\xP@off\splineval@
\xP@tempvar\xP@ta\splinedepth@
\xP@tempvar\xP@tb\splinetol@
\xP@tempvar\xP@tc\splinelength@
\xP@tempvar\xP@M\L@
```

\xP@oldobj \xP@Tax \xP@Tay \xP@Tdx \xP@Tdy \xP@Tmx \xP@Tmy \xP@xa \xP@ya \xP@xb \xP@yb

•3 We need 15 more temporary registers. If possible, we use exisitng LaTeX dimension registers. If XY-pic is used in plain TeX mode, there are hopefully still free slots for dimension registers. We take them for the temporary variables but release them afterwards so that other packages can use them.

```
\@tempcnta\count11\relax
\xP@oldornewdimen\xP@oldobj{normallineskiplimit}
\xP@oldornewdimen\xP@Tax{@savsk}
```

•4

```
\xP@oldornewdimen\xP@Tay{lower@bound}
\xP@oldornewdimen\xP@Tdx{upper@bound}
\xP@oldornewdimen\xP@Tdy{leftmargin}
\xP@oldornewdimen\xP@Tmx{rightmargin}
\xP@oldornewdimen\xP@Tmy{listparindent}
\xP@oldornewdimen\xP@xa{itemindent}
\xP@oldornewdimen\xP@ya{labelwidth}
\xP@oldornewdimen\xP@xb{labelsep}
\xP@oldornewdimen\xP@yb{linewidth}
\xP@oldornewdimen\xP@xc{@totalleftmargin}
\xP@oldornewdimen\xP@yc{leftmargini}
\xP@oldornewdimen\xP@xd{leftmarginii}
\xP@oldornewdimen\xP@yd{leftmarginiii}
\count11\@tempcnta
```

\xP@a \xP@b \xP@c \xP@valA \xP@valB \xP@devA \xP@devB \xP@ti \xP@tip

•5 Fifth set of temporary variables: Parameters for drawing part of a spline segment.

```
\xP@tempvar\xP@a\X@ybase
\xP@tempvar\xP@b\Y@ybase
\xP@tempvar\xP@c\X@min
\xP@tempvar\xP@valA\Y@min
\xP@tempvar\xP@valB\X@max
\xP@tempvar\xP@devA\Y@max
\xP@tempvar\xP@devB\almostz@
\xP@tempvar\xP@ti\K@dXdY
\xP@tempvar\xP@tip\K@dYdX
```

\xP@sa \xP@sb \xP@sc \xP@Ab \xP@AAb \xP@Aba \xP@Abb \xP@Abc \xP@AAba \xP@AAbb \xP@AAbc \xP@dta \xP@dtb \xP@dtc

•6 Sixth set of temporary variables: Solving a linear system approximately.

```
\xP@tempvar\xP@sa\xP@Tax
\xP@tempvar\xP@sb\xP@Tay
\xP@tempvar\xP@sc\xP@Tdx
\xP@tempvar\xP@Ab\xP@Tdy
\xP@tempvar\xP@AAb\xP@Tmx
\xP@tempvar\xP@Aba\xP@Tmy
\xP@tempvar\xP@Abb\xP@xa
\xP@tempvar\xP@Abc\xP@ya
\xP@tempvar\xP@AAba\xP@xb
\xP@tempvar\xP@AAbb\xP@yb
\xP@tempvar\xP@AAbc\xP@xc
\xP@tempvar\xP@dta\xP@yc
\xP@tempvar\xP@dtb\xP@xd
\xP@tempvar\xP@dtc\xP@yd
```

\xP@temppar \xP@tempvel \xP@posX \xP@posY \xP@oldpar \xP@lastpar \xP@tempvel@ \xP@parinc \xP@squiglen

•7 Seventh set of temporary registers: For multiple dotted splines.

```
\xP@tempvar\xP@temppar\X@origin
\xP@tempvar\xP@tempvel\Y@origin
\xP@tempvar\xP@posX\X@xbase
\xP@tempvar\xP@posY\Y@xbase
\xP@tempvar\xP@oldpar\X@ybase
\xP@tempvar\xP@lastpar\Y@ybase
```

```
\xP@tempvar\xP@tempvel@\X@min
\xP@tempvar\xP@parinc\Y@min
\xP@tempvar\xP@squiglen\almostz@
```

\xP@scaleone \xP@scaletwo \xP@scalethree

We also use temporary numerical registers for scaling factors in `\xP@solvelinearsystem`.

```
\xP@tempvar\xP@scaleone\K@
\xP@tempvar\xP@scaletwo\KK@
\xP@tempvar\xP@scalethree\Direction
```

9.6 Bézier curves

\splinesolid@ \splinedashed@ \splinedotted@

These are the hooks for single-stroke splines (solid, dashed and dotted).

```
\xP@hook{curve}{splinesolid@}
\newcommand*\xP@splinesolid@{\xP@spline\xP@setsolidpat}
\xP@hook{curve}{splinedashed@}
\newcommand*\xP@splinedashed@{\xP@spline\xP@setdashpat}
\xP@hook{curve}{splinedotted@}
\newcommand*\xP@splinedotted@{\xP@spline\xP@setdottedpat}
```

\xP@spline

Output a spline segment. Parameter: Macro for the dash pattern generation.

```
\newcommand*\xP@spline[1]{%
  \readsplineparams@
```

Neglect splines which are drawn “backwards”. Somehow XY-pic draws curves forward and backward, but we need it to be drawn only once.

```
  \ifdim\dimen5<\dimen7
    \xP@preparespline
```

Neglect splines of length zero.

```
    \ifdim\@tempdimb>\z@
```

Set the dash pattern.

```
      #1%
```

Draw the spline.

```
      \xP@stroke{\xP@coor\X@p\Y@p m %
        \xP@coor\L@c\U@c\xP@coor\R@c\D@c\xP@coor\X@c\Y@c c}%
```

Record the end point for pattern continuation.

```
        \xP@savec
    \fi
  \fi
}
```

\xP@preparespline

```
\newcommand*\xP@preparespline{%
```

If we have a quadratic Bézier segment, convert it to a cubic one.

```
    \ifx\splineinfo@\squineinfo@
      \L@c\dimexpr(\X@p+2\A@)/3\relax
      \U@c\dimexpr(\Y@p+2\B@)/3\relax
      \R@c\dimexpr(\X@c+2\A@)/3\relax
      \D@c\dimexpr(\Y@c+2\B@)/3\relax
    \fi
```

Cut the spline according to that start and end parameters in `\dimen5` and `\dimen7`.

```
    \xP@shavespline
```

Determine the spline length (for the pattern generation; unnecessary for solid splines).

```
    \xP@bezierlength
}
```

`\xP@inibigdim` •1 Initialize `\xP@bigdim` every time a macro that uses this register is called. See e. g. `\xP@shaveprec`.

```
\newcommand*\xP@inibigdim{\xP@bigdim5040pt}
```

`\xP@shavespline` Shave a cubic spline at both ends at the parameter values in `\dimen5` and `\dimen7`. For normal use, the parameters fulfill 0pt ≤ `\dimen5` < `\dimen7` ≤ 1pt.

(Note that `\xP@bigdim` only occurs in the arguments to `\xP@shaveprec`, so this use is safe.)

```
\newcommand*\xP@shavespline{%
  \xP@shaveprec{\dimen5*\xP@bigdim/\p@}{\dimen7*\xP@bigdim/\p@}%
}
```

`\xP@shaveprec` •1 Shave a cubic spline at both ends at the parameter values in `#1` and `#2`. For normal use, the parameters fulfill 0pt ≤ `#1` < `#2` ≤ `\xP@bigdim`. The control points for the cubic Bézier curve are (`\X@p`,`\Y@p`), (`\L@c`,`\U@c`), (`\R@c`,`\D@c`), (`\X@c`,`\Y@c`). The XY-pic registers `\A@`, `\B@`, `\L@p`, `\U@p`, `\R@p`, `\D@p`, `\X@min` and `\Y@min` are used as temporary registers, but safely encapsulated in a group.

```
\newcommand*\xP@shaveprec[2]{{%
  \xP@inibigdim
  \A@\dimexpr#1\relax
  \B@\dimexpr#2\relax
```

Shortcut in case the spline is not changed.

```
  \@tempswatrue
  \ifdim\A@=\z@\ifdim\B@=\xP@bigdim\@tempswafalse\fi\fi
  \if@tempswa
    \L@p\dimexpr\L@c-\X@p\relax
    \U@p\dimexpr\R@c-\L@p-\L@c\relax
    \R@p\dimexpr\X@c-3\R@c+3\L@c-\X@p\relax
    \D@p\dimexpr\U@c-\Y@p\relax
    \X@min\dimexpr\D@c-\D@p-\U@c\relax
    \Y@min\dimexpr\Y@c-3\D@c+3\U@c-\Y@p\relax
    \xdef\@gtempa{%
      \X@p\the\dimexpr\X@p+(3\L@p+(3\U@p+\R@p*\A@/\xP@bigdim)%
        *\A@/\xP@bigdim)*\A@/\xP@bigdim\relax
      \Y@p\the\dimexpr\Y@p+(3\D@p+(3\X@min+\Y@min*\A@/\xP@bigdim)%
        *\A@/\xP@bigdim)*\A@/\xP@bigdim\relax
      \L@c\the\dimexpr\X@p+(2\A@+\B@)*\L@p/\xP@bigdim+((\A@+2\B@)%
        *\U@p/\xP@bigdim+\R@p*\A@/\xP@bigdim*\B@/\xP@bigdim)%
        *\A@/\xP@bigdim\relax
      \U@c\the\dimexpr\Y@p+(2\A@+\B@)*\D@p/\xP@bigdim+((\A@+2\B@)%
        *\X@min/\xP@bigdim+\Y@min*\A@/\xP@bigdim*\B@/\xP@bigdim)%
        *\A@/\xP@bigdim\relax
      \R@c\the\dimexpr\X@p+(2\B@+\A@)*\L@p/\xP@bigdim+((\B@+2\A@)%
        *\U@p/\xP@bigdim+\R@p*\B@/\xP@bigdim*\A@/\xP@bigdim)%
        *\B@/\xP@bigdim\relax
      \D@c\the\dimexpr\Y@p+(2\B@+\A@)*\D@p/\xP@bigdim+((\B@+2\A@)%
        *\X@min/\xP@bigdim+\Y@min*\B@/\xP@bigdim*\A@/\xP@bigdim)%
        *\B@/\xP@bigdim\relax
      \X@c\the\dimexpr\X@p+(3\L@p+(3\U@p+\R@p*\B@/\xP@bigdim)%
        *\B@/\xP@bigdim)*\B@/\xP@bigdim\relax
      \Y@c\the\dimexpr\Y@p+(3\D@p+(3\X@min+\Y@min*\B@/\xP@bigdim)%
```

```
        *\B@/\xP@bigdim)*\B@/\xP@bigdim\relax}%
  \else
    \global\let\@gtempa\relax
  \fi
  }\@gtempa
}
```

`\xP@bezierlength` •1 •2 Compute the arc length of a cubic Bézier segment.

The following algorithm is used: The velocity for a partial segment is fitted at three points (A-C-E) by a quadratic function, and the arc length is approximated by the integral over this quadratic function.

Each interval is recursively divided in halves (A-B-C, C-D-E) as long as the result for the arc length changes more than the precision parameter `\xP@tolerance`. If the desired precision is reached, the arc length in the small interval is added to the total arc length, and the next interval is considered.

The result goes into `\@tempdimb`.

```
\newcommand*\xP@bezierlength{{%
  \xP@inibigdim
  \@tempdimb\z@
  \xP@parA\z@
  \xP@velocity\z@\xP@velA
  \xP@parC.5\xP@bigdim
  \xP@velocity\xP@parC\xP@velC
  \xP@velocity\xP@bigdim\xP@velE
```

Arc length (integral over the quadratic approximation)

```
  \xP@oldpartlen\dimexpr(\xP@velA+4\xP@velC+\xP@velE)/6\relax
```

Tolerance parameter: It is set to 1/100000 of the approximate arc length, but at least 1sp.

```
  \xP@tolerance\xP@max{1sp}{\dimexpr\xP@oldpartlen/100000\relax}%
```

Initiate the recursive algorithm with the interval $[0, 1]$.

```
  \xP@arclength\xP@parC\xP@velC\xP@bigdim\xP@velE\xP@oldpartlen
```

Pass the result to outside the group.

```
  \global\dimen@i\@tempdimb
  }\@tempdimb\dimen@i
}
```

`\xP@velocity` •1 Compute the velocity at the point `#1` on a cubic Bézier curve. Needs: Bézier control points `\X@p`,...,`\Y@c`. Parameter `#2`: dimension register for the result. Temporary: `\L@p`, `\U@p`, `\d@X`, `\d@Y`.

```
\newcommand*\xP@velocity[2]{{%
  \@tempdima\dimexpr#1\relax
  \xP@tangent
  \global\dimen@i\@tempdimb
  }#2\dimen@i
}
```

`\xP@tangent` •1

```
\newcommand*\xP@tangent{%
  \d@X3\xP@precbeziertan\X@p\L@c\R@c\X@c\@tempdima
  \d@Y3\xP@precbeziertan\Y@p\U@c\D@c\Y@c\@tempdima
  \xP@veclen
}
```

`\xP@tangentvec` •1 Tangent vector on a Bézier curve. Parameter `#1`: Parameter on the segment. Needs: Bézier parameters `\X@p`,...,`\Y@c`. Returns: vector in (`\d@X`, `\d@Y`), norm in `\@tempdimb`.

```
\newcommand*\xP@tangentvec[1]{{%
    \@tempdima#1\relax
    \xP@tangent
```

If the velocity is zero at some point, take the second derivative for the tangent vector.

```
    \ifdim\@tempdimb=\z@
      \L@p\dimexpr\X@c-\X@p+(\L@c-\R@c)*3\relax
      \U@p\dimexpr\Y@c-\Y@p+(\U@c-\D@c)*3\relax
      \d@X\dimexpr\L@p*\@tempdima/\xP@bigdim+(\X@p-2\L@c+\R@c)\relax
      \d@Y\dimexpr\U@p*\@tempdima/\xP@bigdim+(\Y@p-2\U@c+\D@c)\relax
      \xP@veclen
```

Or even the third derivative.

```
      \ifdim\@tempdimb=\z@
        \d@X\L@p
        \d@Y\U@p
        \xP@veclen
        \ifdim\@tempdimb=\z@
          \xP@warning{xypdf}{Cannot determine a tangent vector to a curve}%
          \@tempdimb\p@
        \fi
      \fi
    \fi
    \global\dimen@i\d@X
    \global\dimen3\d@Y
    \global\dimen5\@tempdimb
  }%
  \d@X\dimen@i
  \d@Y\dimen3\relax
  \@tempdimb\dimen5\relax
}
```

`\xP@arclength` •2 The recursive step for the arc length computation.
Needs: `\xP@tolerance`, `\xP@parA`, `\xP@velA`. Parameter: `#1` is the middle parameter, `#2` the velocity at `#1`, `#3` the third parameter, `#4` the velocity at `#3`, `#5` the approximate arc length in the interval from `\xP@parA` to `#3`.

```
\newcommand*\xP@arclength[5]{%
  \xP@parE#3%
  \xP@velE#4%
  \xP@parC#1%
  \xP@velC#2%
  \xP@oldpartlen#5%
```

Compute two more pairs (parameter, velocity) at positions $\frac{1}{4}$ and $\frac{3}{4}$ of the interval.

```
  \xP@parB\dimexpr(\xP@parC+\xP@parA)/2\relax
  \xP@velocity\xP@parB\xP@velB
  \xP@parD\dimexpr(\xP@parE+\xP@parC)/2\relax
  \xP@velocity\xP@parD\xP@velD
```

Compute the approximations for the arc length on the two smaller parameter intervals (A-B-C) and (C-D-E).

```
  \xP@lenA
    \dimexpr(\xP@velA+4\xP@velB+\xP@velC)/6*(\xP@parC-\xP@parA)/\xP@bigdim\relax
  \xP@lenB
    \dimexpr(\xP@velC+4\xP@velD+\xP@velE)/6*(\xP@parE-\xP@parC)/\xP@bigdim\relax
  \xP@partlen\dimexpr\xP@lenA+\xP@lenB\relax
```

Check whether the approximation for the arc length has changed more than the precision parameter. The code is a hack to compare the absolute value without occupying another dimension register.

```
{\@tempdima\dimexpr\xP@oldpartlen-\xP@partlen\relax
\expandafter}\ifdim\ifdim\@tempdima<\z@-\fi\@tempdima>\xP@tolerance
```

Yes? Subdivide the interval. The input queue serves as a LIFO stack here!

```
  \edef\next@{%
    \noexpand\xP@arclength\xP@parB\xP@velB\xP@parC\xP@velC\xP@lenA
    \noexpand\xP@arclength{\the\xP@parD}{\the\xP@velD}{\the\xP@parE}%
    {\the\xP@velE}{\the\xP@lenB}%
  }%
\else
```

No? Proceed to the next parameter interval.

```
  \xP@parA\xP@parE
  \xP@velA\xP@velE
  \advance\@tempdimb\xP@partlen
  \DN@{}%
\fi
\next@
}
```

9.7 New improved curve styles

\@crv@ \xP@@crv@ Extend the list of curve styles for which special routines exist. New curve styles: `3{.}`, `{~}`, `2{~}`, `3{~}`, `{~~}`, `2{~~}`, `3{~~}`

```
\xP@hook{curve}{@crv@}
\newcommand*\xP@@crv@[2]{\DN@{#1#2}%
\ifx\next@\@empty \edef\next@{\crv@defaultshape}%
\ifx\bstartPLACE@\@empty\xdef\crvSTYLE@@{{\crv@defaultshape}}\fi
\else
\ifx\bstartPLACE@\@empty\gdef\crvSTYLE@@{#1{#2}}\fi
\fi
\ifx\next@\@empty\crv@noobject \DN@{\crv@{}{\xy@@crvaddstack@}}%
\else\def\tmp@{-}\ifx\next@\tmp@ \DN@{\crv@{}{\xy@@crvaddstack@}}%
\else\def\tmp@{=}\ifx\next@\tmp@
\DN@{\expandafter\crv@\crv@normaltemplate{\dir{=}}}%
\else\def\tmp@{2-}\ifx\next@\tmp@
\DN@{\expandafter\crv@\crv@normaltemplate{\dir{2.}}}%
\else\def\tmp@{3-}\ifx\next@\tmp@
\DN@{\expandafter\crv@\crv@normaltemplate{\dir{3.}}}%
\else\def\tmp@{--}\ifx\next@\tmp@
\DN@{\expandafter\crv@\crv@specialtemplate@{--}}%
\else\def\tmp@{==}\ifx\next@\tmp@
\DN@{\expandafter\crv@\crv@normaltemplate{\dir2{--}}}%
\else\def\tmp@{2--}\ifx\next@\tmp@
\DN@{\expandafter\crv@\crv@normaltemplate{\dir2{--}}}%
\else\def\tmp@{3--}\ifx\next@\tmp@
\DN@{\expandafter\crv@\crv@normaltemplate{\dir3{--}}}%
\else\def\tmp@{.}\ifx\next@\tmp@
\DN@{\expandafter\crv@\crv@specialtemplate@{.}}%
\else\def\tmp@{:}\ifx\next@\tmp@
\DN@{\expandafter\crv@\crv@normaltemplate{\dir{:}}}%
\else\def\tmp@{2.}\ifx\next@\tmp@
\DN@{\expandafter\crv@\crv@normaltemplate{\dir{:}}}%
```

```
  \else\def\tmp@{3.}\ifx\next@\tmp@
  \DN@{\expandafter\crv@\crv@normaltemplate{\dir3{.}}}%
  \else\def\tmp@{~}\ifx\next@\tmp@
  \DN@{\expandafter\crv@\crv@normaltemplate{\dir{~}}}%
  \else\def\tmp@{2~}\ifx\next@\tmp@
  \DN@{\expandafter\crv@\crv@normaltemplate{\dir2{~}}}%
  \else\def\tmp@{3~}\ifx\next@\tmp@
  \DN@{\expandafter\crv@\crv@normaltemplate{\dir3{~}}}%
  \else\def\tmp@{~~}\ifx\next@\tmp@
  \DN@{\expandafter\crv@\crv@normaltemplate{\dir{~~}}}%
  \else\def\tmp@{2~~}\ifx\next@\tmp@
  \DN@{\expandafter\crv@\crv@normaltemplate{\dir2{~~}}}%
  \else\def\tmp@{3~~}\ifx\next@\tmp@
  \DN@{\expandafter\crv@\crv@normaltemplate{\dir3{~~}}}%
  \else\def\tmp@{..}\ifx\next@\tmp@
  \DN@{\expandafter\crv@\crv@specialtemplate@{.}}%
  \else
  \DN@{\expandafter\crv@\crv@othertemplate{\dir#1{#2}}}%
  \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\next@}
```

`\xysplinespecialcases@`
`\xP@xysplinespecialcases@`

New: `\dir3{.}`, `\dir2{--}`, `\dir3{--}`, `\dir{~}`, `\dir2{~}`, `\dir3{~}`, `\dir{~~}`, `\dir2{~~}`, `\dir3{~~}`

```
\xP@hook{curve}{xysplinespecialcases@}
\newcommand*\xP@xysplinespecialcases@{%
  \ifx\@empty\xycrvdrop@
  \ifx\@empty\xycrvconn@\DN@{\splinesolid@}%
  \else\DN@{ \dir{-}}\ifx\next@\xycrvconn@\DN@{\splinesolid@}%
  \else\DN@{ \dir2{-}}\ifx\next@\xycrvconn@\DN@{\splinedoubled@}%
  \else\DN@{ \dir{=}}\ifx\next@\xycrvconn@\DN@{\splineribboned@}%
  \else\DN@{ \dir{2.}}\ifx\next@\xycrvconn@\DN@{\splinedoubled@}%
  \else\DN@{ \dir3{-}}\ifx\next@\xycrvconn@\DN@{\splinetrebled@}%
  \else\DN@{ \dir{3.}}\ifx\next@\xycrvconn@\DN@{\splinetrebled@}%
  \else\DN@{ \dir{--}}\ifx\next@\xycrvconn@\DN@{\splinedashed@}%
  \else\DN@{ \dir{.}}\ifx\next@\xycrvconn@\DN@{\splinedotted@}%
  \else\DN@{ \dir{:}}\ifx\next@\xycrvconn@\DN@{\splinedbldotted@}%
```

The next line does not occur in XY-pic for an unknown reason. However, it seems reasonable to define the special pattern `\dir2{.}` in the same way as for straight lines.

```
  \else\DN@{ \dir2{.}}\ifx\next@\xycrvconn@\DN@{\splinedbldotted@}%
  \else\DN@{ \dir3{.}}\ifx\next@\xycrvconn@\DN@{\xP@splinetrbldotted}%
  \else\DN@{ \dir2{--}}\ifx\next@\xycrvconn@\DN@{\xP@splinedbldashed}%
  \else\DN@{ \dir3{--}}\ifx\next@\xycrvconn@\DN@{\xP@splinetrbldashed}%
  \else\DN@{ \dir{~}}\ifx\next@\xycrvconn@\DN@{\xP@splinesquiggled}%
  \else\DN@{ \dir2{~}}\ifx\next@\xycrvconn@\DN@{\xP@splinedblsquiggled}%
  \else\DN@{ \dir3{~}}\ifx\next@\xycrvconn@\DN@{\xP@splinetrblsquiggled}%
  \else\DN@{ \dir{~~}}\ifx\next@\xycrvconn@\DN@{\xP@splinebrokensquiggled}%
  \else\DN@{ \dir2{~~}}\ifx\next@\xycrvconn@\DN@{\xP@splinebrokendblsquiggled}%
  \else\DN@{ \dir3{~~}}\ifx\next@\xycrvconn@\DN@{\xP@splinebrokentrblsquiggled}%
  \else\ifdim\splinetol@>\z@\else\splinedefaulttol@\fi
  \DN@{\splineset@@}\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
  \else
  \DN@{\splineset@@}%
  \fi
  \ifInvisible@\DN@{}\fi
  \next@
}
```

9.8 Multiple solid curves

\xP@splinedoubled@

```
\xP@hook{curve}{splinedoubled@}
\newcommand*\xP@splinedoubled@{%
  \xP@checkspline\xP@splinemultsolid\xP@doublestroke}
```

\xP@splineribboned@

```
\xP@hook{curve}{splineribboned@}
\@ifdefinable\xP@splineribboned@\relax
\let\xP@splineribboned@\xP@splinedoubled@
```

\xP@splinetrebled@

```
\xP@hook{curve}{splinetrebled@}
\newcommand*\xP@splinetrebled@{%
  \xP@checkspline\xP@splinemultsolid\xP@trblstroke}
```

\xP@doublestroke Offset parameters for double lines and curves

```
\newcommand*\xP@doublestroke{\xydashh@/2,-\xydashh@/2}
```

\xP@trblstroke Offset parameters for treble lines and curves

```
\newcommand*\xP@trblstroke{\xydashh@,\z@,-\xydashh@}
```

\xP@checkspline Get and check spline parameters before the macro in `#1` is executed.

```
\newcommand*\xP@checkspline[1]{%
  \readsplineparams@
```

Neglect splines which are drawn “backwards”. Somehow XY-pic draws curves forward and backward, but we need it to be drawn only once.

```
  \let\next@\@gobble
  \ifdim\dimen5<\dimen7
    \xP@preparespline
```

Neglect splines of zero length.

```
    \ifdim\@tempdimb>\z@
```

If the path length is less than twice the line width, just draw a solid path.

```
      \ifdim\@tempdimb<2\dimexpr\xP@preclw\relax
        \let\next@\xP@splinemultsolid
      \else
        \let\next@#1%
      \fi
    \fi
  \fi
  \next@
}
```

\xP@splinemultsolid •1

```
\newcommand*\xP@splinemultsolid[1]{{%
  \xP@inibigdim
  \@temptokena{}%
  \xP@setsolidpat
```

The `\@for` loop does the multiple strokes. `\@tempa` records the respective offset distance.

```
  \@for\@tempa:={#1}\do{\xP@paintsolid\z@\xP@bigdim}%
  \xP@stroke{\the\@temptokena}%
}}
```

`\xP@paintsolid` •1 •5 Draw a solid spline in the parameter interval [`#1`, `#2`] $\subseteq$ [0pt, `\xP@bigdim`] with a certain offset. The offset distance is expected in `\@tempa`.

```
\newcommand*\xP@paintsolid[2]{{%
```

Record the original anchor points.

```
  \xP@savepts
  \xP@a#1\relax
  \xP@c#2\relax
  \xP@movetotrue
  \xP@paintsolid@
  \xdef\@gtempa{\the\@temptokena}%
  }%
  \@temptokena\expandafter{\@gtempa}%
}
```

`\xP@paintsolid@` •1 •5

```
\newcommand*\xP@paintsolid@{%
```

These parameters record which part of the spline is currently being offset. They are varied as the spline may be subdivided for a precise offset curve.

```
  \xP@b\xP@c
```

Offset distance

```
  \xP@off\dimexpr\@tempa\relax
  \ifdim\xP@off=\z@
    \xP@shaveprec\xP@a\xP@c
  \else
    \loop
```

Restore the original anchor points.

```
      \xP@restorepts
```

Compute the approximate offset curve. Note that `\xP@a` and `\xP@b` contain the boundary parameters for the partial spline.

```
      \xP@offsetsegment
```

Test if the offset curve is good enough.

```
      \xP@testoffset
```

If not, shorten the parameter interval by 30%.

```
    \unless\ifxP@offsetok
      \xP@b\dimexpr\xP@a+(\xP@b-\xP@a)*7/10\relax
    \repeat
  \fi
```

Append the new segment to the path.

```
  \xP@append\@temptokena{\ifxP@moveto\xP@coor\X@p\Y@p m \fi
    \xP@coor\L@c\U@c\xP@coor\R@c\D@c\xP@coor\X@c\Y@c c }%
  \xP@movetofalse
```

Test if the end of the spline has been reached. If not, offset the rest of the curve.

```
  \ifdim\xP@b<\xP@c\relax
    \xP@a\xP@b
    \expandafter\xP@paintsolid@
  \fi
}
```

`\ifxP@moveto` We need a PDF `moveto` operator only for the first partial segment. Additional segments connect seamlessly.

```
\@ifdefinable\ifxP@moveto\relax
\@ifdefinable\xP@movetotrue\relax
\@ifdefinable\xP@movetofalse\relax
\newif\ifxP@moveto
```

`\xP@savepts` •5 Save the anchor points to the second set of reserved variables.

```
\newcommand*\xP@savepts{%
  \xP@xa\X@p
  \xP@ya\Y@p
  \xP@xb\L@c
  \xP@yb\U@c
  \xP@xc\R@c
  \xP@yc\D@c
  \xP@xd\X@c
  \xP@yd\Y@c
}
```

`\xP@restorepts` •5 Restore the anchor points from the second set of reserved variables.

```
\newcommand*\xP@restorepts{%
  \X@p\xP@xa
  \Y@p\xP@ya
  \L@c\xP@xb
  \U@c\xP@yb
  \R@c\xP@xc
  \D@c\xP@yc
  \X@c\xP@xd
  \Y@c\xP@yd
}
```

9.9 A Bézier curve offset algorithm

First, all control points are offset by the desired distance and in the direction of the normal vectors at the boundary points of the curve. We then adjust the distance of the inner two control points to the boundary control points along the tangents at the boundary points: $x_b = x_a + f_a T_{ax}$, $x_c = x_d + f_d T_{dx}$, and likewise for the y-coordinates. In nondegenerate cases, we have $T_{ax} = x_b - x_a$ and $T_{dx} = x_c - x_d$.

Let $P(a,b,c,d,t)$ denote the Bézier polynomial $a(1-t)^3 + 3bt(1-t)^2 + 3ct^2(1-t) + dt^3$. In order to determine the factors f_a and f_d, we set up a system of three equations.

- Two equations: The old point at parameter $\frac{1}{2}$ plus offset, (x_m, y_m), is the new point at parameter t_m.

$$\begin{aligned} x_m &= P(x_a, x_a + f_a T_{ax}, x_d + f_d T_{dx}, x_d, t_m) \\ y_m &= P(y_a, y_a + f_a T_{ay}, y_d + f_d T_{dy}, y_d, t_m) \end{aligned}$$

- Third equation: The old tangent at parameter $\frac{1}{2}$ is in the same direction as the new tangent at t_m.

$$\begin{aligned} &\tfrac{\partial}{\partial t_m} P(x_a, x_a + f_a T_{ax}, x_d + f_d T_{dx}, x_d, t_m) \cdot T_{my} \\ = &\tfrac{\partial}{\partial t_m} P(y_a, y_a + f_a T_{ay}, y_d + f_d T_{dy}, y_d, t_m) \cdot T_{mx} \end{aligned}$$

Up to a scalar factor of $-3/4$, (T_{mx}, T_{my}) is the velocity vector to the original curve at parameter $\frac{1}{2}$. We have $T_{mx} = (X_a + X_b - X_c - X_d)/2$ (in the old coordinates!) and T_{my} analogously. The system above is a nonlinear system of three equations in three variables, which we solve by Newton's method. Let f_a, f_d, and t_m be approximate solutions, and denote by Δf_a, Δf_d, and Δt_m the increments to the next approximation. In the first order, the three equations become:

$$\begin{aligned}
x_m &= P(x_a, x_b, x_c, x_d, t_m) + \Delta f_a \cdot T_{ax} \cdot 3t_m(1-t_m)^2 + \Delta f_d \cdot T_{dx} \cdot 3t_m^2(1-t_m) \\
&\qquad + \Delta t_m \tfrac{\partial}{\partial t_m} P(x_a, x_b, x_c, x_d, t_m) \\
y_m &= P(y_a, y_b, y_c, y_d, t_m) + \Delta f_a \cdot T_{ay} \cdot 3t_m(1-t_m)^2 + \Delta f_d \cdot T_{dy} \cdot 3t_m^2(1-t_m) \\
&\qquad + \Delta t_m \tfrac{\partial}{\partial t_m} P(y_a, y_b, y_c, y_d, t_m)
\end{aligned}$$

$$\begin{aligned}
&\Big(\tfrac{\partial}{\partial t_m} P(x_a, x_b, x_c, x_d, t_m) + \Delta f_a \cdot T_{ax} \cdot 3(1 - 4t_m + 3t_m^2) + \Delta f_d \cdot T_{dx} \cdot 3(2t_m - 3t_m^2) \\
&\qquad + \Delta t_m \cdot 6\left((x_a - 2x_b + x_c) + t_m(x_d - x_a + 3(x_b - x_c))\right)\Big) \cdot T_{my} \\
&= \Big(\tfrac{\partial}{\partial t_m} P(y_a, y_b, y_c, y_d, t_m) + \Delta f_a \cdot T_{ay} \cdot 3(1 - 4t_m + 3t_m^2) + \Delta f_d \cdot T_{dy} \cdot 3(2t_m - 3t_m^2) \\
&\qquad + \Delta t_m \cdot 6\left((y_a - 2y_b + y_c) + t_m(y_d - y_a + 3(y_b - y_c))\right)\Big) \cdot T_{mx}
\end{aligned}$$

Rewrite the equations so that they resemble the TeX code.

$$\begin{aligned}
8P(x_a, x_b, x_c, x_d, t_m) - 8x_m &= -\Delta f_a \cdot 3T_{ax} \cdot 2t_m \cdot (2(1-t_m))^2 \\
&\quad - \Delta f_d \cdot 3T_{dx} \cdot 4t_m^2 \cdot 2(1-t_m) - \Delta t_m \cdot 8\tfrac{\partial}{\partial t_m} P(x_a, x_b, x_c, x_d, t_m) \\
8P(y_a, y_b, y_c, y_d, t_m) - 8y_m &= -\Delta f_a \cdot 3T_{ay} \cdot 2t_m \cdot (2(1-t_m))^2 \\
&\quad - \Delta f_d \cdot 3T_{dy} \cdot 4t_m^2 \cdot 2(1-t_m) - \Delta t_m \cdot 8\tfrac{\partial}{\partial t_m} P(y_a, y_b, y_c, y_d, t_m) \\
T_{mx} \cdot 8\tfrac{\partial}{\partial t_m} P(y_a, y_b, y_c, y_d, t_m) &- T_{my} \cdot 8\tfrac{\partial}{\partial t_m} P(x_a, x_b, x_c, x_d, t_m) \\
&= -\Delta f_a \cdot (3T_{ay} \cdot 2T_{mx} - 3T_{ax} \cdot 2T_{my}) \cdot 2(1-3t_m) \cdot 2(1-t_m) \\
&\quad - \Delta f_d \cdot (3T_{dy} \cdot 2T_{mx} - 3T_{dx} \cdot 2T_{my}) \cdot 2(2-3t_m) \cdot 2t_m \\
-\Delta t_m \cdot (((y_d - y_a + 3(y_b - y_c)) &\cdot 2t_m + 2(y_a - 2y_b + y_c)) \cdot 3 \cdot 8T_{mx} \\
&- ((x_d - x_a + 3(x_b - x_c)) \cdot 2t_m + 2(x_a - 2x_b + x_c)) \cdot 3 \cdot 8T_{my})
\end{aligned}$$

Substitute $2t_m = \tau_m$.

$$\begin{aligned}
8P(x_a, x_b, x_c, x_d, t_m) - 8x_m &= -\Delta f_a \cdot 3T_{ax} \cdot \tau_m(2-\tau_m)^2 \\
&\quad - \Delta f_d \cdot 3T_{dx} \cdot \tau_m^2(2-\tau_m) - \tfrac{1}{2}\Delta\tau_m \cdot 8\tfrac{\partial}{\partial t_m} P(x_a, x_b, x_c, x_d, t_m) \\
8P(y_a, y_b, y_c, y_d, t_m) - 8y_m &= -\Delta f_a \cdot 3T_{ay} \cdot \tau_m \cdot (2-\tau_m)^2 \\
&\quad - \Delta f_d \cdot 3T_{dy} \cdot \tau_m^2(2-\tau_m) - \tfrac{1}{2}\Delta\tau_m \cdot 8\tfrac{\partial}{\partial t_m} P(y_a, y_b, y_c, y_d, t_m) \\
T_{mx} \cdot 8\tfrac{\partial}{\partial t_m} P(y_a, y_b, y_c, y_d, t_m) &- T_{my} \cdot 8\tfrac{\partial}{\partial t_m} P(x_a, x_b, x_c, x_d, t_m) \\
&= -\Delta f_a \cdot (3T_{ay} \cdot 2T_{mx} - 3T_{ax} \cdot 2T_{my}) \cdot (2-3\tau_m)(2-\tau_m) \\
&\quad - \Delta f_d \cdot (3T_{dy} \cdot 2T_{mx} - 3T_{dx} \cdot 2T_{my}) \cdot (4-3\tau_m)\tau_m \\
-\Delta\tau_m \cdot (((y_d - y_a + 3(y_b - y_c)) &\cdot \tau_m + 2(y_a - 2y_b + y_c)) \cdot 12T_{mx} \\
&- ((x_d - x_a + 3(x_b - x_c)) \cdot \tau_m + 2(x_a - 2x_b + x_c)) \cdot 12T_{my})
\end{aligned}$$

The translation into TeX dimensions:

- f_a = `\xP@fa`, f_d = `\xP@fd`
- τ_m = `\xP@tm`
- x_a = `\xP@xa`, …, x_d = `\xP@xd`, …, y_d = `\xP@yd`
- $8P(x_1, x_2, x_3, x_4, \frac{1}{2}x_5)$ = `\xP@bezierpoly#1#2#3#4#5`
- $8x_m$ = `\xP@xm`, $8y_m$ = `\xP@ym`
- $3T_{ax}$ = `\xP@Tax`, $3T_{dx}$ = `\xP@Tdx`, $3T_{ay}$ = `\xP@Tay`, $3T_{dy}$ = `\xP@Tdy`
- $8\frac{\partial}{\partial x_5}P(x_1, x_2, x_3, x_4, \frac{1}{2}x_5)$ = `\xP@beziertan#1#2#3#4#5`

Temporary:

- $2 - \tau_m$ = `\xP@ta`
- $\tau_m(2 - \tau_m)$ = `\xP@tb`
- T_{mx} = `\xP@Tmx`, T_{my} = `\xP@Tmy`
- $2 - 3\tau_m$ = `\xP@tb`
- $4 - 3\tau_m$ = `\xP@tc`

Since the linear system above tends to be singular or ill-conditioned (think about the frequent case when all control points are nearly collinear!), the Gauss algorithm `\xP@solvelinearsystem` does not always return a valid solution. In these cases, the system is not solved exactly but approximated iteratively in `\xP@applinsys`.

`\xP@tmx`
`\xP@tmy`

```
\@ifdefinable\xP@tmx\relax
\@ifdefinable\xP@tmy\relax
```

`\xP@Tmxy`
`\xP@Tmyx`

•4

```
\newcommand*\xP@Tmxy{*\xP@Tmx/\xP@Tmy}
\newcommand*\xP@Tmyx{*\xP@Tmy/\xP@Tmx}
```

`\xP@Tmzero`

```
\newcommand*\xP@Tmzero{*\z@}
```

`\xP@offsetsegment`

•1 •3 •4 Offset a cubic segment. The offset distance is given in `\xP@off`. The anchor points are given in `\X@p`,…,`\Y@c`. The partial spline in the parameter interval [`\xP@a`, `\xP@b`] $\subseteq$ [0pt, `\xP@bigdim`] is offset. The new Bézier curve is returned in `\xP@xa`,…,`\xP@yd`.

```
\newcommand*\xP@offsetsegment{{%
```

New first anchor point and tangent vector at 0

```
  \xP@tangentvec\xP@a
  \xP@xa\dimexpr\xP@precbezierpoly\X@p\L@c\R@c\X@c\xP@a/8%
    +\d@Y*\xP@off/\@tempdimb\relax
  \xP@ya\dimexpr\xP@precbezierpoly\Y@p\U@c\D@c\Y@c\xP@a/8%
    -\d@X*\xP@off/\@tempdimb\relax
  \xP@scaleT
  \xP@Tax\d@X
  \xP@Tay\d@Y
  \xP@E\@tempdimb
```

New last anchor point and tangent vector at 1

```
\xP@tangentvec\xP@b
\xP@xd\dimexpr\xP@precbezierpoly\X@p\L@c\R@c\X@c\xP@b/8%
  +\d@Y*\xP@off/\@tempdimb\relax
\xP@yd\dimexpr\xP@precbezierpoly\Y@p\U@c\D@c\Y@c\xP@b/8%
  -\d@X*\xP@off/\@tempdimb\relax
\xP@scaleT
\xP@Tdx-\d@X
\xP@Tdy-\d@Y
\xP@F\@tempdimb
```

Scalar product of the tangent vectors

```
\xP@M\z@
\xP@Max\xP@M\xP@Tdx
\xP@Max\xP@M\xP@Tdy
\xP@L\dimexpr\xP@Tax*\xP@Tdx/\xP@M+\xP@Tay*\xP@Tdy/\xP@M\relax
\xP@tm\dimexpr(\xP@a+\xP@b)/2\relax
\ifdim\xP@L>\dimexpr\xP@E*\xP@F/\xP@M*49/50\relax
```

Trick to improve the offset algorithm near sharp bends and cusps: If the tangent vectors (T_{ax}, T_{ay}) and (T_{dx}, T_{dy}) point nearly in the same direction, we do not use the true tangent vector for (T_{mx}, T_{my}) at the middle point but a fake one. (The exact condition is that their normed scalar product is greater that 49/50. For a straight line, the vectors would point in opposite directions.) The fake tangent vector is defined to be $(T_{ax} + T_{dx}, T_{ay} + T_{dy})$ rotated by $\pm 90°$. Its direction is chosen such that the scalar product with $(X_d - X_a, Y_d - Y_a)$ is nonnegative. (Use $(X_c - X_b, Y_c - Y_b)$ in the degenerate case $(X_d - X_a, Y_d - Y_a) = (0, 0)$.)

Rationale: In the presence of a sharp bend or cusp, the offset algorithm will hardly meet the tip. Since the tangent/normal at the tip is needed for a good offset curve, we provide this artificially.

```
  \d@X-\dimexpr\xP@Tay+\xP@Tdy\relax
  \d@Y\dimexpr\xP@Tax+\xP@Tdx\relax
  \xP@veclen
  \xP@A\dimexpr\X@c-\X@p\relax
  \xP@B\dimexpr\Y@c-\Y@p\relax
  \xP@M\z@
  \xP@Max\xP@M\xP@A
  \xP@Max\xP@M\xP@B
  \ifdim\xP@M=\z@
    \xP@A\dimexpr\R@c-\L@c\relax
    \xP@B\dimexpr\D@c-\U@c\relax
    \xP@Max\xP@M\xP@A
    \xP@Max\xP@M\xP@B
  \fi
  \xP@M\dimexpr\d@X*\xP@A/\xP@M+\d@Y*\xP@B/\xP@M\relax
  \ifdim\xP@M<\z@
    \multiply\d@X\m@ne
    \multiply\d@Y\m@ne
  \fi
\else
```

Normal case: tangent vector at the middle point.

```
  \xP@tangentvec\xP@tm
\fi
```

From here on, `\xP@a` and `\xP@b` will not be used any more, so these variables can be used under their other names `\xP@I`, `\xP@J` for the linear systems below.

8 times (middle point plus offset)

```
\xP@xm\dimexpr\xP@precbezierpoly\X@p\L@c\R@c\X@c\xP@tm%
  +8\d@Y*\xP@off/\@tempdimb\relax
\xP@ym\dimexpr\xP@precbezierpoly\Y@p\U@c\D@c\Y@c\xP@tm%
  -8\d@X*\xP@off/\@tempdimb\relax
```

Tangent at middle point

```
\xP@Tmx\d@X
\xP@Tmy\d@Y
\xP@ifabsless\xP@Tmy\xP@Tmx
  \let\xP@tmy\xP@Tmyx
  \let\xP@tmx\@empty
\else
  \ifdim\xP@Tmy=\z@
    \let\xP@tmx\xP@Tmzero
    \let\xP@tmy\xP@Tmzero
  \else
    \let\xP@tmy\@empty
    \let\xP@tmx\xP@Tmxy
  \fi
\fi
```

Initial guesses for the tangent vector scalings `\xP@fa`, `\xP@fd` and the near-middle position `\xP@tm`

```
\xP@fa\p@
\xP@fd\p@
\xP@tm\p@
```

The main loop for finding the offset curve

```
\count@\z@
\loop
```

Set the new control points up.

```
  \xP@offsetpoints
  \@tempswafalse
```

At most 10 iterations

```
  \ifnum10>\count@
```

Determine the quality of the approximation by an objective function.

```
    \xP@objfun\xP@oldobj
    \ifdim\xP@oldobj>\xP@maxobjfun\relax\@tempswatrue\fi
  \fi
\if@tempswa
  \xP@offsetloop
\repeat
```

Return the new anchor points.

```
\xdef\@gtempa{\X@p\the\xP@xa\Y@p\the\xP@ya
  \L@c\the\xP@xb\U@c\the\xP@yb\R@c\the\xP@xc\D@c\the\xP@yc
  \X@c\the\xP@xd\Y@c\the\xP@yd\relax}%
}%
\@gtempa
}
```

`\xP@scaleT` •1 •3 •4 This macro contains another trick to improve the offset algorithm around sharp bends and cusps. It adjusts the length of the tangent/velocity vectors. Let (`\d@X`, `\d@Y`) be the velocity vector to the original curve at some point with velocity v_0. The velocity at the same point, considered on a partial segment scales linearly with the length of the parameter interval. Hence, the velocity v_1 in the partial segment is $v_1 = v_0 \cdot ($`\xP@b` $-$

\xP@a)/\xP@bigdim. Additionally the offset curve goes with a radius of $r+$\xP@off around bends with radius r in the original curve. As an approximation to the velocity in the offset curve, we therefore scale the velocity vector in the end to the norm $v_1 + 2\pi \cdot |$\xP@off$|$.

```
\newcommand*\xP@scaleT{%
  \xP@B6.28\xP@off
  \xP@abs\xP@B
  \xP@C\dimexpr\d@X*\xP@B/\@tempdimb\relax
  \xP@D\dimexpr\d@Y*\xP@B/\@tempdimb\relax
  \xP@A\dimexpr\xP@b-\xP@a\relax
  \d@X\dimexpr\xP@C+\d@X*\xP@A/\xP@bigdim\relax
  \d@Y\dimexpr\xP@D+\d@Y*\xP@A/\xP@bigdim\relax
```

Also record the change to the norm of the vector.

```
  \@tempdimb\dimexpr\xP@B+\@tempdimb*\xP@A/\xP@bigdim\relax
}
```

\xP@offsetloop •1 •3 •4 The iteration in the offset loop: set up and solve (or approximate) the linear system.

```
\newcommand*\xP@offsetloop{%
  \xP@C\dimexpr\xP@C/2\relax
  \xP@G\dimexpr\xP@G/2\relax
```

1st linear equation

```
  \xP@ta\dimexpr2\p@-\xP@tm\relax
  \xP@tb\dimexpr\xP@tm*\xP@ta/\p@\relax
  \xP@A\dimexpr\xP@Tax*\xP@tb/\p@*\xP@ta/\p@\relax
  \xP@B\dimexpr\xP@Tdx*\xP@tb/\p@*\xP@tm/\p@\relax
```

2nd linear equation

```
  \xP@E\dimexpr\xP@Tay*\xP@tb/\p@*\xP@ta/\p@\relax
  \xP@F\dimexpr\xP@Tdy*\xP@tb/\p@*\xP@tm/\p@\relax
```

3rd linear equation

```
  \xP@tb\dimexpr2\p@-3\xP@tm\relax
  \xP@tc\dimexpr\xP@tb+2\p@\relax
  \xP@I\dimexpr(2\xP@Tay\xP@tmx-2\xP@Tax\xP@tmy)*\xP@tb/\p@*\xP@ta/\p@\relax
  \xP@J\dimexpr(2\xP@Tdy\xP@tmx-2\xP@Tdx\xP@tmy)*\xP@tc/\p@*\xP@tm/\p@\relax
  \xP@K\dimexpr((\xP@yd-\xP@ya+(\xP@yb-\xP@yc)*3)
    *\xP@tm/\p@+(\xP@yc-2\xP@yb+\xP@ya)*2)*12\xP@tmx
    -((\xP@xd-\xP@xa+(\xP@xb-\xP@xc)*3)
    *\xP@tm/\p@+(\xP@xc-2\xP@xb+\xP@xa)*2)*12\xP@tmy\relax
```

Solve the system.

```
  \xP@solvelinearsystem
  \ifxP@validsol
```

Check whether the result is feasible and whether it actually improves the approximation.

```
    \xP@correctsol
    \ifdim\xP@ta=\z@
    \ifdim\xP@tb=\z@
    \ifdim\xP@tc=\z@
      \xP@validsolfalse
    \fi\fi\fi
  \fi
```

If the exact solution is not valid, try to at least approximate a solution.

```
  \ifxP@validsol
  \else
    \xP@applinsys
```

This time, the solution is not checked but applied immediately.

```
    \advance\xP@fa-\xP@ta
    \advance\xP@fd-\xP@tb
    \advance\xP@tm-\xP@tc
```

The near-middle parameter on the curve must not lie outside the segment.

```
    \ifdim\xP@tm<\z@\xP@tm\z@\fi
    \ifdim\xP@tm>2\p@\xP@tm2\p@\fi
  \fi
  \advance\count@\@ne
}
```

`\xP@maxsol` Heuristic: maximal solution so that no arithmetic overflow is produced.

```
\newcommand*\xP@maxsol{3pt}
```

`\xP@correctsol` •3 •4 Check whether the solution is feasible and actually improves the objective function.

```
\newcommand*\xP@correctsol{%
```

If the solution is too big, scale all variables uniformly.

```
  \xP@M\z@
  \xP@Max\xP@M\xP@ta
  \xP@Max\xP@M\xP@tb
  \xP@Max\xP@M\xP@tc
  \ifdim\xP@M>\xP@maxsol
    \xP@ta\dimexpr\xP@maxsol*\xP@ta/\xP@M\relax
    \xP@tb\dimexpr\xP@maxsol*\xP@tb/\xP@M\relax
    \xP@tc\dimexpr\xP@maxsol*\xP@tc/\xP@M\relax
  \fi
```

Apply the solution. Save the old value of `\xP@tm` to be able to restore it.

```
  \advance\xP@fa-\xP@ta
  \advance\xP@fd-\xP@tb
  \xP@M\xP@tm
  \advance\xP@tm-\xP@tc
```

The near-middle parameter must lie on the segment.

```
  \ifdim\xP@tm<\z@\xP@tm\z@\fi
  \ifdim\xP@tm>2\p@\xP@tm2\p@\fi
```

Check whether the solution actually improves the objective function.

```
  {\xP@offsetpoints
    \xP@objfun\xP@M
  \expandafter}%
```

If not, restore the old values and declare the solution invalid.

```
  \ifdim\xP@M>\xP@oldobj
    \advance\xP@fa\xP@ta
    \advance\xP@fd\xP@tb
    \xP@tm\xP@M
    \xP@validsolfalse
  \fi
}
```

`\xP@objfun` •3 •4 The objective function: sum of squares of the deviation in x- and y-direction and the angular deviation at the middle point. We also compute some terms which will be used in the linear system.

```
\newcommand*\xP@objfun[1]{%
  \xP@D\dimexpr\xP@bezierpoly\xP@xa\xP@xb\xP@xc\xP@xd\xP@tm-\xP@xm\relax
```

```
\xP@H\dimexpr\xP@bezierpoly\xP@ya\xP@yb\xP@yc\xP@yd\xP@tm-\xP@ym\relax
\xP@C\xP@beziertan\xP@xa\xP@xb\xP@xc\xP@xd\xP@tm
\xP@G\xP@beziertan\xP@ya\xP@yb\xP@yc\xP@yd\xP@tm
\xP@L\dimexpr\xP@G\xP@tmx-\xP@C\xP@tmy\relax
```

If the deviation is too big, let the objective function be `\maxdimen`. Otherwise, compute the sum of squares.

```
#1\z@
\xP@Max#1\xP@D
\xP@Max#1\xP@H
\xP@Max#1\xP@L
#1\ifdim#1>4843165sp
  \maxdimen
\else
  \dimexpr\xP@D*\xP@D/\p@+\xP@H*\xP@H/\p@+\xP@L*\xP@L/\p@\relax
\fi
}
```

`\xP@offsetpoints` •3 •4 Compute the new control points from the factors `\xP@fa`, `\xP@fd`.

```
\newcommand*\xP@offsetpoints{%
  \xP@xb\dimexpr\xP@xa+\xP@Tax*\xP@fa/196608\relax
  \xP@yb\dimexpr\xP@ya+\xP@Tay*\xP@fa/196608\relax
  \xP@xc\dimexpr\xP@xd+\xP@Tdx*\xP@fd/196608\relax
  \xP@yc\dimexpr\xP@yd+\xP@Tdy*\xP@fd/196608\relax
}
```

`\xP@bezierpoly` Formula for the polynomial $8\left(\texttt{\#1}\cdot(1-t)^3+3\cdot\texttt{\#2}\cdot t(1-t)^2+3\cdot\texttt{\#3}\cdot t^2(1-t)+\texttt{\#4}\cdot t^3\right)$, $t=\frac{1}{2}\texttt{\#5}$.

```
\newcommand*\xP@bezierpoly[5]{%
  \dimexpr(((#4-#1+(#2-#3)*3)*#5/\p@+(#1-2#2+#3)*6)*#5/\p@+(#2-#1)*12)*#5/\p@
    +#1*8\relax
}
```

`\xP@precbezierpoly` Formula for the polynomial $8\left(\texttt{\#1}\cdot(1-t)^3+3\cdot\texttt{\#2}\cdot t(1-t)^2+3\cdot\texttt{\#3}\cdot t^2(1-t)+\texttt{\#4}\cdot t^3\right)$, $t=\texttt{\#5}/\texttt{\textbackslash xP@bigdim}$.

```
\newcommand*\xP@precbezierpoly[5]{%
  \dimexpr(((#4-#1+(#2-#3)*3)*2*#5/\xP@bigdim+(#1-2#2+#3)*6)*2*#5/\xP@bigdim
    +(#2-#1)*12)*2*#5/\xP@bigdim+#1*8\relax
}
```

`\xP@beziertan` Formula for the polynomial

$$24\left(-\texttt{\#1}\cdot(1-t)^2+\texttt{\#2}\cdot(3t^2-4t+1)+\texttt{\#3}\cdot(-3t^2+2t)+\texttt{\#4}\cdot t^2\right),\quad t=\tfrac{1}{2}\texttt{\#5}.$$

Up to a scalar factor, this is the derivative of the third order Bézier polynomial above.

```
\newcommand*\xP@beziertan[5]{%
  \dimexpr((#4-#1+(#2-#3)*3)*3*#5/32768+(#1-2#2+#3)*24)*#5/\p@+(#2-#1)*24\relax
}
```

`\xP@precbeziertan` Formula for the polynomial

$$\left(-\texttt{\#1}\cdot(1-t)^2+\texttt{\#2}\cdot(3t^2-4t+1)+\texttt{\#3}\cdot(-3t^2+2t)+\texttt{\#4}\cdot t^2\right),\quad t=\texttt{\#5}/\texttt{\textbackslash xP@bigdim}.$$

This is $\frac{1}{3}$ times the derivative of the third order Bézier polynomial.

```
\newcommand*\xP@precbeziertan[5]{%
  \dimexpr((#4-#1+(#2-#3)*3)*#5/\xP@bigdim+(#1-2#2+#3)*2)*#5/\xP@bigdim
  +#2-#1\relax
}
```

\xP@solvelinearsystem

•3 The macro \xP@solvelinearsystem solves a system of three linear equations by the Gauss algorithm. The coefficients and desired values are passed in the extended matrix

$$\left(\begin{array}{ccc|c} \texttt{\textbackslash xP@A} & \texttt{\textbackslash xP@B} & \texttt{\textbackslash xP@C} & \texttt{\textbackslash xP@D} \\ \texttt{\textbackslash xP@E} & \texttt{\textbackslash xP@F} & \texttt{\textbackslash xP@G} & \texttt{\textbackslash xP@H} \\ \texttt{\textbackslash xP@I} & \texttt{\textbackslash xP@J} & \texttt{\textbackslash xP@K} & \texttt{\textbackslash xP@L} \end{array}\right)$$

The solution is returned in the vector (\xP@ta, \xP@tb, \xP@tc).

\xP@varone
\xP@vartwo
\xP@varthree

With column swapping in the Gauss algorithm, variable names might be changed. These macros record the variables.

```
\@ifdefinable\xP@varone\relax
\@ifdefinable\xP@vartwo\relax
\@ifdefinable\xP@varthree\relax
```

\ifxP@validsol

Records if a valid solution to the linear system is returned.

```
\@ifdefinable\ifxP@validsol\relax
\@ifdefinable\xP@validsoltrue\relax
\@ifdefinable\xP@validsolfalse\relax
\newif\ifxP@validsol
```

```
\newcommand*\xP@solvelinearsystem{{%
```

Scale the matrix so that the highest absolute value in each row and each column is $\geq$ 2048pt and $<$ 4096pt.

```
  \xP@scalerow\xP@A\xP@B\xP@C\xP@D
  \xP@scalerow\xP@E\xP@F\xP@G\xP@H
  \xP@scalerow\xP@I\xP@J\xP@K\xP@L
  \xP@scalecol\xP@A\xP@E\xP@I\xP@scaleone
  \xP@scalecol\xP@B\xP@F\xP@J\xP@scaletwo
  \xP@scalecol\xP@C\xP@G\xP@K\xP@scalethree
```

Record the initial variable-to-column assignment.

```
  \let\xP@varone\xP@ta
  \let\xP@vartwo\xP@tb
  \let\xP@varthree\xP@tc
```

Find the pivot position. \xP@M is used temporarily.

```
  \count@\m@ne
  \@tempcnta\m@ne
  \xP@ifabsless\xP@A\xP@B\@tempcnta\z@\xP@M\xP@B
             \else\xP@M\xP@A\fi
  \xP@ifabsless\xP@M\xP@C\@tempcnta\@ne\xP@M\xP@C\fi
  \xP@ifabsless\xP@M\xP@E\@tempcnta\m@ne\count@\z@\xP@M\xP@E\fi
  \xP@ifabsless\xP@M\xP@F\@tempcnta\z@\count@\z@\xP@M\xP@F\fi
  \xP@ifabsless\xP@M\xP@G\@tempcnta\@ne\count@\z@\xP@M\xP@G\fi
  \xP@ifabsless\xP@M\xP@I\@tempcnta\m@ne\count@\@ne\xP@M\xP@I\fi
  \xP@ifabsless\xP@M\xP@J\@tempcnta\z@\count@\@ne\xP@M\xP@J\fi
  \xP@ifabsless\xP@M\xP@K\@tempcnta\@ne\count@\@ne\fi
```

Swap rows

```
  \ifcase\count@
    \xP@swapdim\xP@A\xP@E
    \xP@swapdim\xP@B\xP@F
    \xP@swapdim\xP@C\xP@G
    \xP@swapdim\xP@D\xP@H
  \or
    \xP@swapdim\xP@A\xP@I
```

```
    \xP@swapdim\xP@B\xP@J
    \xP@swapdim\xP@C\xP@K
    \xP@swapdim\xP@D\xP@L
  \fi
```

Swap columns

```
  \ifcase\@tempcnta
    \xP@swapdim\xP@A\xP@B
    \xP@swapdim\xP@E\xP@F
    \xP@swapdim\xP@I\xP@J
    \let\xP@varone\xP@tb
    \let\xP@vartwo\xP@ta
    \xP@swapnum\xP@scaleone\xP@scaletwo
  \or
    \xP@swapdim\xP@A\xP@C
    \xP@swapdim\xP@E\xP@G
    \xP@swapdim\xP@I\xP@K
    \let\xP@varone\xP@tc
    \let\xP@varthree\xP@ta
    \xP@swapnum\xP@scaleone\xP@scalethree
  \fi
```

First elimination

```
  \multiply\xP@E\m@ne
  \multiply\xP@I\m@ne
```

Absolute values below are < 8192pt.

```
  \ifdim\xP@A=\z@
  \else
    \advance\xP@F\dimexpr\xP@B*\xP@E/\xP@A\relax
    \advance\xP@G\dimexpr\xP@C*\xP@E/\xP@A\relax
    \advance\xP@H\dimexpr\xP@D*\xP@E/\xP@A\relax
    \advance\xP@J\dimexpr\xP@B*\xP@I/\xP@A\relax
    \advance\xP@K\dimexpr\xP@C*\xP@I/\xP@A\relax
    \advance\xP@L\dimexpr\xP@D*\xP@I/\xP@A\relax
  \fi
```

Find the second pivot element. `\xP@M` is used temporarily.

```
  \count@\m@ne
  \xP@ifabsless\xP@F\xP@G\@tempcnta\z@\xP@M\xP@G
    \else\@tempcnta\m@ne\xP@M\xP@F\fi
  \xP@ifabsless\xP@M\xP@J\@tempcnta\m@ne\count@\z@\xP@M\xP@J\fi
  \xP@ifabsless\xP@M\xP@K\@tempcnta\z@\count@\z@\fi
```

Swap rows

```
  \ifnum\count@=\z@
    \xP@swapdim\xP@F\xP@J
    \xP@swapdim\xP@G\xP@K
    \xP@swapdim\xP@H\xP@L
  \fi
```

Swap columns

```
  \ifnum\@tempcnta=\z@
    \xP@swapdim\xP@B\xP@C
    \xP@swapdim\xP@F\xP@G
    \xP@swapdim\xP@J\xP@K
    \let\@tempa\xP@varthree
    \let\xP@varthree\xP@vartwo
    \let\xP@vartwo\@tempa
```

```
    \xP@swapnum\xP@scaletwo\xP@scalethree
  \fi
```

Second elimination. Absolute values are < 16384pt.

```
  \ifdim\xP@F=\z@
  \else
    \advance\xP@K\dimexpr-\xP@G*\xP@J/\xP@F\relax
    \advance\xP@L\dimexpr-\xP@H*\xP@J/\xP@F\relax
  \fi
```

Compute the result from the upper triagonal form. Since the matrix can be singular, we have to ensure in every step that no overflow occurs. In general, we do not allow any solution greater than 60pt.

```
  \xP@ifabsless{\dimexpr\xP@L/60\relax}{\dimexpr\xP@K/\xP@scalethree\relax}%
    \xP@validsoltrue
    \xP@varthree\dimexpr\xP@L*(\xP@scalethree*\p@)/\xP@K\relax
  \else
    \xP@validsolfalse
  \fi
  \xP@checkabs{\xP@H/8191}{\xP@F/\xP@scaletwo}%
  \xP@checkabs{\xP@G/\xP@scalethree/136}{\xP@F/\xP@scaletwo}%
  \ifxP@validsol
    \xP@vartwo\dimexpr\xP@H*(\xP@scaletwo*\p@)/\xP@F
      -\xP@varthree*\xP@scaletwo/\xP@scalethree*\xP@G/\xP@F\relax
    \xP@checkabs\xP@vartwo{60pt}%
  \fi
  \xP@checkabs{\xP@D/5461}{\xP@A/\xP@scaleone}%
  \xP@checkabs{\xP@B/\xP@scaletwo/91}{\xP@A/\xP@scaleone}%
  \xP@checkabs{\xP@C/\xP@scalethree/91}{\xP@A/\xP@scaleone}%
  \ifxP@validsol
    \xP@varone\dimexpr\xP@D*(\xP@scaleone*\p@)/\xP@A
      -\xP@vartwo*\xP@scaleone/\xP@scaletwo*\xP@B/\xP@A
      -\xP@varthree*\xP@scaleone/\xP@scalethree*\xP@C/\xP@A\relax
    \xP@checkabs\xP@varone{60pt}%
  \fi
```

Return the result.

```
  \xdef\@gtempa{%
    \ifxP@validsol
      \xP@ta\the\xP@ta\relax
      \xP@tb\the\xP@tb\relax
      \xP@tc\the\xP@tc\relax
      \noexpand\xP@validsoltrue
    \else
      \noexpand\xP@validsolfalse
    \fi
  }%
  }\@gtempa
}
```

`\xP@scalerow` •3 Scale a row of the matrix to improve numerical precision. We scale by a power of two such that the maximal length is between 2048pt and 4096pt.

```
\newcommand*\xP@scalerow[4]{%
  \xP@M\z@
  \xP@Max\xP@M#1%
  \xP@Max\xP@M#2%
  \xP@Max\xP@M#3%
  \xP@Max\xP@M#4%
```

$134217727 = 2048 \cdot 65536 - 1$

```
  \count@134217727
  \loop
    \divide\xP@M\tw@
  \ifdim\xP@M>\z@
    \divide\count@\tw@
  \repeat
  \advance\count@\@ne
  \multiply#1\count@
  \multiply#2\count@
  \multiply#3\count@
  \multiply#4\count@
}
```

`\xP@scalecol` •3 Scale a column of the matrix to improve numerical precision. The scaling factor has to be recorded for the solution assignment later.

```
\newcommand*\xP@scalecol[4]{%
  \xP@M\z@
  \xP@Max\xP@M#1%
  \xP@Max\xP@M#2%
  \xP@Max\xP@M#3%
```

$16777215 = 2048 \cdot 8192 - 1$

```
  #416777215
  \loop
    \divide\xP@M\tw@
  \ifdim\xP@M>\z@
    \divide#4\tw@
  \repeat
  \advance#4\@ne
  \multiply#1#4%
  \multiply#2#4%
  \multiply#3#4%
}
```

`\xP@checkabs`

```
\newcommand*\xP@checkabs[2]{%
  \xP@ifabsless{\dimexpr#1\relax}{\dimexpr#2\relax}\else\xP@validsolfalse\fi}
```

`\xP@applinsys` •1 •3 •6 This is the second, alternative algorithm for Newton's method in the offset algorithm. Approximate a solution x for the linear system $Ax = b$ for a (3×3)-matrix A. The aim is to make the norm $\|Ax - b\|$ small with small values of $\|x\|$. The approach: Set $x = \lambda A^t b$ since the normed scalar product $\langle Ax, b\rangle / \|x\|$ is maximal in this case. The norm $\|Ax - b\|$ is then minimal for $\lambda = \|A^t b\|^2 / \|AA^t b\|^2$.

This approximation is performed between one and three times.

```
\newcommand*\xP@applinsys{{%
```

First iteration: approximate a solution and record the result.

```
  \xP@applinsys@
  \xP@ta\xP@dta
  \xP@tb\xP@dtb
  \xP@tc\xP@dtc
```

If the result is nonzero...

```
  \xP@checkapp
  \if@tempswa
```

…modify the objective function by the estimated change, approximate again,…

```
    \xP@modobj
    \xP@applinsys@
```

…and test for a nonzero result. If it is nonzero, repeat it a third time.

```
    \xP@checkapp
    \if@tempswa
      \xP@modsol
      \xP@modobj
      \xP@applinsys@
      \xP@modsol
    \fi
  \fi
```

Return the accumulated approximation from one to three iterations.

```
  \xdef\@gtempa{%
    \xP@ta\the\xP@ta\relax
    \xP@tb\the\xP@tb\relax
    \xP@tc\the\xP@tc\relax
  }}\@gtempa
}
```

`\xP@checkapp` •6 Check whether the solution is nonzero.

```
\newcommand*\xP@checkapp{%
  \@tempswatrue
  \ifdim\xP@dta=\z@
  \ifdim\xP@dtb=\z@
  \ifdim\xP@dtc=\z@
    \@tempswafalse
  \fi\fi\fi
}
```

`\xP@modobj` •3 •6 Modify the objective function by the estimated difference, according to the first-order approximation.

```
\newcommand*\xP@modobj{%
  \advance\xP@D
    \dimexpr-\xP@A*\xP@dta/\p@-\xP@B*\xP@dtb/\p@-\xP@C*\xP@dtc/\p@\relax
  \advance\xP@H
    \dimexpr-\xP@E*\xP@dta/\p@-\xP@F*\xP@dtb/\p@-\xP@G*\xP@dtc/\p@\relax
  \advance\xP@L
    \dimexpr-\xP@I*\xP@dta/\p@-\xP@J*\xP@dtb/\p@-\xP@K*\xP@dtc/\p@\relax
}
```

`\xP@modsol` •3 •6 Modify the solution vector by the approximation.

```
\newcommand*\xP@modsol{%
  \advance\xP@ta\xP@dta
  \advance\xP@tb\xP@dtb
  \advance\xP@tc\xP@dtc
}
```

`\xP@applinsys@` •1 •3 •6 The heart of the approximation routine.

```
\newcommand*\xP@applinsys@{{%
```

Determine scaling factors `\xP@sa` and `\xP@sb` to improve numerical precision.

```
  \xP@sa\z@
  \xP@Max\xP@sa\xP@A
  \xP@Max\xP@sa\xP@B
```

```
\xP@Max\xP@sa\xP@C
\xP@Max\xP@sa\xP@E
\xP@Max\xP@sa\xP@F
\xP@Max\xP@sa\xP@G
\xP@Max\xP@sa\xP@I
\xP@Max\xP@sa\xP@J
\xP@Max\xP@sa\xP@K
\xP@sa\ifdim\xP@sa<5460pt\thr@@\xP@sa\else\maxdimen\fi
\xP@sb\z@
\xP@Max\xP@sb\xP@D
\xP@Max\xP@sb\xP@H
\xP@Max\xP@sb\xP@L
```

Scale the vector b.

```
\ifdim\xP@sb>\z@
  \xP@D\dimexpr\xP@D*\maxdimen/\xP@sb\relax
  \xP@H\dimexpr\xP@H*\maxdimen/\xP@sb\relax
  \xP@L\dimexpr\xP@L*\maxdimen/\xP@sb\relax
\fi
```

Vector $A^t b$ (scaled)

```
\xP@Aba\dimexpr\xP@A*\xP@D/\xP@sa+\xP@E*\xP@H/\xP@sa+\xP@I*\xP@L/\xP@sa\relax
\xP@Abb\dimexpr\xP@B*\xP@D/\xP@sa+\xP@F*\xP@H/\xP@sa+\xP@J*\xP@L/\xP@sa\relax
\xP@Abc\dimexpr\xP@C*\xP@D/\xP@sa+\xP@G*\xP@H/\xP@sa+\xP@K*\xP@L/\xP@sa\relax
```

Vector $AA^t b$ (scaled)

```
\xP@AAba\dimexpr\xP@A*\xP@Aba/\xP@sa+\xP@B*\xP@Abb/\xP@sa
  +\xP@C*\xP@Abc/\xP@sa\relax
\xP@AAbb\dimexpr\xP@E*\xP@Aba/\xP@sa+\xP@F*\xP@Abb/\xP@sa
  +\xP@G*\xP@Abc/\xP@sa\relax
\xP@AAbc\dimexpr\xP@I*\xP@Aba/\xP@sa+\xP@J*\xP@Abb/\xP@sa
  +\xP@K*\xP@Abc/\xP@sa\relax
```

Another scaling factor.

```
\xP@sc\z@
\xP@Max\xP@sc\xP@Aba
\xP@Max\xP@sc\xP@Abb
\xP@Max\xP@sc\xP@Abc
\xP@Max\xP@sc\xP@AAba
\xP@Max\xP@sc\xP@AAbb
\xP@Max\xP@sc\xP@AAbc
```

$\|A^t b\|^2$ and $\|AA^t b\|^2$

```
\ifdim\xP@sc=\z@
  \xP@AAb\z@
\else
  \xP@Ab\dimexpr\xP@Aba*\xP@bigdim/\xP@sc*\xP@Aba/\xP@sc
               +\xP@Abb*\xP@bigdim/\xP@sc*\xP@Abb/\xP@sc
               +\xP@Abc*\xP@bigdim/\xP@sc*\xP@Abc/\xP@sc
  \relax
  \xP@AAb\dimexpr\xP@AAba*\xP@bigdim/\xP@sc*\xP@AAba/\xP@sc
                +\xP@AAbb*\xP@bigdim/\xP@sc*\xP@AAbb/\xP@sc
                +\xP@AAbc*\xP@bigdim/\xP@sc*\xP@AAbc/\xP@sc
  \relax
\fi
```

The approximation $x = \lambda A^t b$ with $\lambda = \|A^t b\|^2 / \|AA^t b\|^2$.

```
\xdef\@gtempa{%
  \ifdim\xP@AAb=\z@
```

```
      \xP@dta\z@
      \xP@dtb\z@
      \xP@dtc\z@
    \else
      \xP@dta\the\dimexpr\xP@Aba*\xP@sb/\xP@sa*\p@/\xP@AAb*\xP@Ab/\maxdimen
        \relax
      \xP@dtb\the\dimexpr\xP@Abb*\xP@sb/\xP@sa*\p@/\xP@AAb*\xP@Ab/\maxdimen
        \relax
      \xP@dtc\the\dimexpr\xP@Abc*\xP@sb/\xP@sa*\p@/\xP@AAb*\xP@Ab/\maxdimen
        \relax
    \fi
  }%
  }\@gtempa
}
```

`\ifxP@offsetok` Switch whether the offset curve is enough

```
\@ifdefinable\ifxP@offsetok\relax
\@ifdefinable\xP@offsetoktrue\relax
\@ifdefinable\xP@offsetokfalse\relax
\newif\ifxP@offsetok
```

`\xP@maxdev` Maximal deviation, measured at 19 points on the curve. The actual tolerance is 1/8 of `\xP@maxdev`. With the current value 0.1pt, the tolerance is 0.0125pt, which is about 1/32 of the line width for the Computer Modern fonts.

```
\newcommand*\xP@maxdev{.1pt}
```

`\xP@maxobjfun` Tolerance for the objective function. Recommended value is $\frac{1}{2}(\texttt{\textbackslash xP@maxdev})^2$.

```
\newcommand*\xP@maxobjfun{.005pt}
```

`\xP@testoffset` •1 •5 Test procedure for the offset curve. It tests whether the Bézier curve defined by the control points `\X@p`,...,`\Y@c` is a good approximation for the offset curve of the partial curve defined by `\xP@xa`,...,`\xP@yd` in the parameter interval [`\xP@a`, `\xP@b`] ⊆ [0pt, `\xP@bigdim`].

The parameter interval is uniformly divided by 20, and the deviation is measured at the 19 inner positions. (Since the boundary points are offset exactly by the algorithm, they do not need to be checked.)

For simplicity, the parameter interval for both curves is normalized to $[0, 1]$ in the following explanations. Denote the original curve by $c_1 : [0, 1] \to \mathbb{R}^2$ and the offset curve by c_2. The quality test is passed if the offset curve fulfills at each of the 19 test points $t_i \in \{\frac{1}{20}, \ldots, \frac{19}{20}\}$ one of the following two conditions:

- Let v be the tangent vector $c_1'(t_i)$. For $w := c_1(t_i) - c_2(t_i)$, denote by w_{par} the component parallel to v and by w_{orth} the component orthogonal to v. The test is passed if $|w_{par}| + |w_{orth} - \texttt{\textbackslash xP@off}| \leq \frac{1}{8}\texttt{\textbackslash xP@maxdev}$. If $\|v\|$ is very small so that the direction cannot be determined precisely, the condition is $\big|\|c_1(t_i) - c_2(t_i)\| - |\texttt{\textbackslash xP@off}|\big| \leq \frac{1}{8}\texttt{\textbackslash xP@maxdev}$.

- Compute the normal line at t_i to the curve c_1 and intersect it with c_2. The intersection point is allowed to have a different parameter $\tilde{t}_i \in [t_i - 0.5, t_i + 0.5] \cap [0, 1]$. Then let $w := c_1(t_i) - c_2(\tilde{t}_i)$ and test whether $|w_{par}| + |w_{orth} - \texttt{\textbackslash xP@off}| \leq \frac{1}{8}\texttt{\textbackslash xP@maxdev}$. ($|w_{par}|$ is very small in this case and is nonzero only because of limited precision, in particular since $\tilde{t}_i$ is determined with an error of $\approx 2^{-17}$ $(= \frac{1}{2}\mathrm{sp})$.)

```
\newcommand*\xP@testoffset{{%
```

Default values for the return statement and the loop continuation.

```
\gdef\xP@afteroffsetok{\xP@offsetoktrue}%
\def\xP@offsetokif{\ifdim\xP@ti<1.85pt}%
\xP@ti.1pt
\loop
```

`\xP@tip` $= t_i$, denormalized for c_1

```
  \xP@tip\dimexpr\xP@a+(\xP@b-\xP@a)*\xP@ti/131072\relax
```

Point on the original curve c_1 (scaled by -8)

```
  \L@p\xP@precbezierpoly\xP@xa\xP@xb\xP@xc\xP@xd\xP@tip
  \U@p\xP@precbezierpoly\xP@ya\xP@yb\xP@yc\xP@yd\xP@tip
```

$8c_2(t_i) - 8c_1(t_i)$

```
  \xP@valA\dimexpr\xP@bezierpoly\X@p\L@c\R@c\X@c\xP@ti-\L@p\relax
  \xP@valB\dimexpr\xP@bezierpoly\Y@p\U@c\D@c\Y@c\xP@ti-\U@p\relax
```

v

```
  \d@X3\xP@precbeziertan\xP@xa\xP@xb\xP@xc\xP@xd\xP@tip
  \d@Y3\xP@precbeziertan\xP@ya\xP@yb\xP@yc\xP@yd\xP@tip
  \xP@veclen
```

Decide if v is big enough (heuristically, may be changed in the future)

```
  \@tempdimc\dimexpr(\xP@b-\xP@a)*\@tempdimb/\xP@bigdim\relax
  \xP@abs\@tempdimc
  \ifdim.01pt<\@tempdimc
```

$8w_{par}$, $8w_{orth} - 8$`\xP@off`,

```
    \xP@devA\dimexpr\xP@valA*\d@X/\@tempdimb+\xP@valB*\d@Y/\@tempdimb\relax
    \xP@devB\dimexpr\xP@valA*\d@Y/\@tempdimb-\xP@valB*\d@X/\@tempdimb-8\xP@off
      \relax
    \xP@abs\xP@devA
    \xP@abs\xP@devB
    \@tempdima\dimexpr\xP@devA+\xP@devB\relax
  \else
```

If the velocity is zero, just pass the test.

```
    \ifdim\@tempdimc=\z@
      \@tempdima\z@
    \else
```

$8\|c_1(t_i) - c_2(t_i)\|$

```
      {%
      \d@X\xP@valA
      \d@Y\xP@valB
      \xP@veclen@
      \global\dimen@i\@tempdimb
      }\@tempdima\dimen@i
      \advance\@tempdima\ifdim\xP@off>\z@-\fi8\xP@off
      \xP@abs\@tempdima
    \fi
  \fi
```

If the first condition is not fulfilled, test the second one.

```
  \ifdim\@tempdima>\xP@maxdev
```

$-c_1(t_i)$

```
    \divide\L@p-8\relax
    \divide\U@p-8\relax
```

Affine transformation of the offset curve: translate by $-c_1(t_i)$ and rotate so that the tangent v to $c_1(t_i)$ becomes the x-axis.

```
      {%
      \xP@transformcoor\X@p\Y@p
      \xP@transformcoor\L@c\U@c
      \xP@transformcoor\R@c\D@c
      \xP@transformcoor\X@c\Y@c
```

Find the parameter $\tilde{t}_i$ and decide whether the approximation at $\tilde{t}_i$ is good.

```
      \xP@findzero
      }%
    \fi
  \xP@offsetokif
    \advance\xP@ti.1pt
  \repeat
  \expandafter}\xP@afteroffsetok
}
```

`\xP@afteroffsetok`

```
\newcommand*\xP@afteroffsetok{}
```

`\xP@offsetokif`

```
\newcommand*\xP@offsetokif{}
```

`\xP@transformcoor` •5 Affine coordinate transformation. First, translate the coordinates in (`#1`, `#2`) by the vector $-($`\L@p`, `\U@p`$)$, then rotate by the angle between $v :=$ (`\d@X`, `\d@Y`) and $(1, 0)$. The register `\@tempdimb` must contain the length $\|v\|$.

```
\newcommand*\xP@transformcoor[2]{%
  \advance#1\L@p
  \advance#2\U@p
  \@tempdima\dimexpr#1*\d@X/\@tempdimb+#2*\d@Y/\@tempdimb\relax
  #2\dimexpr#2*\d@X/\@tempdimb-#1*\d@Y/\@tempdimb\relax
  #1\@tempdima
}
```

`\xP@findzero` •5 Find the parameter $\tilde{t}_i$ by nested intervals/intermediate value theorem.

```
\newcommand*\xP@findzero{%
  \xP@setleftvalue{.05}%
  \xP@setrightvalue{.05}%
```

Normalize: function value (x-coordinate) should be nonnegative at the upper end.

```
  \ifdim\xP@valB<\z@\xP@reversecoeff\fi
```

If the function value at the lower end is also positive, try a smaller parameter interval $t_i \pm \delta$ pt for $\delta \in \{.5, .35, .25, .2, .15, .1, .05\}$. Maybe we have different signs for the x-coordinate for the larger boundary parameters.

```
  \ifdim\xP@valA>\z@
    \@tempswatrue
    \@for\@tempa:={.1,.15,.2,.25,.35,.5,1.1}\do{%
      \if@tempswa
        \xP@setleftvalue\@tempa
        \ifdim\xP@valA<\z@\@tempswafalse\fi
        \if@tempswa
          \xP@setrightvalue\@tempa
          \ifdim\xP@valB<\z@
            \@tempswafalse
```

```
      \xP@reversecoeff
     \fi
    \fi
   \fi
  }%
```

Last resort: Try the midpoint.

```
  \if@tempswa
   \L@p\xP@ti
   \xP@valA\xP@bezierpoly\X@p\L@c\R@c\X@c\L@p
```

If the midpoint leads to a negative value, we can proceed with a small interval. Otherwise, set both boundary points to the midpoint and effectively skip nested intervals.

```
   \ifdim\xP@valA<\z@
```

We had this before, so we know that the value is positive.

```
    \xP@setrightvalue{.05}%
   \else
    \U@p\L@p
    \xP@valB\xP@valA
   \fi
  \fi
 \fi
```

The actual nested interval algorithm

```
 \loop
 \ifnum\numexpr\U@p-\L@p\relax>\@ne
  \xP@ti\dimexpr(\L@p+\U@p)/2\relax
  \xP@devA\xP@bezierpoly\X@p\L@c\R@c\X@c\xP@ti
  \ifdim\xP@devA>\z@
   \U@p\xP@ti
   \xP@valB\xP@devA
  \else
   \L@p\xP@ti
   \xP@valA\xP@devA
  \fi
 \repeat
```

Take the left or right boundary point (only 1sp apart), depending on which one yields the smaller x-coordinate.

```
 \xP@ifabsless\xP@valB\xP@valA
  \L@p\U@p
  \xP@valA\xP@valB
 \fi
```

Compare the y-coordinate with `\xP@off`.

```
 \xP@valB\dimexpr\xP@bezierpoly\Y@p\U@c\D@c\Y@c\L@p+8\xP@off\relax
 \xP@abs\xP@valA
 \xP@abs\xP@valB
 \ifdim\dimexpr\xP@valA+\xP@valB\relax>\xP@maxdev\relax
  \xP@failed
 \fi
}
```

`\xP@failed` Break the loop for the t_i in `\xP@testoffset`. Set the return value to false.

```
\newcommand*\xP@failed{%
 \global\let\xP@offsetokif\iffalse
 \gdef\xP@afteroffsetok{\xP@offsetokfalse}%
}
```

\xP@reversecoeff Reverse the function for the nested interval algorithm.

```
\newcommand*\xP@reversecoeff{%
    \multiply\X@p\m@ne
    \multiply\L@c\m@ne
    \multiply\R@c\m@ne
    \multiply\X@c\m@ne
    \multiply\xP@valA\m@ne
    \multiply\xP@valB\m@ne
}
```

\xP@setleftvalue •5

```
\newcommand*\xP@setleftvalue[1]{%
  \L@p\dimexpr\xP@ti-#1\p@\relax
  \ifdim\L@p<-.1pt\L@p-.1pt\fi
  \xP@valA\xP@bezierpoly\X@p\L@c\R@c\X@c\L@p
}
```

\xP@setrightvalue •5

```
\newcommand*\xP@setrightvalue[1]{%
  \U@p\dimexpr\xP@ti+#1\p@\relax
  \ifdim\U@p>2.1\p@\U@p2.1\p@\fi
  \xP@valB\xP@bezierpoly\X@p\L@c\R@c\X@c\U@p
}
```

9.10 Multiple dashed curves

\xP@splinedbldashed

```
\newcommand*\xP@splinedbldashed{%
  \xP@checkspline\xP@splinemultdashed\xP@doublestroke}
```

\xP@splinetrbldashed

```
\newcommand*\xP@splinetrbldashed{%
  \xP@checkspline\xP@splinemultdashed\xP@trblstroke}
```

\xP@splinemultdashed

```
\newcommand*\xP@splinemultdashed[1]{%
```

Expected dash number. It is an even number if the spline is the continuation of the previous one, otherwise (default case) an odd number.

```
  \xP@testcont\xP@dashmacro
  \@tempcnta
  \ifxP@splinecont
    \numexpr2*((\@tempdimb-\xydashl@/3)/(2*\xydashl@))\relax
  \else
    \numexpr2*((\@tempdimb+\xydashl@)/(2*\xydashl@))-1\relax
  \fi
  \ifnum\@tempcnta>\@ne
    \xP@splinemultdashed@#1%
  \else
```

One dash: paint a solid line. Less than one dash: Leave the segment out, just record the end point.

```
    \ifnum\@tempcnta=\@ne
      \xP@splinemultsolid#1
    \else
      \xP@savec
```

```
    \fi
  \fi
  \global\let\xP@lastpattern\xP@dashmacro
}
```

\xP@splinemultdashed@

•1 •7 Make a list of parameter pairs for the start and end point of a dash.

```
\newcommand*\xP@splinemultdashed@[1]{{%
  \xP@inibigdim
```

Dash length

```
  \@tempdima\dimexpr\@tempdimb/\@tempcnta\relax
  \xP@temppar\z@
  \toks@{}%
  \xP@savec
  \ifodd\@tempcnta
  \else
    \xP@slide
  \fi
  \@tempcnta\z@
  \loop
    \advance\@tempcnta\@ne
    \xP@append\toks@{\ifodd\@tempcnta\noexpand\xP@paintdash\fi
      {\the\xP@temppar}}%
    \xP@oldpar\xP@temppar
    \xP@slide
  \ifdim\xP@temppar<\xP@bigdim
  \repeat
```

The last position is kept as a scaling factor so that the last dot can be drawn at exactly the parameter 1. Use the last or the next-to-last position, depending on the parity of segments.

```
  \xP@lastpar
    \ifodd\@tempcnta
      \xP@temppar
      \xP@append\toks@{{\the\xP@temppar}}%
    \else
      \xP@oldpar
    \fi
```

Convert the list of parameters to a list of PDF tokens.

```
  \@temptokena{}%
  \xP@setsolidpat
  \global\let\xP@lastpattern\xP@dashmacro
  \@for\@tempa:={#1}\do{\the\toks@}%
  \xP@stroke{\the\@temptokena}%
}}
```

\xP@paintdash

•1 •7

```
\newcommand*\xP@paintdash[2]{%
  \xP@paintsolid{\dimexpr#1*\xP@bigdim/\xP@lastpar\relax}%
    {\dimexpr#2*\xP@bigdim/\xP@lastpar\relax}%
}
```

9.11 Multiple dotted curves

\splinedbldotted@
\xP@splinedbldotted@

```
\xP@hook{curve}{splinedbldotted@}
\newcommand*\xP@splinedbldotted@{%
```

```
  \let\xP@normalmult\@ne
  \xP@checkspline\xP@splinemultdotted\xP@doublestroke}
```

\xP@splinetrbldotted

```
\newcommand*\xP@splinetrbldotted{%
  \let\xP@normalmult\tw@
  \xP@checkspline\xP@splinemultdotted\xP@trblstroke}
```

\xP@multidottedpat Dotted lines with multiple strokes are drawn in a different way from single-stroked lines. They are composed of many small, straight lines normal to the curve at every dot position. Hence, the dot pattern for multiple curves has dots which are spaced by the normal distance between strokes.

```
\newcommand*\xP@multidottedpat{%
  \def\xP@pattern{0 J [\xP@coor\xP@preclw{\xydashh@-\xP@preclw}]0 d}%
  \global\let\xP@lastpattern\xP@dotmacro
}
```

\xP@normalmult

```
\@ifdefinable\xP@normalmult\relax
```

\xP@splinemultdotted •1 •7

```
\newcommand\xP@splinemultdotted[1]{{%
  \xP@inibigdim
```

Make a list of dot positions on the spline segment.

```
  \xP@temppar\z@
  \xP@testcont\xP@dotmacro
  \ifxP@splinecont
```

Expected dot distance (see the formula in `\xP@setdottedpat`)

```
    \@tempdimc\dimexpr\@tempdimb/(\@tempdimb/131072+1)\relax
    \@tempdima\dimexpr\@tempdimc-\xP@preclw/2\relax
    \xP@slide
    \@tempdima\@tempdimc
  \else
    \@tempdima\dimexpr\xP@preclw/2\relax
    \xP@slide
```

Expected dot distance (see the formula in `\xP@setdottedpat`)

```
    \@tempdima\dimexpr\@tempdimb-\xP@preclw\relax
    \ifdim\@tempdima<\z@\@tempdima\z@\fi
    \@tempdima\dimexpr\@tempdima/(\@tempdima/131072+1)\relax
  \fi
  \xP@savec
  \toks@{}%
```

If the end of the segment is reached before the first dot position, leave the segment out.

```
  \ifdim\xP@temppar<\xP@bigdim
    \loop
      \xP@append\toks@{\noexpand\xP@paintdot{\the\xP@temppar}}%
      \xP@oldpar\xP@temppar
      \xP@slide
    \ifdim\xP@temppar<\xP@bigdim
    \repeat
    \xP@velocity\xP@bigdim\xP@tempvel
```

Test whether the last or the next-to-last dot is closer to `\xP@bigdim`. Measure from the end of the dot, hence the contribution of `\xP@preclw`. Also consider the case that the velocity at the end point is very small. In this case, always choose the next-to-last dot as the final one.

```
    \ifdim
      \ifdim\xP@preclw<\xP@tempvel
        \dimexpr2\xP@bigdim-\xP@oldpar-\xP@preclw*\xP@bigdim/\xP@tempvel\relax
      \else
        -\maxdimen
      \fi<\xP@temppar
      \xP@temppar\xP@oldpar
    \else
      \xP@append\toks@{\noexpand\xP@paintdot{\the\xP@temppar}}%
    \fi
    \@tempdima\dimexpr\xP@preclw/2\relax
    \xP@slide
    \xP@lastpar\xP@temppar
```

Convert the list of parameters to a list of PDF tokens.

```
    \@temptokena{}%
    \the\toks@
```

Actually draw the points in the list.

```
    \xP@multidottedpat
    \xP@stroke{\the\@temptokena}%
  \else
```

Leave the segment out because it is too short.

```
    \global\let\xP@lastpattern\@empty
  \fi
}}
```

`\xP@slide` •1 •7 Slide along the Bézier segment by `\@tempdima`. Needs: XY-pic spline parameter, current position parameter `\xP@temppar`, total spline length `\@tempdimb`.

```
\newcommand*\xP@slide{{%
  \xP@slide@
```

Return the new spline parameter after sliding.

```
  \global\dimen@i\xP@temppar
  }\xP@temppar\dimen@i
}
```

`\xP@slide@` •1 •7

```
\newcommand*\xP@slide@{%
```

Compute the velocity at two points, the starting point and an estimate for the end point.

```
  \xP@velocity\xP@temppar\xP@tempvel
```

The first estimate for the parameter increment is based on the total spline length.

```
  \@tempdimc\dimexpr\xP@bigdim*\@tempdima/\@tempdimb\relax
  \count@\z@
  \@tempswatrue
```

Improve the parameter increment iteratively.

```
  \loop
```

Velocity at the estimated end point.

```
    \xP@velocity{\xP@temppar+\@tempdimc}\xP@tempvel@
```

Prevent arithmetic overflow.

```
  \ifdim\dimexpr\@tempdima*4/13\relax>\xP@tempvel@
    \@tempswafalse
  \else
```

Difference to the old parameter increment. This is Newton's method, applied to the estimated spline length based on the velocities `\xP@tempvel` and `\xP@tempvel@` at `\xP@temppar` and (`\xP@temppar` + `\@tempdimc`).

```
    \xP@parinc\dimexpr\@tempdima*\xP@bigdim/\xP@tempvel@
      -(\xP@tempvel+\xP@tempvel@)/2*\@tempdimc/\xP@tempvel@\relax
    \advance\@tempdimc\xP@parinc
```

If the estimated parameter increment is bigger than .12, increase the parameter by .1 and slide only partially. This increases the precision if the parameter increment is big.

```
    \ifdim\@tempdimc>.12\xP@bigdim
      \@tempswafalse
    \else
```

If the estimate is not improved, break the loop.

```
      \ifdim\xP@parinc=\z@
        \@tempswafalse
      \else
```

Also break the loop after 10 iterations.

```
         \ifnum\count@=9\relax
           \@tempswafalse
         \fi
      \fi
    \fi
  \fi
\if@tempswa
  \advance\count@\@ne
\repeat
```

Note that `\if@tempswa` is always false here.

If the parameter increment would be more than .1 and if the parameter is not too big already, increase the parameter by .1 and slide again.

```
\ifdim\xP@temppar<5461pt
\ifdim\@tempdimc>.1\xP@bigdim
  \@tempswatrue
\fi
\fi
\if@tempswa
  {%
    \dimen5\xP@temppar
    \advance\xP@temppar.1\xP@bigdim
```

Cap the end parameter to prevent arithmetic overflows.

```
    \ifdim\xP@temppar>5461pt\xP@temppar5461pt\fi
    \dimen7\xP@temppar
```

Determine the exact distance of the partial slide.

```
    \xP@shaveprec{\dimen5}{\dimen7}%
    \xP@bezierlength
    \global\dimen@i\dimexpr\@tempdima-\@tempdimb\relax
    \global\dimen3\xP@temppar
  }%
  \@tempdima\dimen@i
  \xP@temppar\dimen3\relax
```

Slide again.

```
    \expandafter\xP@slide@
  \else
```

Finish the slide and return the new parameter.

```
    \advance\xP@temppar\@tempdimc
  \fi
}
```

`\xP@paintdot` •1 •7

```
\newcommand*\xP@paintdot[1]{%
```

Scale the parameter with a correction factor

```
  \@tempdima\dimexpr#1*\xP@bigdim/\xP@lastpar\relax
```

Position at parameter value `\xP@temppar`

```
  \xP@tangent
  \xP@posX\dimexpr\xP@precbezierpoly\X@p\L@c\R@c\X@c\@tempdima/8\relax
  \xP@posY\dimexpr\xP@precbezierpoly\Y@p\U@c\D@c\Y@c\@tempdima/8\relax
```

Normal vector to the curve with length `\xydashh@`

```
  \@tempdima\dimexpr(\xydashh@+\xP@preclw/\xP@normalmult)/2\relax
  \L@p\dimexpr\d@Y*\@tempdima/\@tempdimb\relax
  \U@p\dimexpr-\d@X*\@tempdima/\@tempdimb\relax
```

Append two points on both sides of the curve to the list. (The “multidottedpat” pattern is made to draw points with distance `\xydashh@`.)

```
  \xP@append\@temptokena{\xP@coor{\xP@posX+\L@p*\xP@normalmult}%
      {\xP@posY+\U@p*\xP@normalmult}m %
    \xP@coor{\xP@posX-\L@p*(\xP@normalmult+\@ne)}%
      {\xP@posY-\U@p*(\xP@normalmult+\@ne)}l }%
}
```

9.12 Squiggled curves

`\xP@splinesquiggled`

```
\newcommand*\xP@splinesquiggled{%
  \xP@checkspline\xP@splinesquiggled@\z@}
```

`\xP@splinedblsquiggled`

```
\newcommand*\xP@splinedblsquiggled{%
  \xP@checkspline\xP@splinesquiggled@\xP@doublestroke}
```

`\xP@splinetrblsquiggled`

```
\newcommand*\xP@splinetrblsquiggled{%
  \xP@checkspline\xP@splinesquiggled@\xP@trblstroke}
```

`\xP@splinesquiggled@` •1 •7

```
\newcommand*\xP@splinesquiggled@[1]{{%
      \xP@inibigdim
```

Reverse the direction of the little arcs, if the last squiggle from the previous segment makes it necessary.

```
      \xP@testcont\xP@oddsquigglemacro
      \ifxP@splinecont
        \def\xP@squigsign{-}%
      \else
```

```
      \let\xP@squigsign\@empty
    \fi
    \xP@savec
```

Expected squiggle length

```
    \@tempcnta=\numexpr\@tempdimb/\xybsqll@\relax
    \ifnum\@tempcnta<\tw@\@tempcnta\tw@\fi
    \multiply\@tempcnta\tw@
    \@tempdima\dimexpr\@tempdimb/\@tempcnta\relax
    \xP@squiglen\@tempdima
```

Make a list of dot positions on the spline segment.

```
    \xP@temppar\z@
    \toks@{}%
    \@tempcnta\z@
    \loop
      \advance\@tempcnta\@ne
      \xP@append\toks@{\noexpand\xP@paintsquiggle{\the\xP@temppar}}%
      \xP@oldpar\xP@temppar
      \xP@slide
    \ifdim\xP@temppar<\xP@bigdim
    \repeat
```

The last position is kept as a scaling factor so that the last dot can be drawn at exactly the parameter 1. Use the last or the next-to-last position, on the parity of the number of positions.

```
    \xP@lastpar
      \ifodd\@tempcnta
        \xP@oldpar
        \advance\@tempcnta\m@ne
      \else
        \xP@temppar
        \xP@append\toks@{\noexpand\xP@paintsquiggle{\the\xP@temppar}}%
      \fi
```

Convert the list of parameters to a list of PDF tokens.

```
    \@temptokena{}%
    \xP@setsolidpat
```

Record the direction of the last squiggle.

```
    \global\expandafter\let\expandafter\xP@lastpattern
    \ifodd\numexpr\@tempcnta/2\if\xP@squigsign-+1\fi\relax
      \xP@oddsquigglemacro
    \else
      \xP@evensquigglemacro
    \fi
```

Draw the squiggles.

```
    \@for\@tempa:={#1}\do{%
      \let\xP@dosquiggle\xP@dosquiggle@
      \count@\z@
      \the\toks@
    }%
    \xP@stroke{\the\@temptokena}%
}}
```

`\xP@paintsquiggle` •1 •7

```
\newcommand*\xP@paintsquiggle[1]{%
  \xP@squigglevectors{#1}%
```

```
  \xP@dosquiggle
  \ifnum\count@=\thr@@\relax\count@\z@\else\advance\count@\@ne\fi
}
```

\xP@squigglevectors •1 •7

```
\newcommand*\xP@squigglevectors[1]{%
```

Scale the parameter with a correction factor

```
  \@tempdima\dimexpr#1*\xP@bigdim/\xP@lastpar\relax
```

Position at parameter value \xP@temppar, offset for multiple curves.

```
  \xP@tangent
  \xP@posX\dimexpr\xP@precbezierpoly\X@p\L@c\R@c\X@c\@tempdima/8%
    -\d@Y*(\@tempa)/\@tempdimb\relax
  \xP@posY\dimexpr\xP@precbezierpoly\Y@p\U@c\D@c\Y@c\@tempdima/8%
    +\d@X*(\@tempa)/\@tempdimb\relax
```

Tangent vector to the curve with correct length

```
  \L@p\dimexpr\d@X*\xP@squiglen/\@tempdimb\relax
  \U@p\dimexpr\d@Y*\xP@squiglen/\@tempdimb\relax
  \R@p\dimexpr\L@p*543339720/1311738121\relax
  \D@p\dimexpr\U@p*543339720/1311738121\relax
  \X@min\dimexpr\L@p*362911648/967576667\relax
  \Y@min\dimexpr\U@p*362911648/967576667\relax
  \X@max\dimexpr(\L@p+\xP@squigsign\U@p)*173517671/654249180\relax
  \Y@max\dimexpr(\L@p-\xP@squigsign\U@p)*173517671/654249180\relax
}
```

\xP@dosquiggle •7

```
\@ifdefinable\xP@dosquiggle@\relax
```

\xP@dosquiggle@ •7

```
\newcommand*\xP@dosquiggle@{%
  \edef\next@{\xP@coor{\xP@posX}{\xP@posY}m
    \xP@coor{\xP@posX+\Y@max}{\xP@posY+\xP@squigsign\X@max}%
  }%
  \let\xP@dosquiggle\xP@dosquiggle@@
}
```

\xP@dosquiggle@@ •7

```
\newcommand*\xP@dosquiggle@@{%
  \xP@append\@temptokena{\next@\expandafter\xP@coor
    \ifcase\count@
      {\xP@posX-\Y@max}{\xP@posY-\xP@squigsign\X@max}%
      \xP@coor\xP@posX\xP@posY
    \or
      {\xP@posX-\xP@squigsign\D@p-\X@min}{\xP@posY+\xP@squigsign\R@p-\Y@min}%
      \xP@coor{\xP@posX-\xP@squigsign\D@p}{\xP@posY+\xP@squigsign\R@p}%
    \or
      {\xP@posX-\X@max}{\xP@posY+\xP@squigsign\Y@max}%
      \xP@coor\xP@posX\xP@posY
    \or
      {\xP@posX+\xP@squigsign\D@p-\X@min}{\xP@posY-\xP@squigsign\R@p-\Y@min}%
      \xP@coor{\xP@posX+\xP@squigsign\D@p}{\xP@posY-\xP@squigsign\R@p}%
    \fi c }%
  \edef\next@{\expandafter\xP@coor
    \ifcase\count@
```

```
        {\xP@posX+\Y@max}{\xP@posY+\xP@squigsign\X@max}%
      \or
        {\xP@posX-\xP@squigsign\D@p+\X@min}{\xP@posY+\xP@squigsign\R@p+\Y@min}%
      \or
        {\xP@posX+\X@max}{\xP@posY-\xP@squigsign\Y@max}%
      \or
        {\xP@posX+\xP@squigsign\D@p+\X@min}{\xP@posY-\xP@squigsign\R@p+\Y@min}%
      \fi
  }%
}
```

`\xP@splinebrokensquiggled`

```
\newcommand*\xP@splinebrokensquiggled{%
  \xP@checkspline\xP@splinebrokensquiggled@\z@}
```

`\xP@splinebrokendblsquiggled`

```
\newcommand*\xP@splinebrokendblsquiggled{%
  \xP@checkspline\xP@splinebrokensquiggled@\xP@doublestroke}
```

`\xP@splinebrokentrblsquiggled`

```
\newcommand*\xP@splinebrokentrblsquiggled{%
  \xP@checkspline\xP@splinebrokensquiggled@\xP@trblstroke}
```

`\xP@splinebrokensquiggled@` •1 •7

```
\newcommand*\xP@splinebrokensquiggled@[1]{{%
      \xP@inibigdim
```

Start with a space if the last squiggle from the previous segment makes it necessary.

```
      \xP@testcont\xP@brokensquigglemacro
      \ifxP@splinecont
        \let\xP@squigsign\@firstoftwo
      \else
        \let\xP@squigsign\@secondoftwo
      \fi
      \xP@savec
```

Expected squiggle length

```
      \@tempcnta\numexpr(\@tempdimb\xP@squigsign{}{+2*\xybsqll@})%
        /(4*\xybsqll@)\relax
      \ifnum\@tempcnta<\@ne\@tempcnta\@ne\fi
      \@tempdima\dimexpr\@tempdimb/(8*\@tempcnta\xP@squigsign{}{-4})\relax
      \xP@squiglen\@tempdima
```

Make a list of dot positions on the spline segment.

```
      \xP@temppar\z@
      \xP@squigsign{\xP@slide\xP@slide\xP@slide\xP@slide}{}%
      \count@\z@
      \toks@{}%
      \loop
        \xP@append\toks@{\noexpand\xP@paintbrokensquiggle{\the\xP@temppar}}%
        \xP@slide
        \xP@append\toks@{{\the\xP@temppar}}%
        \xP@slide
        \xP@append\toks@{{\the\xP@temppar}}%
        \xP@slide
        \xP@append\toks@{{\the\xP@temppar}}%
        \xP@slide
```

```
        \xP@append\toks@{{\the\xP@temppar}}%
        \xP@lastpar\xP@temppar
        \advance\count@\@ne
      \ifnum\count@<\@tempcnta
        \xP@slide
        \xP@slide
        \xP@slide
        \xP@slide
      \repeat
```

Convert the list of parameters to a list of PDF tokens.

```
      \@temptokena{}%
      \xP@setsolidpat
      \global\let\xP@lastpattern\xP@brokensquigglemacro
```

Draw the squiggles.

```
      \let\xP@squigsign\@empty
      \@for\@tempa:={#1}\do{\the\toks@}%
      \xP@stroke{\the\@temptokena}%
}}
```

`\xP@paintbrokensquiggle` •1 •7

```
\newcommand*\xP@paintbrokensquiggle[5]{%
  \xP@squigglevectors{#1}%
  \xP@append\@temptokena{%
    \xP@coor\xP@posX\xP@posY m %
    \xP@coor{\xP@posX+\Y@max}{\xP@posY+\X@max}%
  }%
  \xP@squigglevectors{#2}%
  \xP@append\@temptokena{%
    \xP@coor{\xP@posX-\D@p-\X@min}{\xP@posY+\R@p-\Y@min}%
    \xP@coor{\xP@posX-\D@p}{\xP@posY+\R@p}c %
    \xP@coor{\xP@posX-\D@p+\X@min}{\xP@posY+\R@p+\Y@min}%
  }%
  \xP@squigglevectors{#3}%
  \xP@append\@temptokena{%
    \xP@coor{\xP@posX-\X@max}{\xP@posY+\Y@max}%
    \xP@coor\xP@posX\xP@posY c %
    \xP@coor{\xP@posX+\X@max}{\xP@posY-\Y@max}%
  }%
  \xP@squigglevectors{#4}%
  \xP@append\@temptokena{%
    \xP@coor{\xP@posX+\D@p-\X@min}{\xP@posY-\R@p-\Y@min}%
    \xP@coor{\xP@posX+\D@p}{\xP@posY-\R@p}c %
    \xP@coor{\xP@posX+\D@p+\X@min}{\xP@posY-\R@p+\Y@min}%
  }%
  \xP@squigglevectors{#5}%
  \xP@append\@temptokena{%
    \xP@coor{\xP@posX-\Y@max}{\xP@posY-\X@max}%
    \xP@coor\xP@posX\xP@posY c %
  }%
}
```

End of the section for XY-pic's "curve" option.

```
\xyendinput
⟨/curve⟩
⟨*basic⟩
```

9.13 Spline continuation

The following code handles the spline continuation (see Section 6). We introduce global macros which store the last end point of a Bézier segment. If the next segment continues at exactly the same coordinates, the dash/dot/squiggle patterns recognize the continuation.

\xP@lastX
\xP@lastY
\xP@lastpattern

```
\newcommand*\xP@lastX{}
\newcommand*\xP@lastY{}
\newcommand*\xP@lastpattern{}
```

\xP@solidmacro
\xP@dotmacro
\xP@dashmacro
\xP@evensquigglemacro
\xP@oddsquigglemacro
\xP@brokensquigglemacro

```
\newcommand*\xP@solidmacro{solid}
\newcommand*\xP@dotmacro{dot}
\newcommand*\xP@dashmacro{dash}
\newcommand*\xP@evensquigglemacro{evensquiggle}
\newcommand*\xP@oddsquigglemacro{oddsquiggle}
\newcommand*\xP@brokensquigglemacro{brokensquiggle}
```

\xyinside@ Reset the last position with every new diagram.

```
\renewcommand*\xyinside@{%
  \global\let\xP@lastpattern\@empty
  \saveXyStyle@ \aftergroup\xycheck@end
  \setboxz@h\bgroup
  \plainxy@
  \X@c=\z@\Y@c=\z@\czeroEdge@
  \X@p=\z@\Y@p=\z@\U@p=\z@\D@p=\z@\L@p=\z@\R@p=\z@\Edge@p={\zeroEdge}%
  \X@min=\hsize\X@max=-\hsize\Y@min=\hsize\Y@max=-\hsize
  \mathsurround=\z@
  \expandafter\POS\everyxy@@
}
```

\xP@savec Save the current end point

```
\newcommand*\xP@savec{%
  \xdef\xP@lastX{\the\X@c}%
  \xdef\xP@lastY{\the\Y@c}%
}
```

\ifxP@splinecont Switch: does the next line/spline continue at the end point of the last one?

```
\@ifdefinable\ifxP@splinecont\relax
\@ifdefinable\xP@splineconttrue\relax
\@ifdefinable\xP@splinecontfalse\relax
\newif\ifxP@splinecont
```

\xP@testcont Test for `\ifxP@splinecont`

```
\newcommand*\xP@testcont[1]{%
  \xP@splinecontfalse
  \ifxP@cont
    \ifx\xP@lastpattern#1%
      \ifdim\xP@lastX=\X@p
        \ifdim\xP@lastY=\Y@p
          \xP@splineconttrue
        \fi
      \fi
    \fi
  \fi
}
```

\ifxP@cont Switch: shall the spline hack be applied?

```
\@ifdefinable\ifxP@cont\relax
\@ifdefinable\xP@conttrue\relax
\@ifdefinable\xP@contfalse\relax
\newif\ifxP@cont
```

\xypdfcontpatternon
\xypdfcontpatternoff

```
\newcommand*\xypdfcontpatternon{\xP@conttrue}
\newcommand*\xypdfcontpatternoff{\xP@contfalse}
\xP@conttrue
⟨/basic⟩
```

9.14 Color

```
⟨*color⟩
\xycatcodes
```

\xypdf-co@loaded

```
\expandafter\let\csname xypdf-co@loaded\endcsname\@empty
```

\xP@colorname
\xP@colA
\xP@colB
\xP@colC
\xP@colD

```
\@ifdefinable\xP@colorname\relax
\@ifdefinable\xP@colA\relax
\@ifdefinable\xP@colB\relax
\@ifdefinable\xP@colC\relax
\@ifdefinable\xP@colD\relax
```

\newxycolor
\xP@newxycolor

```
\xP@hook{color}{newxycolor}
\newcommand*\xP@newxycolor[2]{%
  \def\xP@colorname{#1}%
  \xP@parsecolor#2 @%
}
```

Notwithstanding the policy followed by the rest of the package that the xypdf macros are enabled at \begin{document}, we overwrite the \newxycolor macro already here with the new version. This makes it possible to define colors with \newxycolor in the preamble, even if the rest of the driver is activated later.

```
\xP@color@on
```

\xP@parsecolor

```
\@ifdefinable\xP@parsecolor\relax
\def\xP@parsecolor#1 #2 #3@{%
  \def\xP@colA{#1}%
  \def\xP@colB{#2}%
  \ifx\xP@colB\xP@gray
    \xP@newcolor\xP@colorname\xP@colA\xP@gray\newxycolor
  \else
    \xP@parsecolor@#3 @%
  \fi
}
```

\xP@parsecolor@

```
\@ifdefinable\xP@parsecolor@\relax
\def\xP@parsecolor@#1 #2 #3 #4@{%
```

```
  \def\xP@colC{#1}%
  \def\xP@colD{#2}%
  \ifx\xP@colD\xP@rgb
    \xP@newcolor\xP@colorname{\xP@colA,\xP@colB,\xP@colC}\xP@rgb\newxycolor
  \else
    \def\@tempa{#3}%
    \ifx\@tempa\xP@cmyk
      \xP@newcolor\xP@colorname{\xP@colA,\xP@colB,\xP@colC,\xP@colD}{cmyk}%
        \newxycolor
    \else
      \PackageError{xypdf}{Syntax error in \string\newxycolor}{}%
    \fi
  \fi
}
```

\xP@gray
\xP@rgb
\xP@cmyk

```
\newcommand*\xP@gray{gray}
\newcommand*\xP@rgb{rgb}
\newcommand*\xP@cmyk{cmyk}
```

\OBJECT@shape
\xP@OBJECT@shape

```
\xP@hook{color}{OBJECT@shape}
\newcommand*\xP@OBJECT@shape[1]{\DN@{shape [#1]}%
  \expandafter\let\expandafter\nextii@\csname\codeof\next@\endcsname
  \ifx\nextii@\relax\DN@{style [#1]}%
    \expandafter\let\expandafter\nextii@\csname\codeof\next@\endcsname
    \ifx\nextii@\relax\DN@{\xP@checkcolor{#1}}%
    \else\DN@{\nextii@\xyFN@\OBJECT@}%
    \fi
  \else\expandafter\addtotoks@\expandafter{\nextii@}%
    \DN@{\xyFN@\OBJECT@}%
  \fi \next@}
```

\xP@checkcolor

```
\newcommand*\xP@checkcolor[1]{%
  \@ifundefined{\string\color@\detokenize{#1}}%
    {\OBJECT@shapei[#1]}%
    {%
    \xP@append\toks@{\noexpand\xP@color{{\detokenize{#1}}}}%
    \xyFN@\OBJECT@
    }%
}
```

\xP@color

```
\newcommand*\xP@color[1]{%
  \def\preStyle@@{\addtostyletoks@{\bgroup\xP@protectedcolor#1}}%
  \def\postStyle@@{\addtostyletoks@{\egroup}}%
  \modXYstyle@
}
```

\xP@protectedcolor Somehow, \protect\color is not enough, so we use ε-TEX's way of \protected macro definitions.

```
\@ifdefinable\xP@protectedcolor\relax
\protected\def\xP@protectedcolor{%
  \@ifundefined{color}\xP@pdfcolor\color
}
```

\xP@pdfcolor

```
\@ifdefinable\xP@pdfcolor\relax
\def\xP@pdfcolor[#1]#2{%
  \edef\@tempa{#1}%
  \ifx\@tempa\xP@gray
    \DN@{\xP@graycolor{#2}}%
  \else\ifx\@tempa\xP@rgb
    \DN@{\xP@rgbcolor#2@}%
  \else\ifx\@tempa\xP@cmyk
    \DN@{\xP@cmykcolor#2@}%
  \fi\fi\fi
  \aftergroup\xP@resetcolor
  \next@
}%
```

\xP@graycolor

```
\newcommand*\xP@graycolor[1]{\xP@setcolor{#1}gG}%
```

\xP@rgbcolor

```
\@ifdefinable\xP@rgbcolor\relax
\def\xP@rgbcolor#1,#2,#3@{\xP@setcolor{#1 #2 #3}{rg}{RG}}
```

\xP@cmykcolor

```
\@ifdefinable\xP@cmykcolor\relax
\def\xP@cmykcolor#1,#2,#3,#4@{\xP@setcolor{#1 #2 #3 #4}kK}
```

\xP@newcolor

```
\newcommand*\xP@newcolor[4]{%
 \expandafter\let\expandafter\next@\csname shape [#1]\endcsname
 \ifx\next@\relax
   \@ifundefined{\string\color@#1}\relax
     {\xP@warning{xypdf}{The color `#1' is overridden by %
       \string#4}}%
   \edef\next@{\noexpand\newxystyle{#1}{\noexpand\xP@unnamedcolor{#2}{#3}}}%
 \else
   \DN@{}%
 \fi
 \next@\relax}
```

\xP@unnamedcolor

```
\newcommand*\xP@unnamedcolor[2]{\xP@color{[#2]{#1}}}
```

\xP@definecrayolacolor

```
\newcommand\xP@definecrayolacolor[2]{%
  \xP@newcolor{#1}{#2}{cmyk}\UseCrayolaColors}%
```

\xP@installCrayolaColors

```
\newcommand*\xP@installCrayolaColors{%
  \xP@definecrayolacolor{GreenYellow}{.15,0,.69,0}%
  \xP@definecrayolacolor{Yellow}{0,0,1,0}%
  \xP@definecrayolacolor{Goldenrod}{0,.1,.84,0}%
  \xP@definecrayolacolor{Dandelion}{0,.29,.84,0}%
  \xP@definecrayolacolor{Apricot}{0,.32,.52,0}%
  \xP@definecrayolacolor{Peach}{0,.5,.7,0}%
  \xP@definecrayolacolor{Melon}{0,.46,.5,0}%
```

```
\xP@definecrayolacolor{YellowOrange}{0,.42,1,0}%
\xP@definecrayolacolor{Orange}{0,.61,.87,0}%
\xP@definecrayolacolor{BurntOrange}{0,.51,1,0}%
\xP@definecrayolacolor{Bittersweet}{0,.75,1,.24}%
\xP@definecrayolacolor{RedOrange}{0,.77,.87,0}%
\xP@definecrayolacolor{Mahogany}{0,.85,.87,.35}%
\xP@definecrayolacolor{Maroon}{0,.87,.68,.32}%
\xP@definecrayolacolor{BrickRed}{0,.89,.94,.28}%
\xP@definecrayolacolor{Red}{0,1,1,0}%
\xP@definecrayolacolor{OrangeRed}{0,1,.5,0}%
\xP@definecrayolacolor{RubineRed}{0,1,.13,0}%
\xP@definecrayolacolor{WildStrawberry}{0,.96,.39,0}%
\xP@definecrayolacolor{Salmon}{0,.53,.38,0}%
\xP@definecrayolacolor{CarnationPink}{0,.63,0,0}%
\xP@definecrayolacolor{Magenta}{0,1,0,0}%
\xP@definecrayolacolor{VioletRed}{0,.81,0,0}%
\xP@definecrayolacolor{Rhodamine}{0,.82,0,0}%
\xP@definecrayolacolor{Mulberry}{.34,.9,0,.02}%
\xP@definecrayolacolor{RedViolet}{.07,.9,0,.34}%
\xP@definecrayolacolor{Fuchsia}{.47,.91,0,.08}%
\xP@definecrayolacolor{Lavender}{0,.48,0,0}%
\xP@definecrayolacolor{Thistle}{.12,.59,0,0}%
\xP@definecrayolacolor{Orchid}{.32,.64,0,0}%
\xP@definecrayolacolor{DarkOrchid}{.4,.8,.2,0}%
\xP@definecrayolacolor{Purple}{.45,.86,0,0}%
\xP@definecrayolacolor{Plum}{.5,1,0,0}%
\xP@definecrayolacolor{Violet}{.79,.88,0,0}%
\xP@definecrayolacolor{RoyalPurple}{.75,.9,0,0}%
\xP@definecrayolacolor{BlueViolet}{.86,.91,0,.04}%
\xP@definecrayolacolor{Periwinkle}{.57,.55,0,0}%
\xP@definecrayolacolor{CadetBlue}{.62,.57,.23,0}%
\xP@definecrayolacolor{CornflowerBlue}{.65,.13,0,0}%
\xP@definecrayolacolor{MidnightBlue}{.98,.13,0,.43}%
\xP@definecrayolacolor{NavyBlue}{.94,.54,0,0}%
\xP@definecrayolacolor{RoyalBlue}{1,.5,0,0}%
\xP@definecrayolacolor{Blue}{1,1,0,0}%
\xP@definecrayolacolor{Cerulean}{.94,.11,0,0}%
\xP@definecrayolacolor{Cyan}{1,0,0,0}%
\xP@definecrayolacolor{ProcessBlue}{.96,0,0,0}%
\xP@definecrayolacolor{SkyBlue}{.62,0,.12,0}%
\xP@definecrayolacolor{Turquoise}{.85,0,.2,0}%
\xP@definecrayolacolor{TealBlue}{.86,0,.34,.02}%
\xP@definecrayolacolor{Aquamarine}{.82,0,.3,0}%
\xP@definecrayolacolor{BlueGreen}{.85,0,.33,0}%
\xP@definecrayolacolor{Emerald}{1,0,.5,0}%
\xP@definecrayolacolor{JungleGreen}{.99,0,.52,0}%
\xP@definecrayolacolor{SeaGreen}{.69,0,.5,0}%
\xP@definecrayolacolor{Green}{1,0,1,0}%
\xP@definecrayolacolor{ForestGreen}{.91,0,.88,.12}%
\xP@definecrayolacolor{PineGreen}{.92,0,.59,.25}%
\xP@definecrayolacolor{LimeGreen}{.5,0,1,0}%
\xP@definecrayolacolor{YellowGreen}{.44,0,.74,0}%
\xP@definecrayolacolor{SpringGreen}{.26,0,.76,0}%
\xP@definecrayolacolor{OliveGreen}{.64,0,.95,.4}%
\xP@definecrayolacolor{RawSienna}{0,.72,1,.45}%
\xP@definecrayolacolor{Sepia}{0,.83,1,.7}%
```

```
  \xP@definecrayolacolor{Brown}{0,.81,1,.6}%
  \xP@definecrayolacolor{Tan}{.14,.42,.56,0}%
  \xP@definecrayolacolor{Gray}{0,0,0,.5}%
  \xP@definecrayolacolor{Black}{0,0,0,1}%
  \xP@definecrayolacolor{White}{0,0,0,0}%
}
\xywithoption{crayon}{%
  \xP@installCrayolaColors
  \renewcommand*\installCrayolaColors@{}%
}
```

End of the section for XY-pic's "color" option.

```
\xyendinput
⟨/color⟩
```

9.15 Frames

```
⟨*frame⟩
\xyrequire{curve}%
\xycatcodes
```

`\xypdf-fr@loaded`

```
\expandafter\let\csname xypdf-fr@loaded\endcsname\@empty
```

`\xP@framedrop`

```
\newcommand*\xP@framedrop[1]{%
  \expandafter\frmDrop@\expandafter{%
  \expandafter\def\expandafter\prevEdge@@\expandafter{\prevEdge@@}%
  \setboxz@h{#1\frmradius@@}%
  \styledboxz@}%
}
```

`\frm{-}`
`\xP@frm{-}`

```
\xP@hook{frame}{frm{-}}
\expandafter\newcommand\expandafter*\csname xP@frm{-}\endcsname{%
  \xP@framedrop\xP@solidframe
}
```

`\xP@solidframe`

```
\newcommand*\xP@solidframe[1]{%
  \R@#1\relax
  \xP@setsolidpat
  \let\xP@fillorstroke\xP@stroke
  \xP@frameifnotzero\xP@oval
}
```

`\xP@ifzerosize`

```
\newcommand*\xP@ifzerosize[2]{%
  \@tempswafalse
  \ifdim\L@c=\z@\ifdim\R@c=\z@\ifdim\U@c=\z@\ifdim\D@c=\z@
    \@tempswatrue
  \fi\fi\fi\fi
  \if@tempswa#1\else#2\fi
}
```

\xP@frameifnotzero

```
\newcommand*\xP@frameifnotzero[1]{%
  \setboxz@h{%
    \hskip\X@c\raise\Y@c\hbox{\xP@ifzerosize{}{#1}}%
  }%
  \wd\z@\z@\ht\z@\z@\dp\z@\z@
  \boxz@
}
```

\xP@oval

```
\newcommand*\xP@oval{%
  \hskip-\L@c
  \lower\D@c\hbox{%
    \dimen@\dimexpr\L@c+\R@c\relax
    \dimen@ii\dimexpr\U@c+\D@c\relax
    \R@\xP@min\R@{.5\dimen@}%
    \R@\xP@min\R@{.5\dimen@ii}%
```

Circumference. $696621973/405764219 \approx 8-2\pi$

```
    \@tempdimb\dimexpr2\dimen@+2\dimen@ii-\R@*696621973/405764219\relax
    \ifdim\R@=\z@
```

Sharp corners: draw a rectangle.

```
        \xP@fillorstroke{0 0 \xP@coor\dimen@\dimen@ii re}%
      \else
```

Rounded corners.

```
      \def\@tempa{*119763188/267309217}%
      \xP@fillorstroke{%
         \xP@dim\R@0 m \xP@dim{\R@\@tempa}0 0 \xP@dim{\R@\@tempa}%
         0 \xP@dim\R@ c %
         \ifdim2\R@=\dimen@ii\else
           0 \xP@dim{\dimen@ii-\R@}l %
         \fi
         0 \xP@dim{\dimen@ii-\R@\@tempa}\xP@coor{\R@\@tempa}\dimen@ii
         \xP@coor\R@\dimen@ii c %
         \ifdim2\R@=\dimen@\else
           \xP@coor{\dimen@-\R@}\dimen@ii l %
         \fi
         \xP@coor{\dimen@-\R@\@tempa}\dimen@ii
         \xP@coor\dimen@{\dimen@ii-\R@\@tempa}\xP@coor\dimen@{\dimen@ii-\R@} c %
         \ifdim2\R@=\dimen@ii\else
           \xP@coor\dimen@\R@ l %
         \fi
         \xP@coor\dimen@{\R@\@tempa}\xP@dim{\dimen@-\R@\@tempa}0 %
         \xP@dim{\dimen@-\R@}0 c h%
         }%
    \fi
  }%
}
```

\frm[o]{-}
\xP@frm[o]{-}

```
\xP@hook{frame}{frm[o]{-}}
\expandafter\newcommand\expandafter*\csname xP@frm[o]{-}\endcsname{%
  \xP@framedrop{\xP@ellipseframe\xP@setsolidpat}%
}
```

\xP@ellipseframe

```
\newcommand*\xP@ellipseframe[2]{%
  \xP@getradii{#2}%
  \xP@ifzerosize{}{%
    \def\xP@fillorstroke{#1\xP@stroke}%
    \setboxz@h{\hskip\X@c\raise\Y@c\hbox{\xP@framedellipse}}%
    \wd\z@\z@\ht\z@\z@\dp\z@\z@
    \boxz@
  }%
}
```

\frm{.}
\xP@frm{.}

```
\xP@hook{frame}{frm{.}}
\expandafter\newcommand\expandafter*\csname xP@frm{.}\endcsname{%
  \xP@framedrop\xP@rectframedotted
}
```

\xP@rectframedotted

```
\newcommand*\xP@rectframedotted[1]{%
  \R@#1\relax
  \xP@frameifnotzero{%
    \ifdim\R@=\z@
      \xP@dottedrect
    \else
      \xP@dottedoval
    \fi
  }%
}
```

\xP@dottedrect Make sure that there is a dot in every corner of the rectangle.

```
\newcommand*\xP@dottedrect{%
  \hskip-\L@c
  \lower\D@c\hbox{%
    \dimen@ii\dimexpr\U@c+\D@c\relax
    \@tempdimc\dimexpr\xP@preclw/-2\relax
    \@tempdimb\dimexpr\L@c+\R@c+\xP@preclw\relax
    \xP@contfalse
    \xP@setdottedpat
```

Draw the horizontal lines $\frac{1}{2}$ whitespace longer to eliminate inaccuracies.

```
    \dimen@\dimexpr\@tempdimb+\@tempdima/2+\@tempdimc\relax
    \xP@stroke{\xP@dim\@tempdimc0 m \xP@dim\dimen@0 l %
      \xP@coor\@tempdimc\dimen@ii m \xP@coor\dimen@\dimen@ii l}%
```

Ensure pattern continuation.

```
    \let\xP@testcont\xP@alwaysconttrue
    \@tempdimb\dimen@ii
    \xP@setdottedpat
    \dimen@\dimexpr\L@c+\R@c\relax
    \advance\dimen@ii\dimexpr\@tempdimc-\@tempdima/2\relax
    \multiply\@tempdimc\m@ne
```

Draw the vertical lines $\frac{1}{2}$ whitespace longer.

```
    \xP@stroke{0 \xP@dim\@tempdimc m 0 \xP@dim\dimen@ii l %
      \xP@coor\dimen@\@tempdimc m \xP@coor\dimen@\dimen@ii l}%
  }%
}
```

\xP@dottedoval

```
\newcommand*\xP@dottedoval{%
  \def\xP@fillorstroke{\xP@setcldottedpat\xP@stroke}%
  \xP@oval
}
```

\frm[o]{.}
\xP@frm[o]{.}

```
\xP@hook{frame}{frm[o]{.}}
\expandafter\newcommand\expandafter*\csname xP@frm[o]{.}\endcsname{%
  \xP@framedrop{\xP@ellipseframe\xP@setcldottedpat}%
}
```

\frm{--}
\xP@frm{--}

```
\xP@hook{frame}{frm{--}}
\expandafter\newcommand\expandafter*\csname xP@frm{--}\endcsname{%
  \xP@framedrop\xP@rectframedashed
}
```

\xP@rectframedashed

```
\newcommand*\xP@rectframedashed[1]{%
  \R@#1\relax
  \xP@frameifnotzero{%
    \ifdim\R@=\z@
      \xP@dashedrect
    \else
      \xP@dashedoval
    \fi
  }%
}
```

\xP@dashedrect

```
\newcommand*\xP@dashedrect{%
  \hskip-\L@c
  \lower\D@c\hbox{%
    \dimen@\dimexpr\L@c+\R@c\relax
    \dimen@ii\dimexpr\U@c+\D@c\relax
    \@tempdimb\dimen@
    \xP@contfalse
    \xP@setdashpat
    \xP@stroke{0 0 m \xP@dim\dimen@0 l %
      0 \xP@dim\dimen@ii m \xP@coor\dimen@\dimen@ii l}%
    \@tempdimb\dimen@ii
    \xP@setdashpat
    \xP@stroke{0 0 m 0 \xP@dim\dimen@ii l %
      \xP@dim\dimen@0 m \xP@coor\dimen@\dimen@ii l}%
  }%
}
```

\xP@dashedoval

```
\newcommand*\xP@dashedoval{%
  \def\xP@fillorstroke{\xP@setcldashpat\xP@stroke}%
  \xP@oval
}
```

\frm[o]{--}
\xP@frm[o]{--}

```
\xP@hook{frame}{frm[o]{--}}
\expandafter\newcommand\expandafter*\csname xP@frm[o]{--}\endcsname{%
  \xP@framedrop{\xP@ellipseframe\xP@setcldashpat}%
}
```

\frm{,}
\xP@frm{,}

```
\xP@hook{frame}{frm{,}}
\expandafter\newcommand\expandafter*\csname xP@frm{,}\endcsname{%
  \xP@framedrop\xP@frameshadow
}
```

\xP@frameshadow

```
\newcommand*\xP@frameshadow[1]{%
  \R@#1\relax
  \ifdim\R@=\z@\R@1.2pt\relax\fi
  \xP@frameifnotzero\xP@shadow
}
```

\xP@shadow

```
\newcommand*\xP@shadow{%
  \hskip\dimexpr\R@c+\R@/2\relax
  \lower\dimexpr\D@c+\R@/2\relax\hbox{%
    \def\xP@pattern{0 J 0 j []0 d}%
    \edef\xP@lw{\xP@dim\R@}%
    \xP@stroke{\xP@dim{\R@/2-\L@c-\R@c} 0 m 0 0 l 0 \xP@dim{\D@c+\U@c-\R@/2}l}%
  }%
}
```

\frm{o-}
\xP@frm{o-}

```
\xP@hook{frame}{frm{o-}}
\expandafter\newcommand\expandafter*\csname xP@frm{o-}\endcsname{%
  \xP@framedrop\xP@roundedrectframe
}
```

\xP@roundedrectframe

```
\newcommand*\xP@roundedrectframe[1]{%
  \R@#1\relax
  \ifdim\R@=\z@\R@\xydashl@\relax\fi
  \xP@frameifnotzero\xP@roundedrectangle
}
```

\xP@roundedrectangle

```
\newcommand*\xP@roundedrectangle{%
  \dimen@\dimexpr\L@c+\R@c\relax
  \dimen@ii\dimexpr\U@c+\D@c\relax
  \R@\xP@min\R@{.5\dimen@}%
  \R@\xP@min\R@{.5\dimen@ii}%
  \hskip-\L@c
  \lower\D@c\hbox{%
```

Rounded corners

```
    \@tempdimb\dimexpr2\dimen@+2\dimen@ii-\R@*696621973/405764219\relax
    \def\@tempa{*119763188/267309217}%
    \xP@setsolidpat
    \xP@stroke{%
      \xP@dim\R@0 m \xP@dim{\R@\@tempa}0 0 \xP@dim{\R@\@tempa}%
```

```
       0 \xP@dim\R@ c %
       \ifdim2\R@=\dimen@ii\else
         0 \xP@dim{\dimen@ii-\R@}m %
       \fi
       0 \xP@dim{\dimen@ii-\R@\@tempa}\xP@coor{\R@\@tempa}\dimen@ii
       \xP@coor\R@\dimen@ii c %
       \ifdim2\R@=\dimen@\else
         \xP@coor{\dimen@-\R@}\dimen@ii m %
       \fi
       \xP@coor{\dimen@-\R@\@tempa}\dimen@ii
       \xP@coor\dimen@{\dimen@ii-\R@\@tempa}\xP@coor\dimen@{\dimen@ii-\R@} c %
       \ifdim2\R@=\dimen@ii\else
         \xP@coor\dimen@\R@ m %
       \fi
       \xP@coor\dimen@{\R@\@tempa}\xP@dim{\dimen@-\R@\@tempa}0 %
       \xP@dim{\dimen@-\R@}0 c%
       }%
```

Upper and lower horizontal dashes.

```
    \xP@contfalse
    \@tempdimb\dimexpr\L@c+\R@c-2\R@\relax
    \ifdim\@tempdimb<\z@\@tempdimb\z@\fi
    \xP@setdashpat
    \ifdim\@tempdima>\z@
      \dimen@\dimexpr\@tempdimb+\R@-\@tempdima/2\relax
      \dimen@ii\dimexpr\U@c+\D@c\relax
      \xP@stroke{%
        \xP@dim{\R@+\@tempdima}0 m \xP@dim\dimen@ 0 l %
        \xP@coor{\R@+\@tempdima}\dimen@ii m \xP@coor\dimen@\dimen@ii l%
      }%
    \fi
```

Left and right vertical dashes.

```
    \@tempdimb\dimexpr\U@c+\D@c-2\R@\relax
    \ifdim\@tempdimb<\z@\@tempdimb\z@\fi
    \xP@setdashpat
    \ifdim\@tempdima>\z@
      \dimen@\dimexpr\L@c+\R@c\relax
      \dimen@ii\dimexpr\@tempdimb+\R@-\@tempdima/2\relax
      \xP@stroke{%
        0 \xP@dim{\R@+\@tempdima}m 0 \xP@dim\dimen@ii l %
        \xP@coor\dimen@{\R@+\@tempdima}m \xP@coor\dimen@\dimen@ii l%
      }%
    \fi
  }%
}
```

\frm{=}
\xP@frm{=}

```
\xP@hook{frame}{frm{=}}
\expandafter\newcommand\expandafter*\csname xP@frm{=}\endcsname{%
  \xP@framedrop\xP@dsframe
}
```

\xP@dsframe

```
\newcommand*\xP@dsframe[1]{%
  \R@#1\relax
  \xP@frameifnotzero\xP@dsoval
}
```

\xP@dsoval

```
\newcommand*\xP@dsoval{%
  \dimen@\dimexpr(\L@c+\R@c)/2\relax
  \ifdim\dimen@<\xydashh@\dimen@\xydashh@\fi
  \dimen@ii\dimexpr(\U@c+\D@c)/2\relax
  \ifdim\dimen@ii<\xydashh@\dimen@ii\xydashh@\fi
  \R@\xP@min\R@\dimen@
  \R@\xP@min\R@\dimen@ii
  \xP@setsolidpat
  \let\xP@fillorstroke\xP@stroke
  \xP@oval
  \hskip\L@c
  \advance\L@c-\xydashh@
  \advance\R@c-\xydashh@
  \advance\U@c-\xydashh@
  \advance\D@c-\xydashh@
  \advance\R@-\xydashh@
  \ifdim\R@<\z@\R@\z@\fi
  \xP@oval
}
```

\frm[o]{=}
\xP@frm[o]{=}

```
\xP@hook{frame}{frm[o]{=}}
\expandafter\newcommand\expandafter*\csname xP@frm[o]{=}\endcsname{%
  \xP@framedrop\xP@dsellframe
}
```

\frm{ee}
\xP@frm{ee}

```
\xP@hook{frame}{frm{ee}}
\expandafter\newcommand\expandafter*\csname xP@frm{ee}\endcsname{%
  \xP@framedrop\xP@dsellframe
}
```

\xP@dsellframe

```
\newcommand*\xP@dsellframe[1]{%
  \xP@getradii{#1}%
  \xP@frameifnotzero\xP@dsellipse
}
```

\xP@temppath

```
\@ifdefinable\xP@temppath\relax
```

\xP@dsellipse

```
\newcommand*\xP@dsellipse{%
  \hskip\dimexpr(\R@c-\L@c)/2\relax
  \lower\dimexpr(\D@c-\U@c)/2\relax
  \hbox{%
% Inner ellipse: true ellipse
    \advance\A@-\xydashh@
    \advance\B@-\xydashh@
    \ifdim\A@<\z@\A@\z@\fi
    \ifdim\B@<\z@\B@\z@\fi
    \def\xP@fillorstroke{\edef\xP@temppath}%
    \xP@ellipse@
% Outer curve: offset ellipse
    \xP@inibigdim
```

```
    \let\@tempa\xydashh@
    \xP@offsetellipse
    \xP@setsolidpat
    \xP@stroke{\xP@temppath\space\the\@temptokena}%
  }%
}
```

\xP@offsetellipse TeX grouping: Not necessary, it’s in an `\hbox` anyway.

```
\newcommand*\xP@offsetellipse{%
  \xP@movetotrue
  \@temptokena{}%
  \xP@offsetelliptseg\A@\z@\A@{\B@*173517671/654249180}%
    {\A@*554561898/619869377}{\B@*34221476/65864945}%
    {\A@*543339720/768398401}{\B@*543339720/768398401}%
  \xP@offsetelliptseg{\A@*543339720/768398401}{\B@*543339720/768398401}%
    {\A@*34221476/65864945}{\B@*554561898/619869377}%
    {\A@*173517671/654249180}\B@\z@\B@
  \xP@mirrorpath
}
```

\xP@mirrorpath

```
\newcommand*\xP@mirrorpath{%
  \edef\@tempa{\the\@temptokena\relax\space\space\space\space}%
  \let\@tempb\@empty
  \let\@tempc\@empty
  \expandafter\xP@mirrorpath@\@tempa
}
```

\xP@mirrorpath@

```
\@ifdefinable\xP@mirrorpath@\relax
\def\xP@mirrorpath@#1 #2 #3 #4 #5 #6 #7 {%
  \ifx\relax#4%
    \xP@append\@temptokena{\@tempb\xP@minus#1 \xP@minus#2 #3 \@tempc h}%
  \else
    \edef\@tempb{\xP@minus#6 #7 \xP@minus#4 #5 \xP@minus#1 #2 c \@tempb%
      \if#3m\else\xP@minus#1 \xP@minus#2 #3 \fi%
      \xP@minus#4 \xP@minus#5 \xP@minus#6 \xP@minus#7 }%
    \edef\@tempc{#6 \xP@minus#7 #4 \xP@minus#5 #1 \xP@minus#2 c \@tempc}%
    \expandafter\xP@mirrorpath@
  \fi
}
```

\xP@minus

```
\@ifdefinable\xP@minus\relax
\def\xP@minus#1 {\if-#1 \else\ifdim\dimexpr#1pt\relax=\z@\else-\fi#1 \fi}
```

\xP@insertbefore

```
\newcommand*\xP@insertbefore[2]{{\edef\@tempa{#1{#2\the#1}}\expandafter}\@tempa}
```

\xP@offsetelliptseg

```
\newcommand*\xP@offsetelliptseg[8]{%
  \X@p\dimexpr#1\relax
  \Y@p\dimexpr#2\relax
  \L@c\dimexpr#3\relax
  \U@c\dimexpr#4\relax
  \R@c\dimexpr#5\relax
```

```
  \D@c\dimexpr#6\relax
  \X@c\dimexpr#7\relax
  \Y@c\dimexpr#8\relax
  \xP@savepts
  \xP@a\z@
  \xP@c\xP@bigdim
  \xP@paintsolid@
}
```

\xP@getradii

```
\newcommand*\xP@getradii[1]{%
  \edef\@tempa{#1}%
  \expandafter\xP@getradii@\@tempa,\maxdimen,@%
}
```

\xP@getradii@

```
\@ifdefinable\xP@getradii@\relax
\def\xP@getradii@#1,#2,#3@{%
  \A@#1\relax
  \B@#2\relax
  \ifdim\B@=\maxdimen
    \A@\dimexpr(\L@c+\R@c)/2\relax
    \B@\dimexpr(\U@c+\D@c)/2\relax
  \fi
}
```

\frm{o}
\xP@frm{o}

```
\xP@hook{frame}{frm{o}}
\expandafter\newcommand\expandafter*\csname xP@frm{o}\endcsname{%
  \xP@framedrop{\xP@circleframe\xP@setsolidpat}%
}
```

\frm{-o}
\xP@frm{-o}

```
\xP@hook{frame}{frm{-o}}
\expandafter\newcommand\expandafter*\csname xP@frm{-o}\endcsname{%
  \xP@framedrop{\xP@circleframe\xP@setcldashpat}%
}
```

\frm{.o}
\xP@frm{.o}

```
\xP@hook{frame}{frm{.o}}
\expandafter\newcommand\expandafter*\csname xP@frm{.o}\endcsname{%
  \xP@framedrop{\xP@circleframe\xP@setcldottedpat}%
}
```

\xP@circleframe

```
\newcommand*\xP@circleframe[2]{%
  \R@#2\relax
  \def\xP@fillorstroke{#1\xP@stroke}%
  \xP@ifzerosize{%
    \ifdim\R@>\z@
      \xP@circleframe@
    \fi
  }{%
    \ifdim\R@=\z@
      \A@\dimexpr(\L@c+\R@c)/2\relax
      \B@\dimexpr(\U@c+\D@c)/2\relax
```

```
      \R@\xP@max\A@\B@
    \fi
    \xP@circleframe@
  }%
}
```

\xP@circleframe@

```
\newcommand*\xP@circleframe@{%
  \setboxz@h{\hskip\X@c\raise\Y@c\hbox{\xP@circle}}%
  \wd\z@\z@\ht\z@\z@\dp\z@\z@
  \boxz@
}
```

\frm{e}
\xP@frm{e}

```
\xP@hook{frame}{frm{e}}
\expandafter\newcommand\expandafter*\csname xP@frm{e}\endcsname{%
  \xP@framedrop{\xP@ellipseframe\xP@setsolidpat}%
}
```

\frm{-e}
\xP@frm{-e}

```
\xP@hook{frame}{frm{-e}}
\expandafter\newcommand\expandafter*\csname xP@frm{-e}\endcsname{%
  \xP@framedrop{\xP@ellipseframe\xP@setcldashpat}%
}
```

\frm{.e}
\xP@frm{.e}

```
\xP@hook{frame}{frm{.e}}
\expandafter\newcommand\expandafter*\csname xP@frm{.e}\endcsname{%
  \xP@framedrop{\xP@ellipseframe\xP@setcldottedpat}%
}
```

\frm2{.e}
\xP@frm2{.e}

```
\xP@hook{frame}{frm2{.e}}
\expandafter\newcommand\expandafter*\csname xP@frm2{.e}\endcsname{%
  \xP@framedrop\xP@dsdottedellframe
}
```

\xP@dsdottedellframe

```
\newcommand*\xP@dsdottedellframe[1]{%
  \xP@getradii{#1}%
  \xP@frameifnotzero\xP@dsdottedellipse
}
```

\xP@dsdottedellipse

```
\newcommand*\xP@dsdottedellipse{%
  \hskip\dimexpr(\R@c-\L@c)/2\relax
  \lower\dimexpr(\D@c-\U@c)/2\relax
  \hbox{%
```

Intermediate ellipse: true ellipse

```
    \@tempdima.5\xydashh@\relax
    \advance\A@-\@tempdima
    \advance\B@-\@tempdima
    \ifdim\A@<\@tempdima\A@\@tempdima\fi
    \ifdim\B@<\@tempdima\B@\@tempdima\fi
    \let\xP@normalmult\@ne
```

```
    \xP@specialellipse{\xP@splinemultdotted\xP@doublestroke}%
  }%
}
```

\xP@specialellipse

```
\newcommand*\xP@specialellipse[1]{%
  \def\@tempa{*147546029/267309217}%
  \X@p\A@
  \Y@p\z@
  \L@c\A@
  \U@c\dimexpr\B@\@tempa\relax
  \R@c\dimexpr\A@\@tempa\relax
  \D@c\B@
  \X@c\z@
  \Y@c\B@
  \xP@bezierlength
  \let\xP@testcont\xP@alwaysconttrue
  #1%
  \X@p\z@
  \Y@p\B@
  \L@c-\R@c
  \D@c\U@c
  \U@c\B@
  \R@c-\A@
  \X@c-\A@
  \Y@c\z@
  #1%
  \X@p-\A@
  \Y@p\z@
  \R@c\L@c
  \L@c-\A@
  \U@c-\D@c
  \D@c-\B@
  \X@c\z@
  \Y@c-\B@
  #1%
  \X@p\z@
  \Y@p-\B@
  \L@c-\R@c
  \D@c\U@c
  \U@c-\B@
  \R@c\A@
  \X@c\A@
  \Y@c\z@
  #1%
}
```

\xP@alwaysconttrue

```
\newcommand*\xP@alwaysconttrue[1]{\xP@splineconttrue}
```

\frm2{-e}
\xP@frm2{-e}

```
\xP@hook{frame}{frm2{-e}}
\expandafter\newcommand\expandafter*\csname xP@frm2{-e}\endcsname{%
  \xP@framedrop\xP@dsdashellframe
}
```

\xP@dsdashellframe

```
\newcommand*\xP@dsdashellframe[1]{%
  \xP@getradii{#1}%
  \xP@frameifnotzero\xP@dsdashellipse
}
```

\xP@dsdashellipse

```
\newcommand*\xP@dsdashellipse{%
  \hskip\dimexpr(\R@c-\L@c)/2\relax
  \lower\dimexpr(\D@c-\U@c)/2\relax
  \hbox{%
```

Inner ellipse: true ellipse

```
    \advance\A@-\xydashh@
    \advance\B@-\xydashh@
    \ifdim\A@<\z@\A@\z@\fi
    \ifdim\B@<\z@\B@\z@\fi
    \xP@specialellipse{\xP@splinemultdashed\xP@elldoublestroke}%
  }%
}
```

\xP@elldoublestroke

```
\newcommand*\xP@elldoublestroke{\z@,\xydashh@}
```

\xP@fill

```
\newcommand*\xP@fill[1]{\xP@literal{#1 f}}
```

\xP@fillstroke

```
\newcommand*\xP@fillstroke[1]{\xP@literal{\xP@dim{\xP@preclw/2}w 1 j 0 G #1 b}}
```

\xP@fillorstroke

```
\newcommand*\xP@fillorstroke{}
```

\frm{*}
\xP@frm{*}

```
\xP@hook{frame}{frm{*}}
\expandafter\newcommand\expandafter*\csname xP@frm{*}\endcsname{%
  \xP@framedrop{\let\xP@fillorstroke\xP@fill\xP@framefill}%
}
```

\frm{**}
\xP@frm{**}

```
\xP@hook{frame}{frm{**}}
\expandafter\newcommand\expandafter*\csname xP@frm{**}\endcsname{%
  \xP@framedrop{\let\xP@fillorstroke\xP@fillstroke\xP@framefill}%
}
```

\xP@framefill

```
\newcommand*\xP@framefill[1]{%
  \R@#1\relax
  \xP@setsolidpat
  \setboxz@h{%
    \hskip\X@c\raise\Y@c\hbox{%
```

It would appear to make more sense to examine the token register `\Edge@c` instead of `\prevEdge@@`. If I (Daniel) had done so, the code `*+<1pc>[o]\frm{*}` had given a filled circle.

This certainly makes sense, but the `dvips` driver lets the frame be drawn in accordance with the object drawn before. Hence, I copied this behavior. E. g., the following code gives a filled ellipse:

```
\xy *[o]{a} *\frm{*} \endxy
```

```
      \DN@{\rectangleEdge}%
      \ifx\next@\prevEdge@@
        \DN@{\xP@oval}%
      \else
        \DN@{\circleEdge}%
        \ifx\next@\prevEdge@@
          \ifdim\R@=\z@
            \DN@{\xP@filledellipse}%
          \else
            \DN@{\restR@max\xP@circle}%
          \fi
        \else
          \ifdim\R@=\z@
            \DN@{\xP@oval}%
          \else
            \DN@{\xP@circle}%
          \fi
        \fi
      \fi
      \next@
    }%
  }%
  \wd\z@\z@\ht\z@\z@\dp\z@\z@
  \boxz@
}
```

\xP@circle

```
\newcommand*\xP@circle{%
  \xP@ellipse\R@\R@
}
```

\xP@filledellipse

```
\newcommand*\xP@filledellipse{%
  \xP@ellipse{\dimexpr(\L@c+\R@c)/2\relax}{\dimexpr(\U@c+\D@c)/2\relax}%
}
```

\xP@framedellipse

```
\newcommand*\xP@framedellipse{%
  \xP@ellipse\A@\B@
}
```

\xP@ellipse

```
\newcommand*\xP@ellipse[2]{%
  \hskip\dimexpr(\R@c-\L@c)/2\relax
  \lower\dimexpr(\D@c-\U@c)/2\relax
  \hbox{%
    \A@#1\relax
    \B@#2\relax
    \xP@ellipse@
  }%
}
```

\xP@ellipse@

```
\newcommand*\xP@ellipse@{%
```

Perimeter, second segment

```
  \X@p\dimexpr\A@*543339720/768398401\relax
  \Y@p\dimexpr\B@*543339720/768398401\relax
  \L@c\dimexpr\A@*34221476/65864945\relax
  \U@c\dimexpr\B@*554561898/619869377\relax
  \R@c\dimexpr\A@*173517671/654249180\relax
  \D@c\B@
  \X@c\z@
  \Y@c\B@
  \xP@bezierlength
  \@tempdima\@tempdimb
```

Perimeter, first segment

```
  \X@p\A@
  \Y@p\z@
  \L@c\A@
  \U@c\dimexpr\B@*173517671/654249180\relax
  \R@c\dimexpr\A@*554561898/619869377\relax
  \D@c\dimexpr\B@*34221476/65864945\relax
  \X@c\dimexpr\A@*543339720/768398401\relax
  \Y@c\dimexpr\B@*543339720/768398401\relax
  \xP@bezierlength
  \@tempdimb4\dimexpr\@tempdima+\@tempdimb\relax
  \edef\@tempa{%
    \xP@dim\A@0 m
    \xP@coor\L@c\U@c
    \xP@coor\R@c\D@c
    \xP@coor\X@c\Y@c c %
    \xP@coor{\A@*34221476/65864945}{\B@*554561898/619869377}%
    \xP@coor{\A@*173517671/654249180}\B@
    0 \xP@dim\B@ c }%
  \@temptokena\expandafter{\@tempa}%
  \xP@mirrorpath
  \xP@fillorstroke{\the\@temptokena}%
}
```

End of the section for XY-pic's "frame" option.

```
\xyendinput
⟨/frame⟩
```

9.16 Line styles

```
⟨*line⟩
\xycatcodes
\expandafter\let\csname xypdf-li@loaded\endcsname\@empty
% Dummy file.
```

End of the section for XY-pic's "line" option.

```
\xyendinput
⟨/line⟩
⟨*basic⟩
```

Finish the package initialization. The \xywithoption commands are wrapped into \next@ so that they cannot change the catcodes for the next \xywithoption command.

```
\let\@tempa\@undefined
\let\nextii@\@undefined
\DN@{%
  \xywithoption{color}{%
  \message{Xy-pic pdf driver: ‘color’ extension support}%
  \@ifundefined{xypdf-co@loaded}{\input xypdf-co\relax}{\message{not reloaded}}%
  }%
  \xywithoption{curve}{%
  \message{Xy-pic pdf driver: ‘curve’ extension support}%
  \@ifundefined{xypdf-cu@loaded}{\input xypdf-cu\relax}{\message{not reloaded}}%
  }%
  \xywithoption{frame}{%
  \message{Xy-pic pdf driver: ‘frame’ extension support}%
  \@ifundefined{xypdf-fr@loaded}{\input xypdf-fr\relax}{\message{not reloaded}}%
  }%
  \xywithoption{line}{%
  \message{Xy-pic pdf driver: ‘line’ extension support}%
  \@ifundefined{xypdf-li@loaded}{\input xypdf-li\relax}{\message{not reloaded}}%
  }%
  \xywithoption{rotate}{%
  \message{Xy-pic pdf driver: ‘rotate’ extension support}%
  \@ifundefined{xypdf-ro@loaded}{\input xypdf-ro\relax}{\message{not reloaded}}%
  }%
}
\next@
\xyendinput
⟨/basic⟩
```

10 Changelog

v1.0 2010/03/24

Initial version

v1.1 2010/03/30

- Added support for the XY-pic “rotate” extension.
- The parts of the style file dealing with XY-pic extensions (currently “curve” and “rotate”) are only executed when those extension were loaded.
- xypdf does not give an error message when used with XY-pic options which query the Postscript drivers (e. g. “all” or “color”).
- In DVI mode, a warning is issued that the DVI file is not portable, like XY-pic does when a Postscript driver is in use.

v1.2 2010/04/08

- Improved precision and numerical stability for the offset algorithm around cusps.
- Improved slide algorithm `\xP@slide@`
- Respect `\pdfdecimaldigits` when dimensions are written to the PDF file.
- Correct continuation for dashed/dotted/squiggled curves consisting of more than one segment.
- Code cleanup

v1.3 2010/04/12

- Bug fix: No "`Extra \fi`" if `\ifpdfabsdim` is not defined.
- Bug fix: Moved the code for the spline continuation out of the optional section for curves since it is also needed for straight lines.
- Check the version of pdfTEX since `\pdfsave` is not defined prior to pdfTEX 1.40.0.
- "Troubleshooting" paragraph for TEX Live without the ε-TEX features enabled.
- Generic PDF code for the `{-}` directional object.

v1.4 2010/05/13

- Support for both plain TEX and LATEX, reorganization of the code and splitting into several files. The LATEX style file xypdf.sty has been replaced by xypdf.tex, which is recognized by XY-pic as a driver.
- Integration into the XY-pic distribution.
- Support for the "color" and "frame" extensions.
- New supported curve style `{~~}` (broken squiggled curves).

v1.5 2010/10/11

- Bug fix: Colored curves.
- Bug fix: PDF syntax in double elliptical frames.

v1.6 2011/02/09

The xypdf package was made even more frugal with dimension registers. If it is used as a LATEX package, it can now be loaded even if there is not a single free dimension register. The primary reason for this improvement is that xypdf now works with the "beamer" document class.

v1.7 2011/03/20

- The `\newxycolor` command is overwritten with the xypdf code in the preamble, not at `\begin{document}` as before, so that new colors can be defined anytime.
- Improved logic to detect zero-sized frames.
- `\newxycolor` bug fix.
- Corrected the position of the center of rotation and scaling.
- Colored frame objects.
- New section "Differences between xypdf and the `dvips` backend" in the documentation.