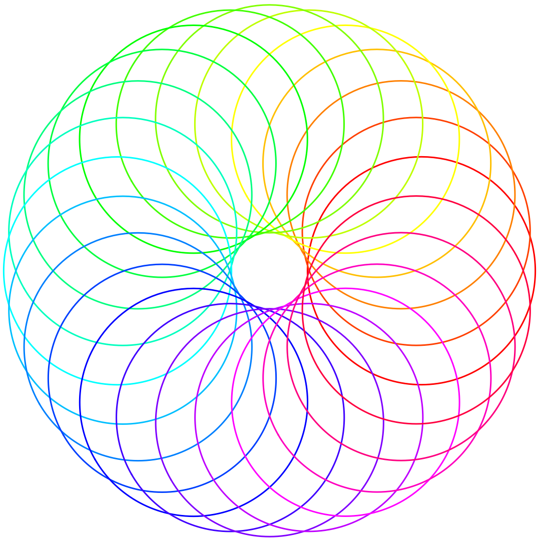
LAPDF

Drawing in TEX with PDF commands

Integral/Rational Bezier Curves
Many Useful Drawing Primitives
A Rich Set Of Math Functions
Function, Parametric & Polar Plots and Grids



Detlef Reimers, detlefreimers@gmx.de

September 1, 2011

Abstract

This package started as a project for drawing arbitrary bezier curves. It implements the integral and rational bezier curves of degree one up to seven. Using quadratic rational bezier curves makes it possible to exactly draw arbitrary conics. Also implemented are most elementary math functions as power series and two loop commands for programming constructs.

This basic functionality gives the ability to draw many geometric shapes like arbitray circles, rotated ellipses, rectangles and polygons. The package also implements function, parametric and polar plotting routines together with the possibility to draw special grids.

You are invited to extend this package to your own needs. This is not very complicated, because you can use all of the math capabilities in your TEX documents directly and so, you can program your drawings (see example above). The package also defines a rainbow color palette and you can step through the colors.

This package needs the standard LATEX `calc` package for multiplication and division of dimensions. No other packages are needed. Most of the drawing primitives rely on the graphic capabilities of PDF, which is mostly like PostScript without programming. With the help of the pdfTEX engine this programming possibility is back at the users hand. Now you can write your PDF-graphics directly in your pdfTEX source file.

The author hopes that LAPDF may be useful and helps, to produce beautiful and sophisticated drawings. I will be thankful for contructive criticism and help from the users to further improve this package.

Contents

1 Introduction

1.1 What is it for?

LAPDF is intended as a practical extension to the standard LATEX `picture` environment. It gives the user the ability to quickly draw many useful geometric shapes like arbitrary lines, circles, rotated ellipses, arcs, vectors, rectangles, triangles and equilateral polygons.

You can also draw bezier curves of degree one to seven. You may choose the integral bezier form or the more general rational form. The letter one enables you to exactly draw all kind of conic curves like parabolas, hyperbolas, ellipses and circles. There is no need to approximate them with quadratic or cubic bezier curves.

This package also implements a rich set of basic math functions like: `Sin`, `Cos`, `Tan`, `Asin`, `Acos`, `Atan`, `Sinh`, `Cosh`, `Tanh`, `Asinh`, `Acosh`, `Atanh`, `Exp`, `Pow`, `Ln`, `Log`, `Sqrt`, `Hypot`, `Len`. They can be used in your documents to program special plot functions or in loop constructs in connection with drawing functions. Al these functions are based on the ability to do floating point multiplication and division with dimension values and with dimension registers. These calculations are provided by the help of the `calc` package, which belongs to the standard LATEX tools.

1.2 How does it work?

TEX and also LATEX were made to typeset arbirary complicated and structured text documents, with the special ability to integrate math formulas into the document. `Donald Knuth`, the inventor of this marvellous typesetting machine, only provided a small peephole to implement inline graphics in TEX documents. The basic shape of these drawing capabilities is the `dot`. If you want to draw a line with pure TEX commands, you have to draw lots of evenly spaced dots. This uses much of TEX memory and it takes a lot of time to place them. `Leslie Lamport`, the inventor of the LATEX macro package, used this technique together with some specially designed graphic fonts for his implementation of the `picture` environment, which allows to produce simple to moderate comlicated graphics from inside the LATEX document. This approach is nice for simple drawings, but you are limited to only a small set of line or vector slopes, you only can draw circles of limited radii and so on. This is the price for general portability, because the line and circle fonts are limited to special drawing primitives.

Other new drawing packages avoid these limitations by using the `Postscript` programming language for calculations and drawing. With the help of `DVIPS` you can produce arbitrary sophisticated drawings to be integrated into you LATEX file. But you are limited to output devices which can handle postscript.

So, there were other approaches to implement more general drawing capabilities. The first one uses so called `Tpic` specials. `Tpic` is a simple graphic language from Unix systems, which was used together with `Groff` to produce documents with inline graphics. Another approach was done by `Eberhard Matthes` in his `EmTeX` distribution. He implemented a small but powerful set of graphic primitives into his DVI drivers, which could set points, draw arbitrary lines and set the linewidth.

LAPDF evolved out of my former `Ladraw` style, which had very similar capabilities, but no way to fill or clip graphics. LAPDF uses the `PDF` capabilities directly, so it does not rely on other packages or tools with the exception of the `calc` style to perform floating point arithmetic in TEX.

The ability to draw arbitrary lines frees us from the use of single dots as basic drawing primitve. We don't need to calculate the number of dots to draw a smooth bezier curve, because we use straight line segments to draw them. This way, we never have white spaces in curve shapes, but we have to take care to use enough line segments to produce a smooth curve.

The Bezier curve drawing is invoked by two general calling routines `Curve` for integral curves and `Rcurve` for rational curves. As an example, the command:

```
\Curve(64)(0,0)(3,6)(6,-7)(9,0)
```

will draw an integral cubic bezier curve (4 points), consisting of 64 line segments and the command:

```
\Rcurve(96)(0,0,1)(5,8,2)(10,0,1)
```

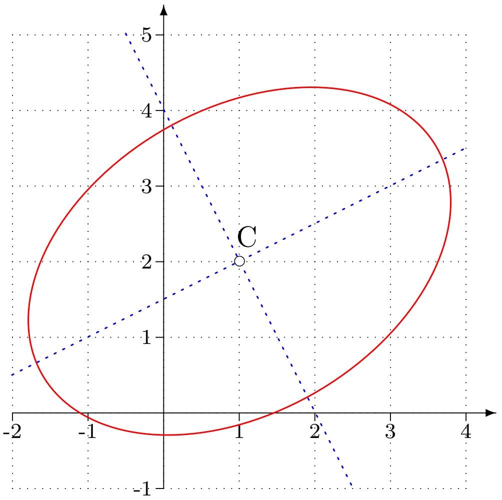
will draw a rational quadratic bezier curve (3 points) with the weights (will be explained later) $w_0 = 1$, $w_1 = 2$ and $w_2 = 1$, consisting of 96 line segments.

This package uses round brackets around command parameters with the only exception of the `Text` command, where the last argument (the actual text) cannot be bracketed with round brackets. This convention makes some functions a little more static, but you don't have to remember complicated calling syntaxes. My first decision was to make this style very easy to use. All LAPDF commands begin with a capital letter to distinguish them from other, equally spelled TEX commands. I hope that the names of the graphic macros helps guessing what they do. The standiest one seems to be `concat`, which is a native `PDF` matrix calculation commands, the others tell the user what they mean.

1.3 How to use it?

Here are some simple examples that will show you the basic usage of the LAPDF package. First, we want to draw an ellipse at (1,2), rotated by an angle of 30 deg with diameters $a = 3cm$ and $b = 2cm$ with red color and linewidth of $0.3pt$. We also want to draw two axes with tick marks. The center point will be marked and also the two main axes of the ellipse will be drawn dashed.

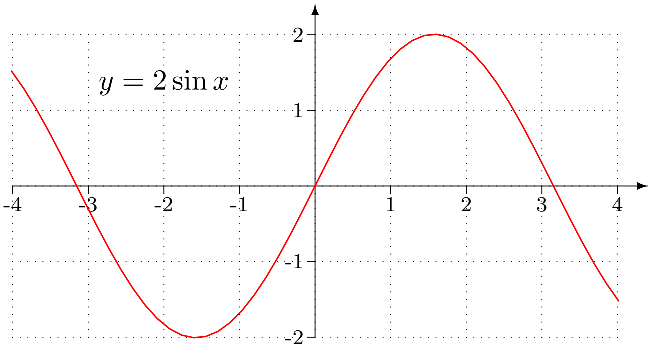
```
\documentclass{article}
\usepackage[color]{Lapdf}
\unitlength1cm
\begin{document}
\begin{lapdf}(6,6)(-2,-1)
 \Lingrid(10)(1,3)(-2,4)(-1,5)
 \Red
 \Ellipse(50)(1,2)(3,2,30) \Stroke
 \Blue
 \Dash(3)
 \Line(-2,0.5)(4,3.5) \Stroke
 \Line(-0.5,5)(2.5,-1) \Stroke
 \Dash(0)
 \Point(1)(1,2)
 \Text(1.1,2.2,cb){C}
\end{lapdf}
\end{document}
```



From this example you can see how the package is invoked in the preamble of the document. If you only want to draw in black and white, you would use the option `black` instead of `color` or simply no option. This means that all color commands will be silently ignored. The ellipse is drawn with 50 line segments in it's first part and with $2 \cdot 50 = 100$ segments in the second part. `Line` draws a line (solid or dashed) between two points. The `Point` command draws a circle filled with a shade of gray (black..white). `Puttext` puts the bracketed text at a specific point with optional position parameters (here centered and bottom).

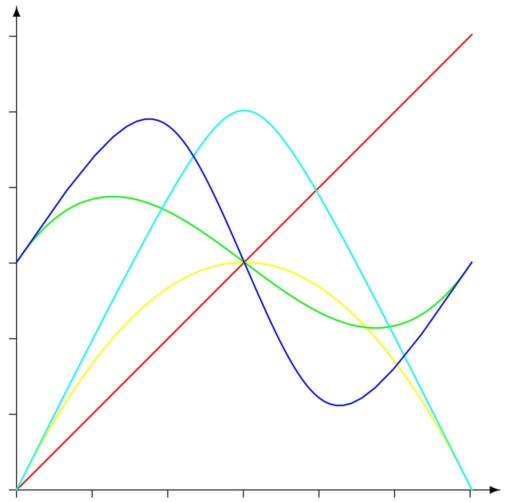
Our next example shows how to plot the function $f(x) = 2 \sin x$ from $x = -4$ to $x = 4$. We also want to write the term into the picture. We have to supply the function definition in `Fx`.

```
\documentclass{article}
\usepackage[color]{Lapdf}
\unitlength1cm
\begin{document}
\begin{lapdf}(8,4)(-7,-2)
 \Lingrid(10)(1,3)(-4,4)(-2,2)
 \Red
 \def\Fx(#1,#2){\Sin(#1,#2) #2=2#2}
 \Fplot(50)(-4,4) \Stroke
 \Text(-2,1.2,cb){$y=2\sin x$}
\end{lapdf}
\end{document}
```



The function definition has two parameters. The first is the x-value and the second is the register for the function value $f(x)$. The `Sin` function also takes two parameters, the same way as `Fx` does. So, $\sin x$ is in register `\#2`. At last we have to multiply this value with 2 to get $2 \sin x$. The function is plotted with 50 line segments. The last introductory example will draw several bezier curves (integral and rational) and axes without a grid. All curves are drawn in different colors.

```
\documentclass{article}
\usepackage[color]{Lapdf}
\unitlength1cm
\begin{document}
\begin{lapdf}(6,6)(0,0)
 \Lingrid(10)(0,2)(0,6)(0,6)
 \Stepcol(0,23,4)
 \Curve(32)(0,0)(6,6) \Stroke
 \Stepcol(0,23,4)
 \Curve(64)(0,0)(3,6)(6,0) \Stroke
 \Stepcol(0,23,4)
 \Curve(64)(0,3)(2,6)(4,0)(6,3) \Stroke
 \Stepcol(0,23,4)
 \Rcurve(64)(0,0,1)(3,6,5)(6,0,1) \Stroke
 \Stepcol(0,23,4)
 \Rcurve(64)(0,3,1)(2,6,10)(4,0,10)(6,3,1)
 \Stroke
\end{lapdf}
\end{document}
```



The grid is disabled with the first 0 value in `Lingrid`. The next value 2 only shows ticks mark without text. The first three curves are integral bezier curves with degree 1, 2 and 3. The next two are rational bezier curves with degree 2 and 3. All curves are drawn with 64 line segments. The rational form needs a so called `weight` parameter. Usually, the first and the last values are set to 1, the other may have arbitrary values. If $w > 1$, the curve is pulled into the direction of the bezier point, otherwise it is pushed away from this point. This allows much more control over the shape of the bezier curve, compared to the integral form. The `Stepcol` command cycles between color 0 and color 23 in step of length 4. Please look at the file `colors.tex` to see a whole color circle with all possible 96 colors.