The PracTEX Journal, 2010, No. 1

Useful Vector Graphic Tools for LaTeX Users

T. Morales de Luna

Email Tomas.Morales@uco.es
Address Dpto. de Matemáticas
Escuela Politécnica Superior
Universidad de Córdoba
Córdoba 14071
Spain

Abstract This paper aims to present some useful tools to create vector graphics that can be included in LaTeX documents. Among all the tools available, we focus on those that can produce graphics in an easy way and that can include any formula in the same way you type them in your documents. In particular, we present here three useful tools: `Xfig`, `LaTeXDraw`, and `Matplotlib`. While the two first are intended to produce sketches and figures, the last one will help us to produce graphs, charts and contours.

1. Introduction

In this article, we describe some vector graphic tools that work with LaTeX code in order to help common users to produce *good* graphics to be included in LaTeX documents.

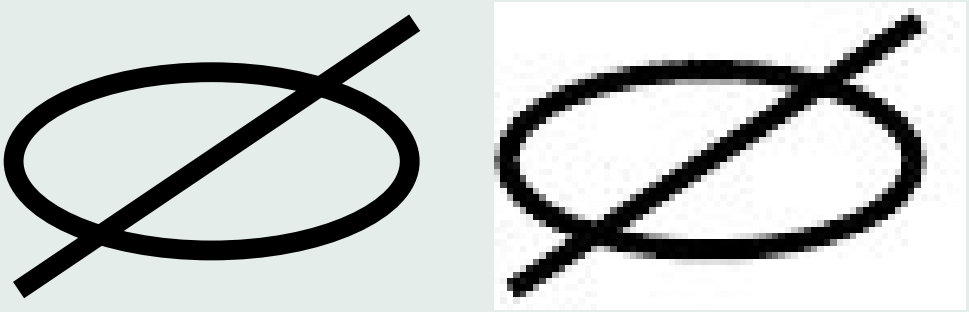
2. Including graphics into LaTeX docs

Although this document focus on tools for creating graphics rather than packages for including graphics in LaTeX documents, it is important to say few words on how to include graphics in `tex` documents. For detail, please read [1] and [2].

First of all, it is important to recall the difference between vector and bitmap graphics. While vector graphics behave well for scaling and rotation without loss of quality, the same is not true with bitmap graphics. So, whenever possible, we will use vector graphics for our documents.

Once you have obtained your figures, you can easily include them in your document by using the package `graphicx`. Just include in the preamble of your `tex` document

```
\usepackage{graphicx}
```



(a) Vector graphic (b) Bitmap graphic

Figure 1: Difference between vector and bitmap graphics.

or if you are going to produce a `pdf` file with `pdflatex`, then use `\usepackage[pdftex]{graphicx}`

Now, you can include a figure by writing

```
\includegraphics[options]{myfigure}
\caption{mycaption}
```

For instance, Figure 1 presents two subfigures put side by side. It was possible because the code

```
\usepackage{subfigure}
```

was included, which can be eventually useful. Here is the whole code:

```
\begin{figure}
  \centering
  \subfigure[Vector graphic]
  {
    \includegraphics[width=0.47\linewidth]{vector}
  }
  \subfigure[Bitmap graphic]
  {
    \includegraphics[width=0.47\linewidth]{bitmap}
  }
\caption{Difference between vector and bitmap graphics}
\end{figure}
```

An interesting feature of the `graphicx` package over figures is that it lets you scale, rotate, trim etc. For further details, please refer to [1] and [3].

Other interesting TEX macro package for generating graphics is `pgf`. It is platform- and format-independent and works together with the most important TEX back-end drivers, including `pdftex` and `dvips`. It comes with a user-friendly syntax layer called `TikZ`. We refer to [2] for more details.

3. Creating graphics

The main question is *How do I include formulas or TEX code in the picture?*. Well, you can use your favorite bitmap graphic editor and, when possible, one that produces vector graphics, but maybe your favorite graphics editor may fail. An answer would be you could generate two pictures: a formula using a temporary TEX file and the background picture, drawn in your bitmap editor. So, you copy the formula and paste it on the background picture. We know that this is not a good practice because of the low quality presents unless you do it with care (care means you should maintain the vector properties of the image when copying-pasting).

Another option is to generate just the picture, include it in your document and then place the formulas over it by including the commands `\pgfputat`. This command lets you place almost anything at a given absolute position. For instance, the code below will produce Figure 2.

```
\begin{pgfpicture}{0cm}{0cm}{3cm}{3cm}
\pgfputat{\pgfxy(1.5,0)}{\pgfbox[center,center]{$\int_0^\infty x dx$}}
\includegraphics[width=3cm]{vector}
\end{pgfpicture}
```

The problem is to put the formulas at the correct position, but you will find many techniques that can help you to obtain the desired result.

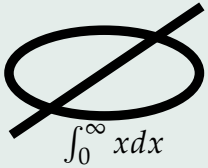


Figure 2: Using the command `\pgfputat`

Another path is you can generate the picture you would like to add directly by `Pstricks`. Thus, you have the control over your picture. Take a look at the example proposed in the `pgf` user guide that produces the Figure 3:

```
\begin{pgfpicture}{0cm}{0cm}{5cm}{2cm}
\pgfputat{\pgfxy(1,1)}{\pgfbox[center,center]{Hi!}}
\pgfcircle[stroke]{\pgfxy(1,1)}{0.5cm}
\pgfsetendarrow{\pgfarrowto}
\pgfline{\pgfxy(1.5,1)}{\pgfxy(2.2,1)}
\pgfputat{\pgfxy(3,1)}{
\begin{pgfrotateby}{\pgfdegree{30}}
\pgfbox[center,center]{$\int_0^\infty xdx$}
\end{pgfrotateby}}
```

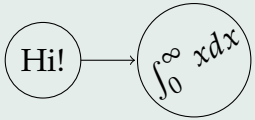


Figure 3: This example was captured from pgf user guide.

```
\pgfcircle[stroke]{\pgfxy(3,1)}{0.75cm}
\end{pgfpicture}
```

Again this can be tricky and it is in general more difficult to obtain graphics with commands rather than using your mouse.

3.1. The Xfig program

Xfig [4] is the program that can help us to obtain vector graphics combined with LaTeX formulas in an easy way. Although this program has a little bit different Graphical User Interface, it is still a handy tool.

In order to include LaTeX code inside the graphics, we have to launch the program with the *special-text* flag.

➤ Step 1. Run inside a shell the following command

```
xfig -specialtext
```

Once the xfig is opened, you can draw the picture and place in it any equation or formula with the *insert text tool*, as you would usually perform but between $ symbols. After finished, save your picture.

➤ Step 2. Draw your picture and add formulas between $ symbols.

➤ Step 3. Save the obtained figure with .fig extension.

Next, we are going to use the shell command fig2dev to produce the desired figure. Here, I assume that you used the filename myfigure.fig.

➤ Step 4. Run the following commands in the shell.

```
fig2dev -L pstex myfigure.fig > myfigure.pstex_t
fig2dev -L pstex_t -p myfigure.pstex_t myfigure.fig > myfigure.temptex
```

The first command generates `.ps` from `.fig`, and the second one generates `.tex` commands from `.fig` based on those specifications in the `.ps` file.

➤ Step 5. Create the file `myfigure.tex` with the following content

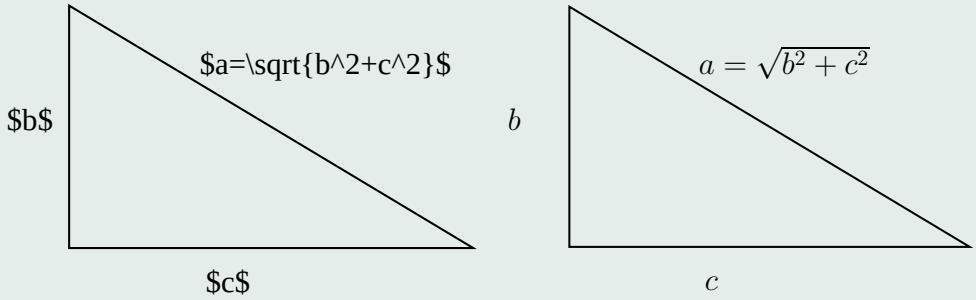
```
\documentclass{article}
\usepackage{graphicx,epsfig,color}
\pagestyle{empty}
\begin{document}
\input{myfigure.temptex}
\end{document}
```

➤ Step 6. Obtain the corresponding `.eps` file by running the following commands in a shell

```
latex myfigure.tex
dvips -E -q -o myfigure.eps myfigure.dvi
```

If you prefer a `.pdf` file, just use `epstopdf` to transform the `.eps` figure previously obtained.

You may put all the previous command inside a script or you may use the one available in [5].



(a) Plot produced with Xfig (b) Final figure obtained

Figure 4: Using `Xfig` to produce graphics

3.2. The LatexDraw program

A more recent program that will help you to produce graphics is LaTeXDraw[6]. It is a free PSTricks code generator or PSTricks editor distributed under the GNU GPL. LaTeXDraw is developed in JAVA, so it runs independently from any operating system. The Graphic User Interface of LaTeXDraw is quite similar to the one that you find in many graphics editors.

➤ Step 1. Draw your picture and include formulas between $

As what you are going to produce is intended to be part of a LaTeX document, you may place any formula with the text button available in the toolbar in LaTeXDraw. Just place the formulas between $ as you would do in your document.

➤ Step 2. Export the picture as PSTricks in a tex file.

Once you have drawn your picture, you can save the PSTricks code with the menu File → Export as → PSTricks code.

A figure similar to 4(a) will produce the following .tex file with LaTeXDraw:

```
% Generated with LaTeXDraw 2.0.3
% Tue Dec 29 12:38:16 CET 2009
% \usepackage[usenames,dvipsnames]{pstricks}
% \usepackage{epsfig}
% \usepackage{pst-grad} % For gradients
% \usepackage{pst-plot} % For axes
\scalebox{1} % Change this value to rescale the drawing.
{
\begin{pspicture}(0,-1.5507812)(6.1871877,1.5307813)
\pspolygon[linewidth=0.04](0.701875,1.5107813)(0.701875,-0.94921875)
(5.481875,-0.96921873)
\usefont{T1}{ptm}{m}{n}
\rput(0.25546876,0.22578125){$b$}
\usefont{T1}{ptm}{m}{n}
\rput(2.5754688,-1.3742187){$c$}
\usefont{T1}{ptm}{m}{n}
\rput(4.3554688,0.8857812){$a=\sqrt{b^2+c^2}$}
\end{pspicture}
}
```

Just make sure that you include the corresponding packages if you insert this code inside your document.

If you prefer to generate independent vector graphics that you can include in your document with the `\includegraphics` command, you can do so by creating the file `myfigure.tex` as follows.

➤ Step 3. Create a new `.tex` with the following content.

```
\documentclass{article}
\usepackage{pstricks}
\usepackage{pst-plot}
\usepackage{pst-eps}
\usepackage{pst-grad}
\pagestyle{empty}
\begin{document}
\begin{TeXtoEPS}
\begin{pspicture}
<...>
\end{pspicture}
\end{TeXtoEPS}
\end{document}
```

replacing <. . . > with the corresponding code generated by `LaTeXDraw`. It is important to place the `pspicture` inside the environment `TeXtoEPS`,

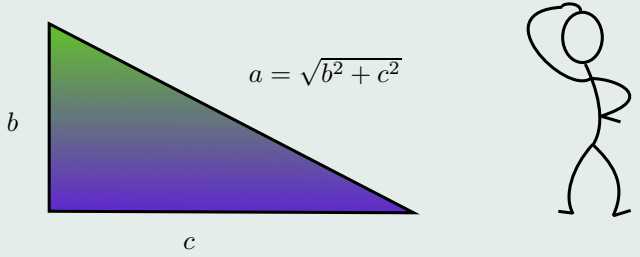


Figure 5: A figure obtained with `LaTeXDraw`.

as this will tell LaTeX the correct bounding box for the picture. If you omit it, you may get a final picture that has been trimmed or in a page size far larger than the size of the picture. You may not include all the packages that have been include above, just the ones needed for your graphic. Now, all that rest to do is

➤ Step 4. Execute the commands

```
latex myfigure.tex
dvips -E -q -o myfigure.eps myfigure.dvi
```

Now you can obtain pretty vector graphics in an easy way with any LaTeX formula like the one shown in 5.

4. Matplotlib in LaTeX docs

When writing technical documents with LaTeX, it is an usual task to plot graphs, charts, contour plots etc using Matlab, Mathematica or any other software, and then include them in your LaTeX document.

On the one hand, Matlab and Mathematica are good programs but commercial programs for fulfilling the task, on the another hand, Matplotlib [7] is a free alternative. Matplotlib is a plotting library for the Python programming language and its NumPy numerical mathematics extension. It provides an object-oriented API which allows you plot picts to be embedded into applications using generic GUI toolkits, like wxPython, Qt, or GTK. There is also a procedural pylab interface based on a state machine (like OpenGL), designed to closely resemble Matlab. The good one is that it is an open-source software.

You could use Matplotlib to generate your plots and then include them in your documents, but you can include a LaTeX package that lets you to use Python code directly in your document: python [8]. You just have to add the line `\usepackage{python}` to the preamble and then insert any Python code inside the environment python (you may need to download this package if you do not have it in your system). In this way, the images are automatically generated when you compile your .tex by including the flag -shell-escape to the command line. Thus,

the steps to follow would be.

- Step 1. Write `\usepackage{python}` before the `\begin{document}`
- Step 2. Insert any `Python` code inside the environment `python`
- Step 3. Compile your `tex` file using the command

```
pdflatex -shell-escape your_tex_file.tex
```

Let's see some examples.

By writing the following code into the file `example.tex`

```
\documentclass{article}
\usepackage[pdftex]{graphicx}
\usepackage{python}
\begin{document}
\begin{figure}
{\Huge Document including a plot}
\begin{python}
from pylab import *
x = linspace(-4*pi,4*pi,200)
plot(x,sin(x)/x)
xlim(-4*pi,4*pi)
savefig('plot-include.pdf')
print r'\includegraphics[width=0.9\linewidth]{plot-include.pdf}'
\end{python}
\caption{$y(x)=\frac{\sin(x)}{x}$}
\end{figure}
\end{document}
```

and then executing the following command, we get the pdf file corresponding to Figure 6.

```
pdflatex -shell-escape example.tex
```

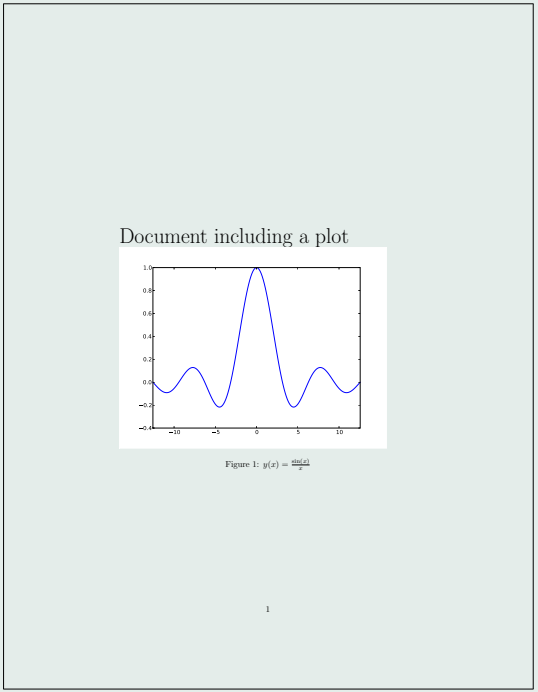


Figure 6: Plotting $y = \dfrac{\sin(x)}{x}$ with python

Many other different examples could be shown. For instance, the code

```
\begin{python}
from pylab import *
x = arange(-3.0, 3.0, .025)
y = arange(-3.0, 3.0, .025)
X, Y = meshgrid(x, y)
Z1 = bivariate_normal(X,Y,1.0,1.0,0.0,0.0)
Z2 = bivariate_normal(X,Y,1.5,0.5,1,1)
Z = 10.0 * (Z2 - Z1)
CS = contour(X, Y, Z)
clabel(CS, inline=1, fontsize=10)
savefig('plot-include2.pdf')
print r'\includegraphics[width=0.9\linewidth]{plot-include2.pdf}'
\end{python}
```

will produce Figure 7(a), and

```
\begin{python}
from pylab import *
figure(1, figsize=(6,6))
ax = axes([0.1, 0.1, 0.8, 0.8])
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
fracs = [15,30,45, 10]
explode=(0, 0.05, 0, 0)
pie(fracs, explode=explode, labels=labels, autopct='%1.1f%%', shadow=T
title('Raining Hogs and Dogs', bbox={'facecolor':'0.8', 'pad':5})
savefig('plot-include3.pdf')
print r'\includegraphics[width=0.9\linewidth]{plot-include3.pdf}'
\end{python}
```

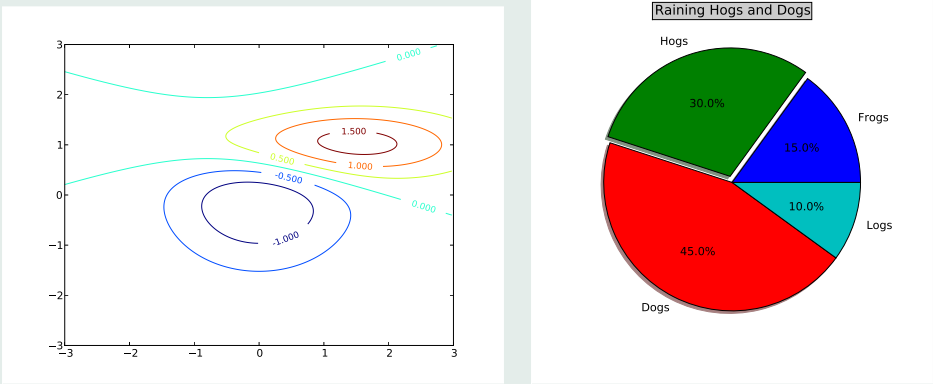
will produce Figure 7(b).

Matplotlib can even include LaTeX commands as is shown in [9].

5. Conclusion

We present many ways to include graphics in LaTeX documents, and two programs that can help you to do so: `Xfig` and `LaTeXDraw`. While both of them give good results, the second one is recent and user-friendly.

These programs give good options for doing sketches or similar figures, but in order to do graphs, charts or contour plots, `Matplotlib` would be a good choice. `Matplotlib` and `Python` code can even be embedded directly into your document.



(a) (b)

Figure 7: Using `Matplotlib` to produce graphics

References

[1] Klaus Hoeppner. *Strategies for including graphics in LaTeX documents*. The PracTEX Journal, 2005. No 3.
http://www.tug.org/pracjourn/2005-3/hoeppner/

[2] Claudio Beccari . *Graphics in LaTeX*. The PracTEX Journal, 2007. No 1.
http://www.tug.org/pracjourn/2005-3/hoeppner/

[3] *Enhanced support for graphics*
http://www.ctan.org/tex-archive/macros/latex/required/graphics/

[4] *Xfig*.
http://www.xfig.org/

[5] *Conversion tools*.
http://www.few.vu.nl/~wkager/tools.htm

[6] *LaTeXDraw*.
http://latexdraw.sourceforge.net

[7] *Matplotlib*
http://matplotlib.sourceforge.net/index.html

[8] *Embedding python into LaTeX*
http://www.imada.sdu.dk/~ehmsen/pythonlatex.php

[9] *pylab_examples example code: integral_demo.py*.
http://matplotlib.sourceforge.net/examples/pylab_examples/integral_demo.html