Getting good quality graphics inside a LaTeX document

Ignas Anikevicius

August 19, 2011

Contents

1 Introduction

For publications and scientific articles, reports and thesis the most important thing is to make well looking graphics. For that there are several guidelines, which help one to get his figures look professionally, but at the same time not to overdo things as there is a limit of how good the quality has to be.

1. Use vector graphics as much as possible. Especially where there is a lot of text in the figure. But remember, if you 'convert' a jpg to eps or svg or any other vector graphics format, you will **not** gain any quality. More on this, please use Google

2. If you have to use raster graphics, please select `.png` format as a better alternative where possible

3. If you use raster graphics, do not exceed the final resolution of the picture to more than 600dpi as most of the printers are printing at 300dpi or 600dpi, so anything more than that might just be wasted time while waiting the figure to be rendered. Of course if you have a good reason why you need more than 600dpi, then go ahead.

4. Have a high quality copy of your figure somewhere in your computer. This is because while converting from one format to another one can **not** improve the quality.

If you feel that you have not found enough information on the graphics usage in LaTeX, please refer to these websites:

- Floats, Figures and Captions
- Importing Graphics
- Creating Graphics

2 Inserting a simple figure

Inserting a figure is a really easy task and it can be accomplished with the following code. Note, that there is an option to the `\includegraphics` command which scales the image, so that its width would not exceed the text width.

```latex
\begin{figure}[H]
    \begin{center}
        \includegraphics[width=.7\textwidth]{plot1.eps}
    \end{center}
    \caption{A properly aligned figure.}
    \label{fig:plot1}
\end{figure}
```

And our figure will look as follows:

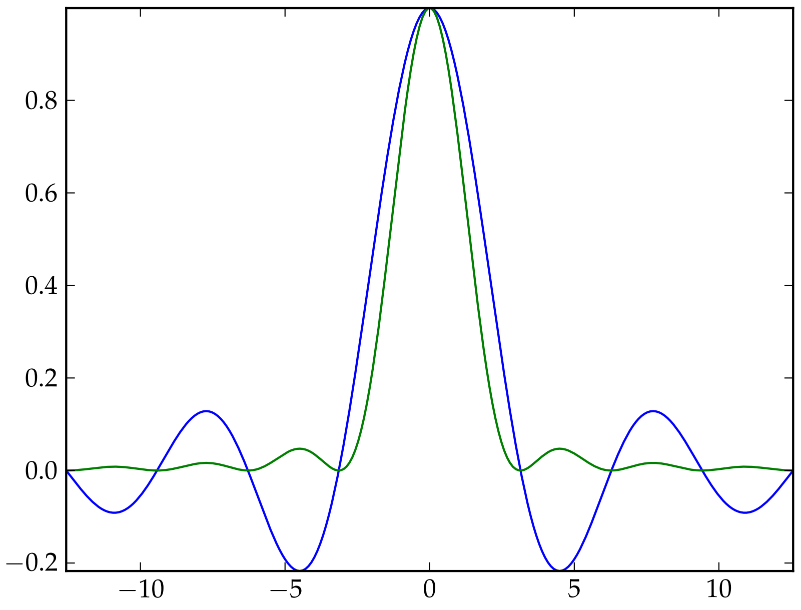

Figure 1. A properly aligned figure.

3 Directory setting for graphics, their formats and LaTeX compilers

The most convenient way nowadays to produce documents from LaTeX source code is to use the `pdftex` or `pdflatex` compiler. It is because it produces a `.pdf` file directly and you do not need to convert the `.ps` file every time before viewing it with a viewer. Also with `pdflatex` you can use more graphics formats:

`.pdf` This is the native graphics vector format for this compiler

`.eps` This format also can be used with pdflatex, but you need to use the `epstopdf` package, which will convert them to `.pdf` files on the fly. There are also options to make the conversion only when the `.eps` figures are changed, which speeds up everything substantially.

.png, .jpg At last you can use this format too if you need to import raster graphics into your document, which might ease a lot of things.

For all these features, one should use the `graphicx` package as this is the recommended way to do it. What is more, this packages provides some other useful commands. You can tell it where to search for figures by issuing the following command:

```
\graphicspath{{./path/to/your/directory}{./second/directory}}
```

The symbols `./` mean, that the top directory is where the `.tex` file is located. So if you have the following directory structure:

```
some document
        |-figures
        |-mytexfile.tex
```

you will need to issue `\graphicspath{{./figures}`. The preamble of this file contains the following lines:

```
\usepackage[pdftex]{graphicx}
\graphicspath{{./figs/}}
\usepackage[update,verbose=false]{epstopdf}
```

The options for the `epstopdf` package mean, that it will convert the `.eps` graphics only when they are updated and it will not output any errors.

4 Overlaying a figure with LaTeX code

There are many ways how to overlay LaTeX macros on top of a figure and I will present some of them in this section. These are using the native `picture` environment, whereas the other ways rely on either of `XyPic`, `overpic` and `TikZ` packages. As pure drawing tools, packages `XyPic` and `TikZ` seem to offer the most comprehensive feature-set, however, `TikZ` package is newer and overcomes some of the limitations imposed by `XyPic`. What is more, `TikZ` works better with the `pdfLaTeX` compiler as it uses internal pdf specifications.

In any case, if you got interested in any of these packages, please read the notes bellow and for further information look upon the documentation on the respective folders on CTAN.

4.1 The `picture` Environment

This is the most interesting capability of LaTeX although it should be considered as too 'fiddly' to actually be the quickest way to produce content. If you want to see an example figure and how it is made, please refer to the sources of the document, the file is named `fig2.tex`

I do not advise you to use this tool as there are several issues with it. To begin with, you need to know the aspect ratio of your picture, which is not a problem most of the times, but still is far from the quickest way of doing things. Another is that the alignment of the elements is really tricky and there are no quick macros to do such things like position text to the right of the specified point.

For more information one should either refer to the LaTeX documentation or some books such as Leslie Lamport's original book on LaTeX updated for LaTeX 2_ε.

4.2 The `overpic` Package

The `overpic` package can be considered as slightly improved version of the picture environment as it can draw a grid automatically and the position of the text and the figure itself is somewhat better. However, it is far too simple if one needs to add some graphics elements on top of the figure as it is mostly suited with overlaying other figures or text on top of the image. What is more, you still got to position everything very carefully and if you want or need more flexibility, than this package will not do the job.

To get more information please refer to the package documentation which can be found by following this link[1].

4.3 The `XyPic` Package

This is yet another alternative for producing graphics inside `.tex` documents. Since it was created more than 10 years ago, many requirements have changed and new technologies have

[1]The URL is `http://www.google.co.uk`

been created, but I believe, that there are still a lot of people using it, because it just suits their needs.

In my opinion the drawback of this package is that might not work as expected if a person uses the `latex` compiler and the does the file conversion `.dvi` →`.ps` →`.pdf`. This is because for producing graphics the package uses glyphs (i.e. to produce a long line it will use a lot of small dashes or dots and will not describe line in a similar way as vector programs do, which is defining a curve). Hence, I do not think, that it is in any way better thank the `TikZ` package, about which I have written in the next section. This said, feel free to test it and do not worry about any limitations if that does not affect your picture quality/work-flow.

To get more information please refer to the package documentation which can be found by following this link[2].

4.4 Portable Graphics Format and `TikZ` macros

Here I will describe shortly how to get the `TikZ` package working and why it might be better than the solutions described above. To begin with, Portable Graphics Format or PGF is a language for drawing quality vector graphics. It was designed after the Metapost language and is a very powerful tool for dealing with vector images. The `TikZ` package is mainly a set of macros for the PGF language and it is in a sense very similar how LaTeX is a set of macros for TeX.

Over time TikZ and PGF combination got very powerful and the documentation (v2.1) now contains over 600 pages, which is basically a lot of examples explaining different functionality in the package. If you can not find a way to do something, then please make sure, that you search inside this huge document as well.

4.4.1 Example 1: Overlaing a Simple Figure

Let's look at the example bellow, which has to do more with chemistry than the figure 1. The thing, which we want to do is to put 4 structures and then label them. The final image is shown in the figure 2, where the work-flow is explained in the figure 3 on page 8.

You can check out the code listing for creating the figure 3 bellow:

```
\begin{figure}[h]
    \centering
    \setlength{\tikzunit}{.085\textwidth}
    \begin{tikzpicture}[scale=1.0,x=\tikzunit,y=\tikzunit]
%---- Draw a grid which should help to position things ------------
%        \draw[step= .5,color=gray,thin,dashed] (-4,-4) grid (4,4);
%        \draw[step=1.0,color=gray]             (-4,-4) grid (4,4);
%        \draw[step=4.0,color=black]            (-4,-4) grid (4,4);
```

[2]The URL is `http://www.google.co.uk`

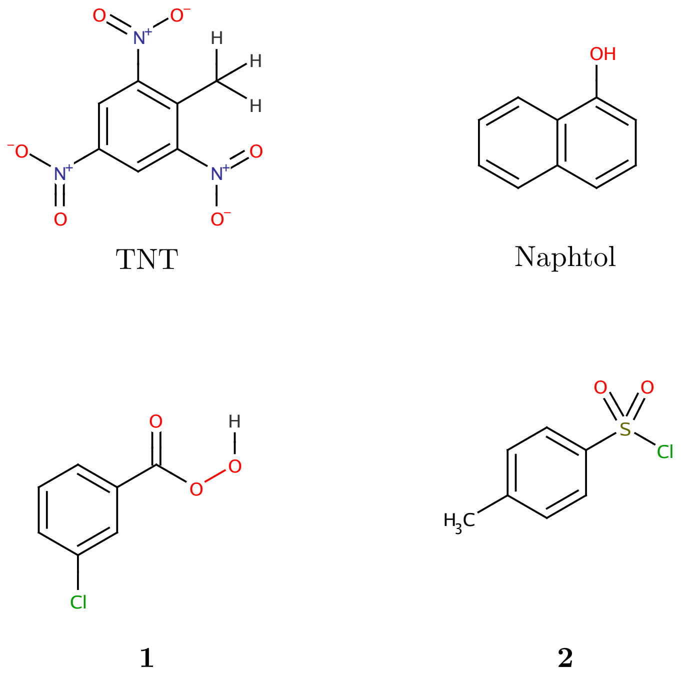

Figure 2. Overlaying LaTeX commands to get some nice things.

```
% Notes:
%       just uncomment the lines with draw commands and the grid
%       will appear. The commands, I believe are self explanatory
%       and it can be drawn as big as you want. The two coordinates
%       denote lower left and upper right corners of the grid.
%------------------------------------------------------------------
        \node(0,0){\includegraphics[width=7\tikzunit]{4struct.eps}};
        \draw (-2.1, 0.5) node{TNT};
        \draw ( 2.1, 0.5) node{Naphtol};
        \draw (-2.1,-3.5) node{\compx{1}};
        \draw ( 2.1,-3.5) node{\compx{2}};
    \end{tikzpicture}
    \caption{Overlaying \LaTeX\ commands to get some nice things.}
    \label{fig:struct1-2}
\end{figure}
```

This way one can create good figures very easily. What is more, if you have just simple structure images you do not have to worry about placing them correctly using your mouse or making sure that the font will be the same as in all the other text if you overlay everything using this LaTeX package.

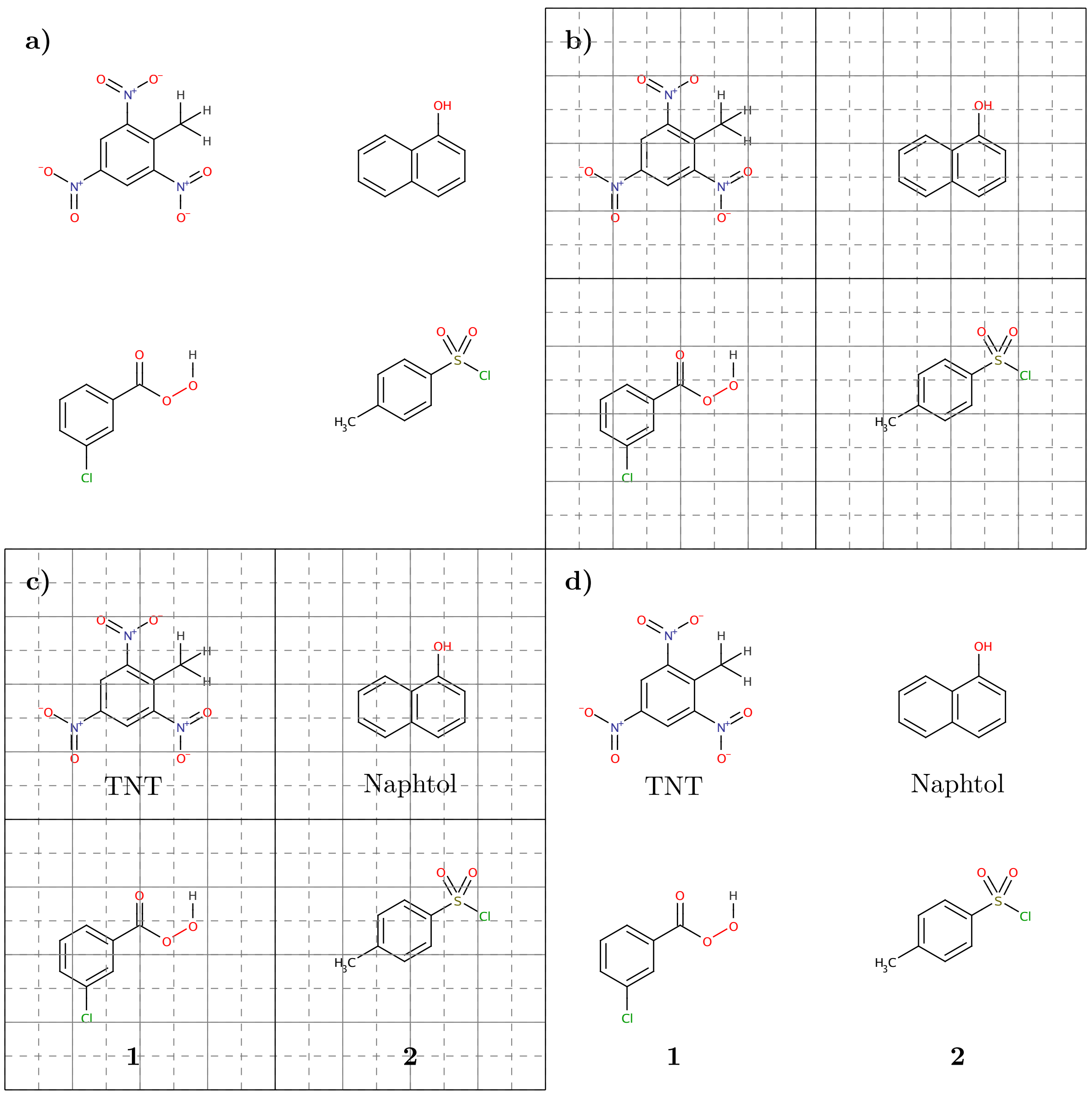

Figure 3. The workflow of placing the text on top of the figure. The 4 steps of producing good graphics are shown from the top left corner. The initial figure **a)** did not contain names and the grid was added at **b)** to help determine the position of the labels, which where placed at **c)** and then the grid was removed at **d)**. NB this illustration was done also using the very same TikZ package.

4.4.2 Example 2: Composing a Scheme of Individual Structures

In this example I have created a simple scheme, which depicts a synthesis pathway for

Figure 4. Reaction scheme

5 Substituting Postscript Code with LaTeX Macros

Another way to modify graphics is to use packages, which ease the replacement of the Postscript code into LaTeX commands, which gives you a lot of control on the looks of your figures. However, one must need to use a standard `latex` compiler to leverage this functionality, because otherwise, the `.eps` figures will not be post-processed by TeX. On the other hand, that was true only until recently when some people thought of packages, which go around these limitations and psfrag-like substitution becomes available using all compilers.

5.1 The `psfrag` package

The mechanism of how this package works is very simple. You create markers inside a `.eps` figure (e.g. TMP1, or TMP2) and then replace them with LaTeX macros by issuing the following command inside the figure environment, but before `includegraphics` command:

```
\psfrag{tag}[position][psposition][scale][rotation]{LaTeX
    construction}
```

You can deduce, that with this approach one can get the same font-faces, which are being used by LaTeX and, what is more, the font sizes are retained the same as in the .eps figure, so you are not tied to the `\normalsize` or `\footnotesize` font sizes. Although this might come handy at times, my personal opinion is that with more flexibility one can introduce more inconsistencies. Hence, one should avoid fiddling with the font-sizes a lot unless it is necessary and everything should be defined via macros such as `\normalsize`, `\tiny`, `\footnotesize` and similar.

5.2 The `pstool` package

This is one of the packages, which let you use `psfrag` type functionality without sacrificing the advantages of `pdflatex` compiler. It basically processes the figures separately before including the as already processed `.pdf` files and one can get most of the psfrag functionality this way.

Another advantage of this approach is that the figures are processes only when needed and the compilation times decrease.

5.3 The `pst-pdf` package

This package does similar thing as `pstool`, but where the first is striving to provide functionality of the `psfrag` package, the `pst-pdf` package is for dealing with `pstricks` routines. The mechanism of how it is done, is basically very similar to the approach described above.