Intermediate Latex

Using Graphics in LaTeX

Jean Hare

Sorbonne Université
Laboratoire Kastler Brossel
jean.hare@lkb.ens.fr

janvier 2021

Table des matières

Introduction

- The problem of graphics in (La)TeX is a very long lasting one.
- Some primitive attempts: the LaTeX `picture` environment (with extensions `epic`, `eepic`, `pict2e`, `bezier` , `overpic`...
- For a long time, the best solution was the creation of graphics with external software, and inclusion of `eps` pr `pdf` with `\includegraphics`.
- Several solutions working *inside* TEX emerged later. The most powerful and most widespread are:
  - PSTricks, a set of macros allowing the inclusion of PostScript drawings directly inside TeX or LaTeX code, in a `pspicture` environment. It can uses Postcript programming, but doesn't work very well with pdfTeX (several package intend to enable it). Very popular, tens of extensions. URL: `http://tug.org/PSTricks`
  - PDF/Ti*k*Zwhich is compatible with both traditional TeX/LaTeX and with pdf(La)TeX.

Sommaire

Basic Tools

- Package graphicx and its \includegraphics[options]{filename} is the basic tool to include pictures produced by an external software:
  - For pdf(La)fTeX, with PDF output, allowed formats are bitmaps JPG, PNG and vectorial/bitmap PDF. On loading graphicx don't set driver, in \includegraphics don't write extension.
  - For DVI output, formats are EPS, & bitmaps depending on distribution.
- For pdfTeX, EPS are converted to PDF by (Ghostscript-based) tools:
  - by CLI programs, like epstopdf (.exe or .pl)
  - by GUI programs, like GSView
  - package epstopdf converts on the fly the EPS files to PDF.
- Most used options of \includegraphics provide size with width= or height= (or both) scale=, trimbox with trim=left bottom right top,clip, rotation with angle= and origin=, page with page= of multi-page pdf.
- Path to files globally defined by \graphicspath{{../img/}{fig2/}}.

Other options

- The size of the graphics can be automatically adjusted with \usepackage[export]{adjustbox}.
  - It adds to \includegraphics a large number of keys; the most useful for size control are min width, max width, min height, max height, min totalheight, max totalheight.
  - It also provides \adjincludegraphics, an enhanced version of \includegraphics and many macros intended to resize any LaTeX material.
- For both \includegraphics and the macros of adjustbox, the dimensions provided to the width or other size options can be defined by using ε-TeX syntax, like, for example:
  width=\dimexpr 0.7*\linewidth-2cm\relax .
- Background images can be added by using the eso-pic package.
- Direct writing of PDF primitives can be achieved with lapdf package.
- Never use \psfig, \epsfig,\epsfbox,\epsffile

Producing figures

- For diagrams and curves, use a *vector* format and for images prefer `pdf`, or `png` by default.
- Do not use `M$-Powerpoint` for drawing, nor `M$_Excel` for plotting.
- **Vectorial drawing**
  - `Illustrator` or `Inkscape` are the references, unless you are satisfied with the basic but effective `Mayura Draw` or `Xfig`.
  - For experimental data, or simulation results, the best graphs are obtained with MATLAB or Python/matplotlib, or PFGPlots.
- **Bitmap pictures**
  - Never import a `jpeg` into Adobe Illustrator: size increase by 10 to 100 !
  - Use insteead the script `jpeg2ps`, or the more powerful `sam2p`
  - I you need to annotate it, prefer `Inkscape` or Ti*k*Z. Don't export to `jpeg`.
  - We often turn to `Photoshop`, `GIMP` and `ImageMagick`, but don't neglect everything you can do in ImageJ or IGOR Pro which are *scientific* softwares.
- Tools for Ti*k*Z are presented below.

Fonts problems in figures

The main problem with figures come from the fonts.

Main problems :

- EPS files generally do not enclose the fonts, oppositely to PDF, if they are available at creation time.
- Notably the 35 standard Postscript fonts, and their clones from `M$_Office` are, by default, never embedded.
- A missing or incorrectly encoded font can render the PDF invalid.
- Font unavailable on the end user's system and/or printer: when displayed or printed, the faulty font will be replaced by `Courier`, size 12pt, of the most beautiful effect.
- You try to use LaTeX fonts for the sake of consistency, but they are no longer available at the end ...

What ever the software, always look for the option that allows exporting fonts in the figure, and check in the properties of the resulting PDF that the fonts are embedded (`embedded subset`).

Fonts problems in figures : solution

- You can export everything in bitmap png :-(...
- lmodern fonts are provided in both Type1 and OpenType, so they can be used in any software, if you install them in the right place.
- MATLAB and Python fonts incorporate stand-alone LaTeX texts.
- In Inkscape, extension Render>>Latex Formula uses outlines.
- Better, the extension TeX text allows to keep the editable property of the embedded LaTeX formulas.
- To embed fonts *a posteriori* use the script:

```
1 gs -I "C:\Progra~1\MiKTeX\fonts\type1" \
2    -dCompatibilityLevel=1.5 -dPDFSETTINGS=/ebook \
3    -dCompressFonts=true -dSubsetFonts=true \
4    -dNOPAUSE -dBATCH -sDEVICE=pdfwrite \
5    -sOutputFile=output.pdf -f input.pdf \
6    -c ".setpdfwrite <</NeverEmbed [ ]>> setdistillerparams"
```

On line 1, adapt the path to Ghostscript and to the TeX distribution, and filenames

- In desperate cases: this post or Acrobat Pro ...

Importing SVG

- pdf(La)TeX alas does not support SVG, (internal format of Inkscape)
- Inkscape can export to useful formats: PDF, PDF+latex, PGF/Ti*k*Z.
- In the PDF+latex route, one get a pic.pdf file and pic.pdf_tex LaTeX file to \input in the main, optionally with a \resizebox. The former contains only drawings; an the latter only the text boxes, to be typeset parsed par LaTeX.
- The PDF+latex route can be automated by the code (with --shell-escape):

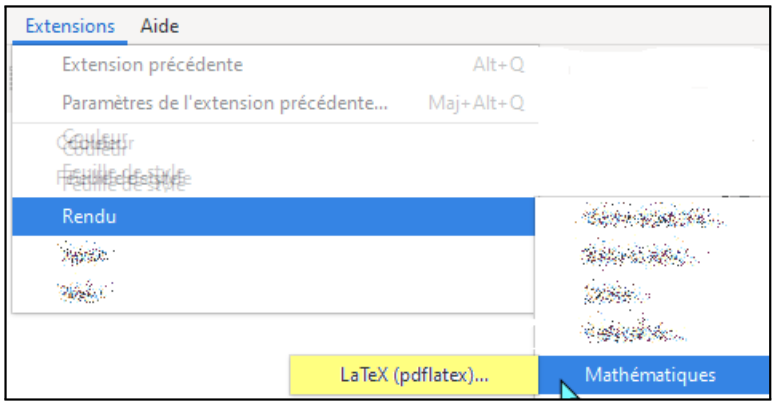
```
\def\filename{dessin-1}
\immediate\write18{inkscape -D \filename.svg -o \filename.pdf --export-latex}
\def\svgwidth{0.5\linewidth}
\IfFileExists{\filename.pdf_tex}{\input{\filename.pdf_tex}}%
{Error on \filename.svg}
```

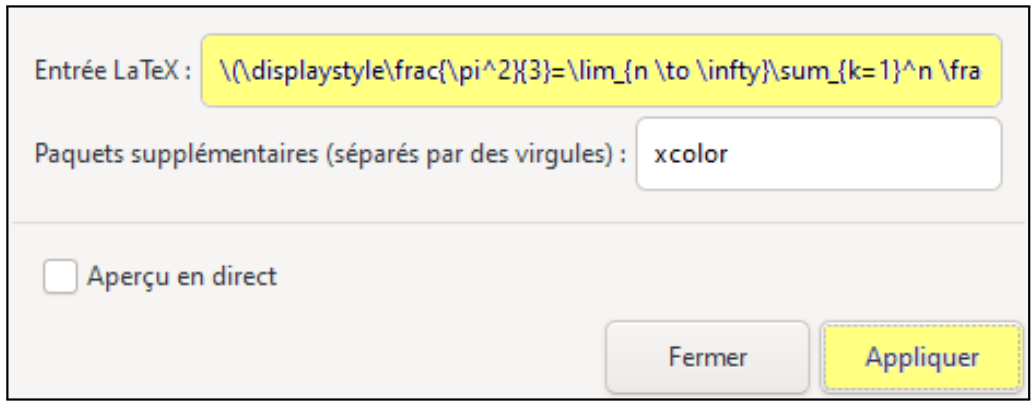
- If often used, a better solution would be to define for this purpose a macro that also performs a check for modifications date. There is the package svg which automates a lot of things at the price of complexity
- For the PDF route, one can embed outlines rendered by LaTeX but still editable, as described below.

See also How to include an SVG image in LaTeX.

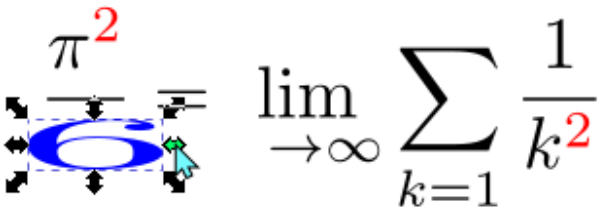
Two methods to annotate with LaTeX in Inkscape (I/II)

1. Inkscape includes an extension to typeset LaTeXtexts and mathematics.
   - Access by: `Extensions»Render»Mathematics»LaTeX`
     In French: `Extensions»Rendu»Mathematiques»LaTeX`
   - It openq a popup where LaTeX code can be entered.





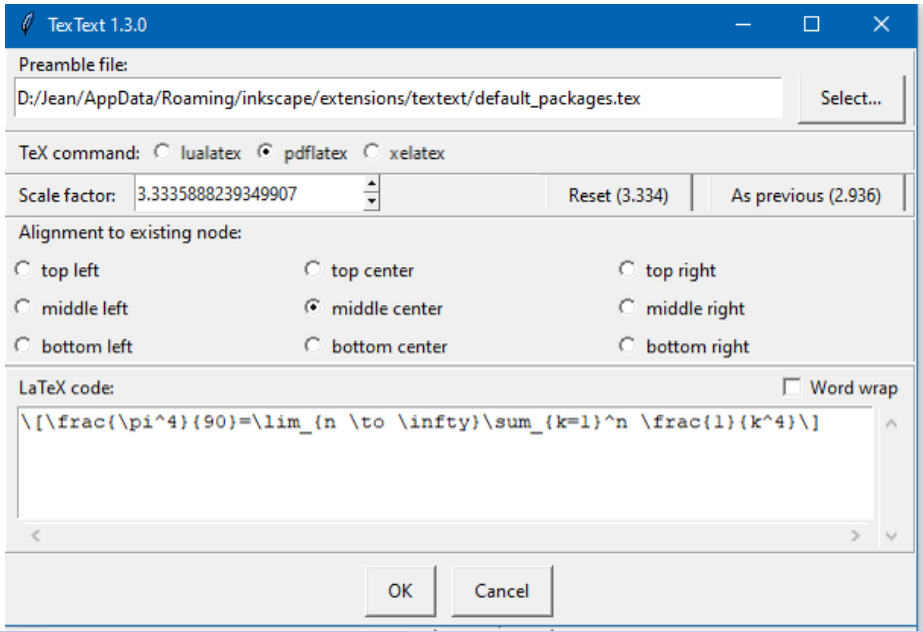
   - It calls `pdflatex` to create a vectorial version of the result, that can be edited as will like any path.



Notice that neither `$$` nor `\[...\]` can be used in the code, hence use `\(\displaystyle...\)`

Two methods to annotate with LaTeXin Inkscape (II/II)

1. With `Inkscape` 1.0.1 hand higher one can install the optional extension TeXText which does the same thing plus some extra features :
   - Enables the customization of the preamble.
   - Apply a custom scaling factor
   - Enables the use of `displaymath` delimiters $$ or better `\[...\]`
   - Remembers the previously rendered formulas, for easier editing
   - In case of errors, displays the pdflatex terminal output.
   - Once installed, access by: `Extensions»Text»Tex Text`



$$\frac{\pi^4}{90} = \lim_{n\to\infty} \sum_{k=1}^{n} \frac{1}{k^4}$$

Sommaire

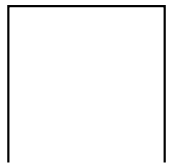
Installation and documentation

- PGF means "Portable Graphics Format" and is a low level language.
- Ti*k*Z means "TikZ ist kein Zeichen programm" (is English "TikZ is not a drawing program") and is high level language based on PGF. `\usepackage{tikz}` automatically loads PGF (huge number of files). Additional libraries must be loaded for specific tasks or decorations.
- The packages and documentation for PGF/are available directly in MiKTeX and TeXLive distributions, and can also be downloaded from https://ctan.org/pkg/pgf. The common manual (version 3.1.7a) is 1320 pages long !!
- The release contains also an 24 pages PDF entitled "Minimal Introduction to Tikz"
- A good introduction "en français" entitled "Ti*k*Z pour l'impatient" is available from http://math.et.info.free.fr/TikZ.
- There is also the package PGFPlots (plotting 2D and 3D data), based on PGF, that can be downloaded at the same time that PGF/Ti*k*Z.

First drawings

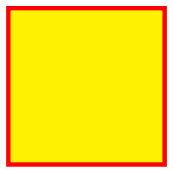
Use of \draw with coordinates (in cm) and -- to get a simple line:

```
\begin{tikzpicture}
\draw (0,0)--(0,1)--(1,1)--(1,0);
\end{tikzpicture}
```



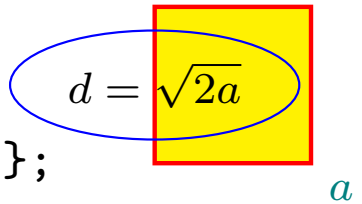
Close polygon with --cycle and add some styling:

```
\begin{tikzpicture}
\draw [color=red, fill=yellow, thick]
    (0,0)--(0,1)--(1,1)--(1,0)--cycle;
\end{tikzpicture}
```



Adding two nodes. A standalone one produced by \node, centered on coordinates (0,0.5) and exhibiting a LaTeX formula. A second one, inside \draw, with node (no \), attached to point (0,1) but offset by the option below right.

```
\begin{tikzpicture}\footnotesize
\draw [color=red, fill=yellow, thick](0,0)-- (0,1)
--(1,1)--(1,0) node[below right,teal] {$a$} --cycle;
\node[ellipse,draw=blue] (A) at(0,0.5){$d=\sqrt{2a}$};
\end{tikzpicture}
```



First plots

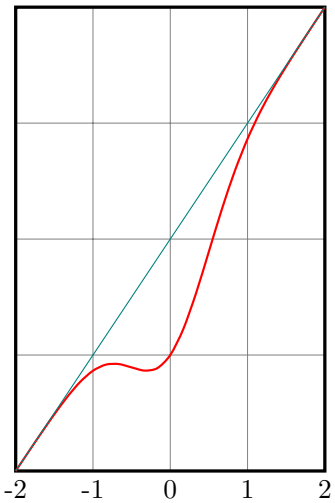
Define canvas and translation of units from cm (screen) to plot:

```
\usetikzlibrary{calc,decorations,arrows,patterns,shapes}
\tikzset{ xmin/.store in=\xmin, xmin/.default=-3, xmin=-3,
          xmax/.store in=\xmax, xmax/.default=3, xmax=3,
          ymin/.store in=\ymin, ymin/.default=-3, ymin=-3,
          ymax/.store in=\ymax, ymax/.default=3, ymax=3 }
```

Disable active characters (if babel-french loaded): `\shorthandoff{?!:;}`

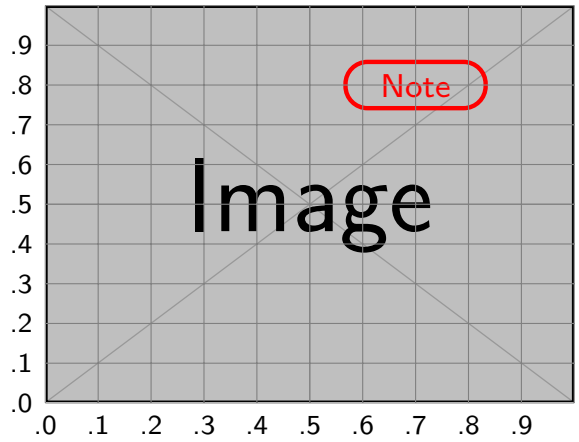
Define and plot a function:

```
\begin{tikzpicture}[xmin=-2,xmax=2,ymin=-2,ymax=2,
yscale=1.5, declare function={ff(\x)=\x-exp(-2*\x*\x);}]
\draw[help lines,xstep=1,ystep=1]
 (\xmin,\ymin) grid (\xmax,\ymax);
\foreach \x in {-2,...,2} \draw (\x,\ymin)--(\x,\ymin)
   node[below] {\x};
\draw[very thick] (\xmin,\ymin) rectangle (\xmax,\ymax);
\draw[thick,red] plot [domain=-2:2,smooth] (\x,{ff(\x)});
\draw[teal] plot [domain=\xmin:\xmax,samples=2] (\x,\x);
\end{tikzpicture}
```



Using TikZ for annotations (see also this website)

```
1 \begin{tikzpicture}
2 \node[inner sep=0pt] (img) at (0,0) {% (0,0) or anything
3    \includegraphics[width=0.9\linewidth]{example-image}};
4 \begin{scope}[shift={(img.south west), % origin and scale
5    x={(img.south east)},y={(img.north west)}] % of image
6    % temporary grids and  ticks  to  help   positioning
7    \draw[help lines,step=.1] (0,0) grid (1,1);
8       \foreach \x in {0,...,9} {
9          \node [below] at (\x/10,0) {.\x};
10         \node [left] at (0,\x/10) {.\x}; }
11   \node[annot] at (0.7,0.8) {\Large Note}; % annotation
12 \end{scope}
13 \end{tikzpicture}
```



- A node can host an `\includegraphics`. To draw on it, simply `\draw` the picture **before** the annotations (lines 2–3 & 11).
- To position the annotations, use **relative** coordinates, defined by a `scope` with a scale and origin relative to the image (lines 4–5 & 12)
- For easy positioning (temporarily) draw a grid with ticks (lines 6–10)
- Here, formating of the node is obtained by the style defined as:

```
\tikzset{annot/.style={draw,red,thick,rounded corners,minimum width=4em}}
```

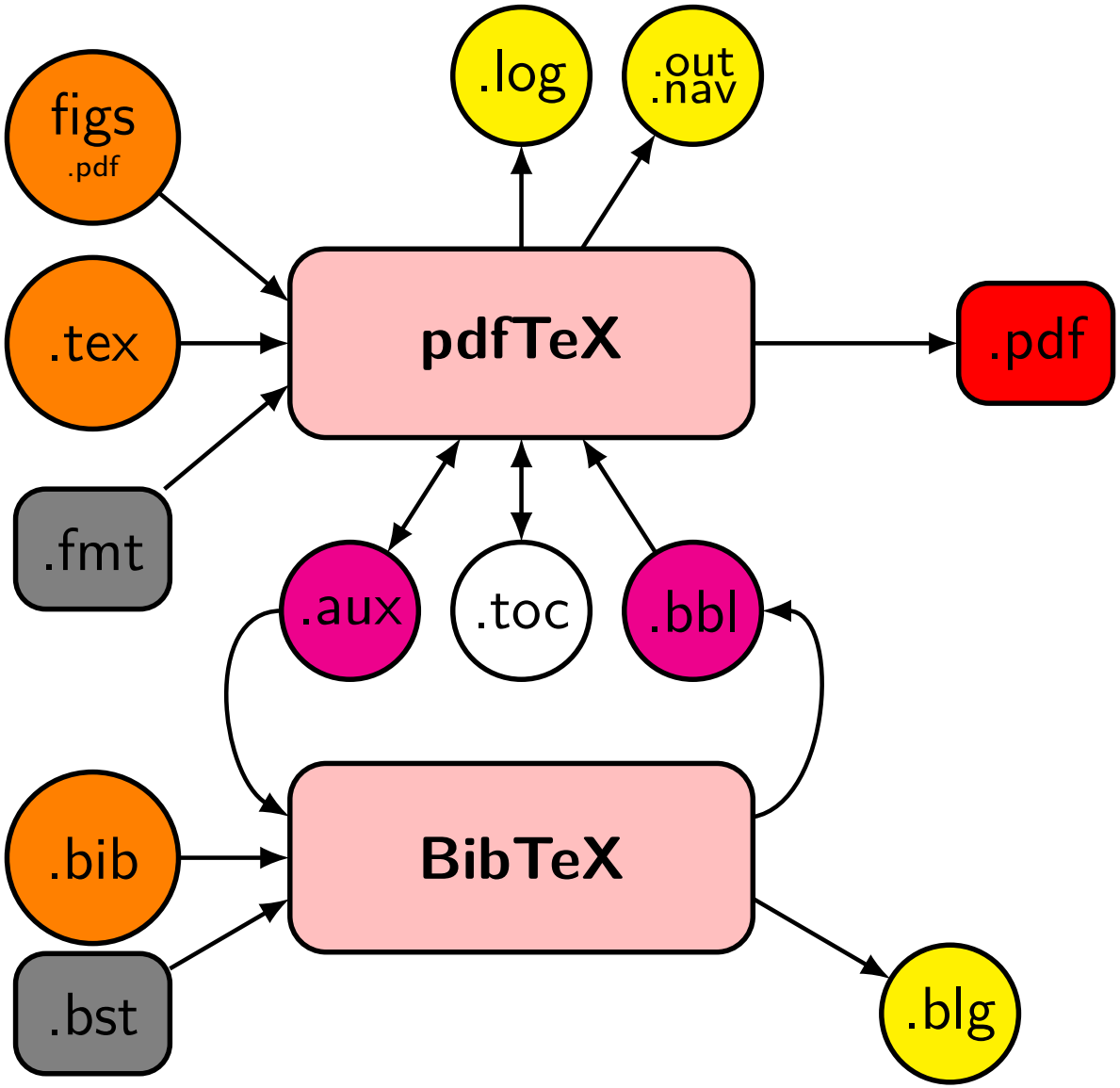
Tools making TikZ more efficient

Here a is a small list of open source softwares more or less TikZ oriented.

- **TikZiT** (multiplatfom) A GUI editor for diagrams. Its native file format is a subset of PGF/TikZ, that can be included directly in papers. Mouse-driven, provide TikZ editing. URL: https://tikzit.github.io
- **TpX** (windows only, but source available to compile on Linux/MacOSX) Another GUI drawing program that can generate good quality EPS/PDF, or can export the code as PGF/TikZ drawing. http://tpx.sourceforge.net
- **KtikZ** (on Linux, and QtikZ on windows) is a code oriented software that provides autocompletion, syntax checking, and compile in the background to show you the result in (almost) real time. Th most efficient to learn TikZ without to much effort.
- Other softwares can export PFG/TikZcode among other formats. Namely: Inkscape, Gnuplot, Matlab, etc.

The best **reference**, with a most complete and up to date list of softwares, and huge collection of nice examples is the TikZ section of the web site **TeXample**.

The compilation chain



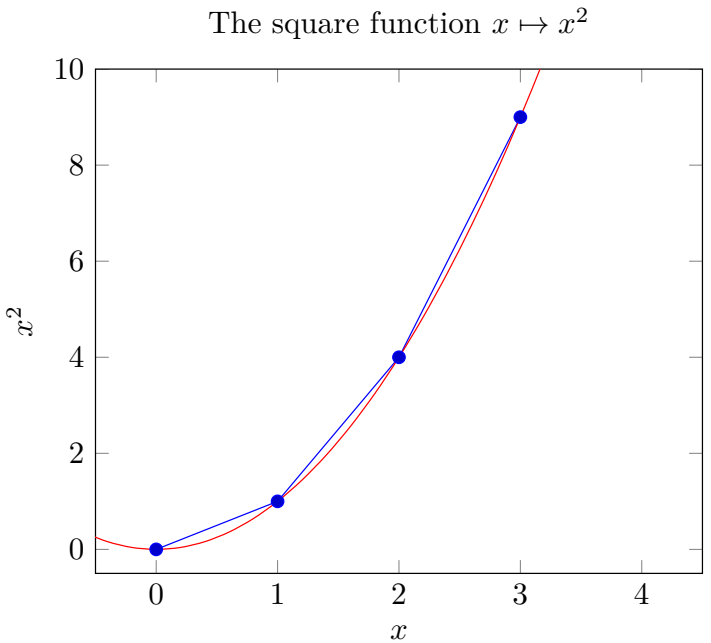
```
\tikzstyle{moteur}=[mythick,minimum width=2.7cm,minimum height=1.1cm,rectangle,rounded corners=6pt,
fill=pink,font=\bfseries\sffamily] \def\lineoffset{1.2cm}
\tikzset{mythick/.style={draw=black,line width=0.3mm,font=\sffamily,align=center}}
\tikzset{circbox/.style={circle,minimum width=10mm,minimum height=10mm}}
\tikzset{smallcircbox/.style={circle,minimum width=8mm,minimum height=8mm}}
\tikzset{rectbox/.style={rectangle,minimum width=9mm,minimum height=7mm, rounded corners=5pt}}
\tikzstyle{arrinput}=[line width=0.25mm,->,>=latex] \tikzstyle{arrex}=[line width=0.25mm,<->,>=latex]
\begin{tikzpicture}[x=0.50mm, y=0.50mm, inner sep=0pt, outer sep=0pt]
% pdflatex and  files
\node[moteur] (pdftex) at (60,0) {\hologo{pdfTeX}};
\node[mythick,circbox,fill=orange] (filetexp) at (10,0) {.tex};
\node[mythick,circbox,fill=orange] (filefigp) at (10,\lineoffset) {figs\\[-5pt]{\tiny .pdf}};
\node[mythick,rectbox,fill=gray] (filefmtp) at (10,-\lineoffset) {.fmt};
\node[mythick,rectbox,fill=red] (pdffilep) at (120,0) {.pdf};
\node[mythick,smallcircbox,fill=magenta] (auxfilep) at (40,-1.3*\lineoffset) {.aux};
\node[mythick,smallcircbox,fill=white] (tocfilep) at (60,-1.3*\lineoffset) {.toc};
\node[mythick,smallcircbox,fill=magenta] (bblfilep) at (80,-1.3*\lineoffset) {.bbl};
\node[mythick,smallcircbox,fill=yellow] (logfilep) at (60,1.3*\lineoffset) {.log};
\node[mythick,smallcircbox,fill=yellow,font=\footnotesize\sffamily]
(outfilep) at (80,1.3*\lineoffset) {.out\\[-5pt].nav};
% pdflatex arrows
\draw[arrinput] (filefmtp) -- (pdftex.190); \draw[arrinput] (filetexp.east) -- (pdftex.west);
\draw[arrinput] (filefigp) -- (pdftex.170); \draw[arrinput] (pdftex)- - (logfilep);
\draw[arrex] (auxfilep) -- (pdftex); \draw[arrex] (tocfilep) -- (pdftex);
\draw[arrinput] (bblfilep) --(pdftex); \draw[arrinput] (pdftex) --(pdffilep);
\draw[arrinput] (pdftex) --(outfilep);
% bibtex and  files
\node[moteur] (bibtexp) at (60,-60) {\hologo{BibTeX}};
\node[mythick,circbox,fill=orange] (filebibp) at (10,-60) {.bib};
\node[mythick,rectbox,fill=Gray] (filebstp) at (10,-1.5*\lineoffset-60) {.bst};
\node[mythick,smallcircbox,fill=yellow] (blgfilep) at (110,-1.5*\lineoffset-60) {.blg};
% bibtex arrows
\draw[arrinput] (auxfilep.west) to [out=180, in=150] (bibtexp.170);
\draw[arrinput] (filebibp.east) -- (bibtexp.west); \draw[arrinput] (filebstp) -- (bibtexp.190);
% linking  arrows
\draw[arrinput] (bibtexp.10) to [out=10, in=0] (bblfilep.east); \draw[arrinput] (bibtexp.350) -- (blgfilep);
\end{tikzpicture}
```

Sommaire

PGFPlots

- PGFPlotsis a very powerful package, based on PGF, for the graphic representation of data or functions, in 2 or 3 dimensions.
- It has a very big documentation file (569 pages), but some good tutorials are available , including those included in the manual itself.
- The most basic use of PGFPlots is as follows:

```
\begin{tikzpicture}
\begin{axis}[
 title={The square function $x\mapsto x^2$},
 xlabel={$x$}, ylabel={$x^2$},
 xmin=-0.5, xmax=4.5,
 ymin=-0.5, ymax=10, yscale=1 ]
 \addplot coordinates
  {(0,0) (1,1) (2,4) (3,9) };
 \addplot [red,samples=51,smooth] {x^2};
\end{axis}
\end{tikzpicture}
```

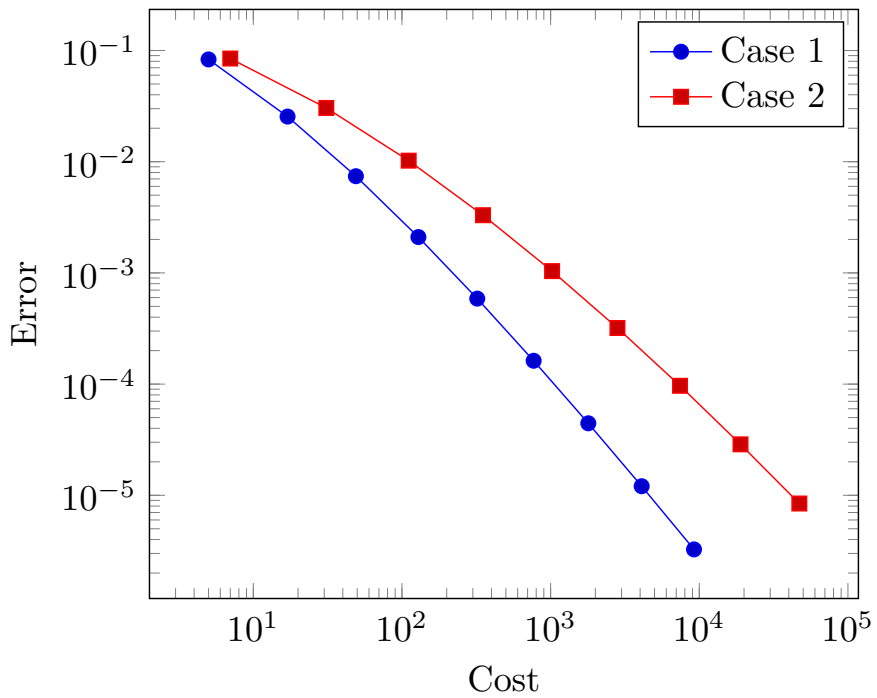


This example shows how to plot data points or a function, set label for axes and plot.

Log scale, legend

One can also uses log/log or semi-log axes, add automatic legend, etc (example from the manual):

```
\begin{tikzpicture}
 \begin{loglogaxis}[xlabel=Cost,
 ylabel=Error]
 \addplot coordinates {
  (5,    8.311e-02) (17,  2.546e-02)
  (49,   7.407e-03) (129, 2.10e-03)
  (321,  5.873e-04) (769, 1.622e-04)
  (1793, 4.442e-05) (4097, 1.207e-05)
  (9217, 3.261e-06) };
 \addplot coordinates {
  (7,    8.471e-02) (31,  3.044e-02)
  (111,  1.022e-02) (351, 3.303e-03)
  (1023, 1.038e-03) (2815, 3.196e-04)
  (7423, 9.657e-05) (18943, 2.873e-05)
  (47103, 8.437e-06) };
 \legend{Case 1,Case 2}
 \end{loglogaxis}
\end{tikzpicture}
```



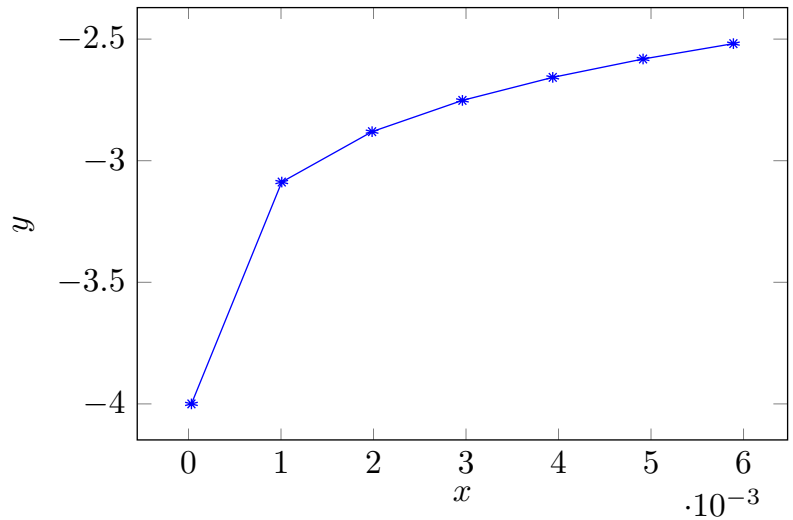
Feeding data

- We have seen two methods for feeding PGFPlots with data:
  - Provide points one by one as a list of `coordinates` (separator = space)
  - Defining a function in Ti*k*Z syntax (basically only y is provided)

There are several other methods :

- Use a table stored in a text file `mydata.dat` like below, loaded and plotted with : `\addplot[options] table {mydata.dat};`

```
x_0 f(x)
# some comment line
3.16693000e-05 -4.00001451e+00
1.00816962e-03 -3.08781504e+00
1.98466995e-03 -2.88058811e+00
2.96117027e-03 -2.75205040e+00
3.93767059e-03 -2.65736805e+00
4.91417091e-03 -2.58181091e+00
5.89067124e-03 -2.51862689e+00
```



- The `table` content can also be specified explicitely by `table [colums-identif] {data-in-table-form};` (see manual).

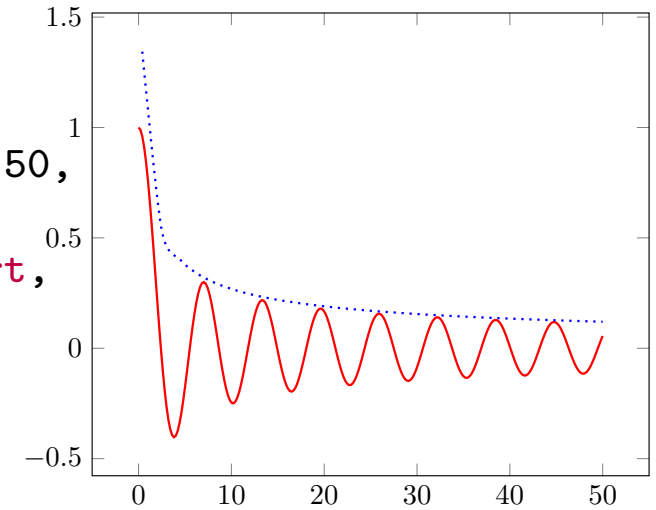
Feeding data, continued

- Provide a picture produced by an external software, as :

```
\addplot graphics [xmin=-3,xmax=3,ymin=-3,ymax=3]{external};
```

- Provide data in a CSV file, that can be heavily processed with the additional package `pgfplotstable`
- Get data from any external program, with `-shell-escape` option. Namely for `gnuplot` software (faster and more accurate than PGF fpu engine), there is a predefined function:

```
\begin{tikzpicture}
\begin{axis}[no markers]
\addplot gnuplot[red,thick,id=besselj0, domain=0:50,
 samples=250] {besj0(x)};
\addplot gnuplot [blue,dotted,thick,smooth,id=sqrt,
 domain=0.4:50] {0.85*x^{-0.5}};
\end{axis}
\end{tikzpicture}
```



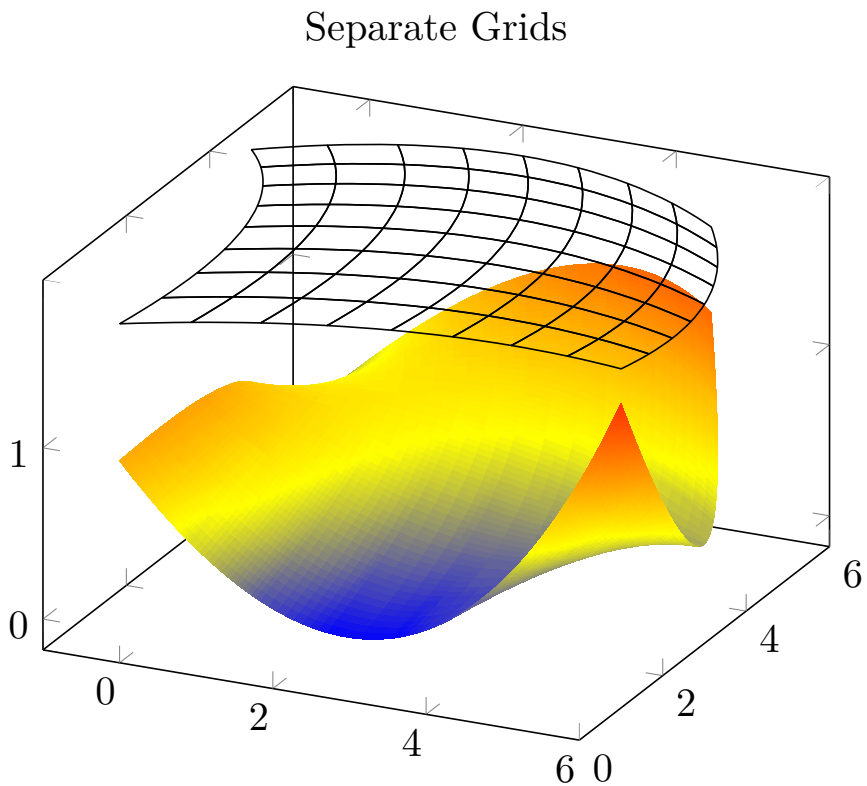
- Compute new columns on the basis of other plotted data

PGFPlots: customize apparency

- Define whole graphics size with `width=...` and/or `height=...`
- Define viewport (in scale units) with `ymin`, `ymax`, `xmin`, `xmax`
  (without, axes are autoscaled, with a boolean option `enlargelimits`)
- Rescale graphics (an not labels)l with `scale` and variants
- Factorize power of 10 on axis with e.g. `scaled x ticks`
- Add grid with: `grid=major`
- Add minor ticks with e.g.: `minor y tick num=3`
- Custom ticks labels with `xticklabel style={...}`
- Custom legend e.g.: `legend entries={{$d=2$},{$d=4$}}`
- Add error bars :
  - common absolute: `error bars/.cd, y fixed=0.1`;
  - common relative: `error bars/.cd, y fixed relative=0.1` (i.e. 10%);
  - explicit, with `+-(errx,erry)` after each point in `coordinates`;
  - or read them from `table` with option `x error`/`y error` in the list of columns.
- For any further information see the manual.

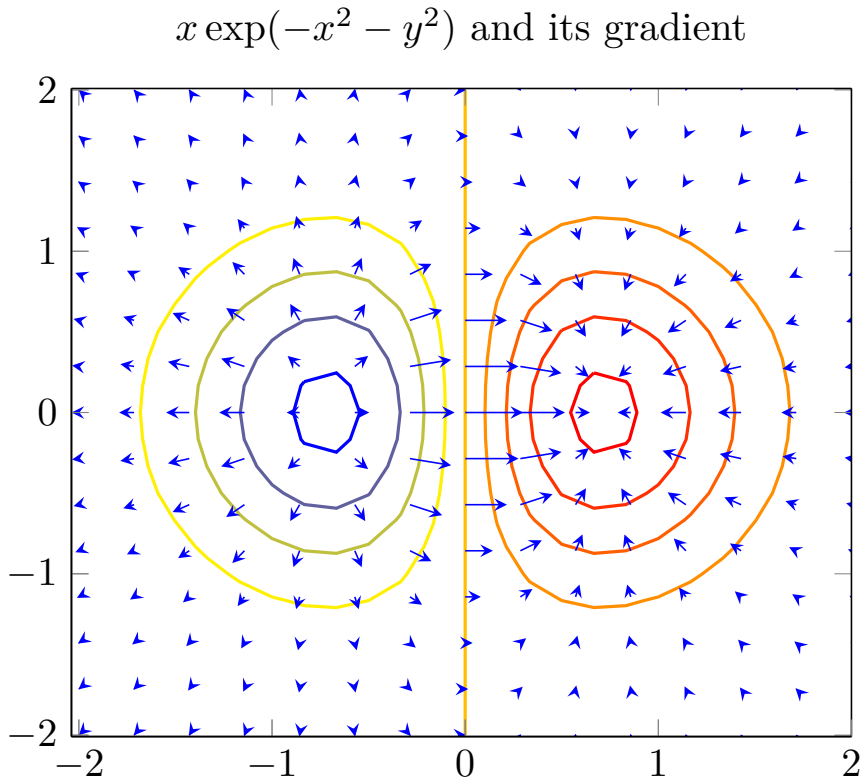
Some cool examples I/II

```
\begin{tikzpicture}
\begin{axis}[title={Separate Grids}]
\addplot3[patch,patch type=biquadratic,shader=interp, patch refines=3]
coordinates{(0,0,1) (6,1,1.6) (5,5,1.3) (-1,5,0) (3,1,0) (6,3,0.4) (2,6,1.1) (0,3,0.9)
\addplot3[patch,patch type=biquadratic, mesh,black,
z filter/.code={\def\pgfmathresult{1.8}}, patch refines=3]
coordinates{(0,0,1) (6,1,1.6) (5,5,1.3) (-1,5,0) (3,1,0) (6,3,0.4) (2,6,1.1) (0,3,0.9)
\end{axis}
\end{tikzpicture}
```



Some cool examples II/II

```
\begin{tikzpicture}
\begin{axis}[title={$x \exp(-x^2-y^2)$ and its gradient}, domain=-2:2,
view={0}{90}, axis background/.style={fill=white}]
\addplot3[contour gnuplot={number=9, labels=false},thick] {exp(-x^2-y^2)*x};
\addplot3[blue,quiver={ u={exp(-x^2-y^2)*(1-2*x^2)}, v={exp(-x^2-y^2)*(-2*x*y)},
scale arrows=0.3}, -stealth,samples=15] {exp(-x^2-y^2)*x};
\end{axis}
\end{tikzpicture}
```



Sommaire

Regulars floats : figure and table

A float is an environment placed by TEX at an optimized position (after its definition), trying to keep the page organization as clever as possible.

- LATEX defines two floats: figure and table. The difference is semantics, as their content can be any thing that doesn't concern page breaking. You can defines other floats with package floats.
- Example:

```
1       \begin{figure}[thbp]
2       \centering
3       \includegraphics[width=0.9\textwidth]{mafigure}
4       \caption{Ma belle figure}
5       \label{f-belle}
6       \end{figure}
```

- Floats have a *critical* option [tbhp] defining the allowed positions: [t]/[b] top/bottom of pages, [p] full page of float(s), [h] here.

Note the \includegraphics[]{} which the swiss-army-knife provided by graphicx to insert external content...

Placement

Placement of floats is the worst headache that LaTeX users can experience.

- The best position is generally `top`, but what ever you choose, it will interfere with page breaking control, and float the floats to the end.
- [h] & [!h] have *badness* maximal and must be avoided.
- To ensure that floats are not floated to the end one can:
  - Allows LaTeXto be more tolerant about floats placement with:

```
\renewcommand{\topfraction}{0.9} % max fraction at top
\renewcommand{\bottomfraction}{0.6} % max fraction at bottom
\renewcommand{\textfraction}{0.1} % minimal text w. figs
\setcounter{topnumber}{2}\setcounter{bottomnumber}{2} % floats/page
```

  - Force a float page (`p`) with command `\afterpage{\clearpage}`.
  - Limit floating with command `\FloatBarrier` of package `placeins` : Option `section` automatically adds `\FloatBarrier` to `\section`.
  - use [!tbph] to relax all subtle constraints.

More : "How to influence the position of float environments..."

Floats (not floating) in text

- Packages `wrapfig`, `picins` and `floatflt` allow to place small floating figures inside text as shown on next slide.
- Specify the requested width, and optionally the placement (r/l).
- Example : The figure is produced by :

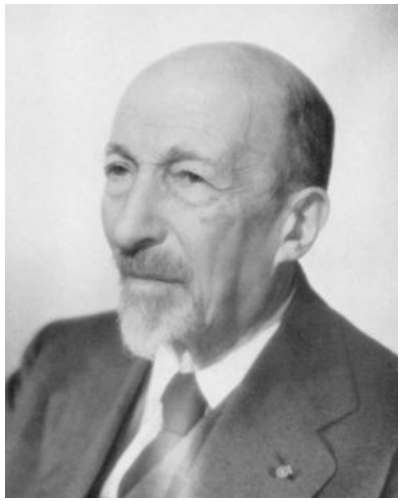
```
L'année suivante, il entra premier à l'École  [...].
\subsection{Carrière d'universitaire}
\begin{floatingfigure}[r]{40mm}
\flushright
\includegraphics[width=35mm]{hadamard-pic}
{\centering Jacques \textsc{Hadamard}\par}
\end{floatingfigure}
En 1889, il enseigna au lycée Saint-Louis  [...]
```

- These floats conflict with lists, and often with sectioning commands.
- `wrapfig` is the most popular, but `wrapfig`, `picins` are claimed as obsolete, and it could be better to use `floatflt` .

Using `floatflt`

L'année suivante, il entra premier à l'École normale supérieure.

1.2 Carrière d'universitaire



Jacques Hadamard

En 1889, il enseigna au lycée Saint-Louis puis à partir de 1890 au Lycée Buffon. Il eut comme élève Maurice Fréchet et eut des contacts avec Émile Borel à l'École normale, jusqu'au départ de ce dernier pour la faculté des sciences de Lille en 1893. Il obtint son doctorat en 1892, sous la direction d'Émile Picard, pour des recherches sur les fonctions définies par séries de Taylor.

Il enseigne alors à la faculté des sciences de l'université de Bordeaux en tant que chargé de cours de juillet 1893 à février 1896 puis professeur titulaire. Il retourna ensuite à Paris en tant que maître de conférences (en remplacement de Paul Painlevé à la faculté des sciences de l'université de Paris, et obtient le titre de professeur adjoint en février 1900. En novembre 1897, il devient également suppléant de Maurice Lévy à la chaire de mécanique analytique et mécanique céleste du Collège de France (à la suite de Paul Painlevé).

Captions

- Floats generally include captions, explaining the content, defined by:

```
\begin{figure}[htbp]
\includegraphics[width=35mm]{hadamard-pic}
\caption[Portrait de Jacques Hadamard] % Short title for \lof
{Jacques \textsc{Hadamard}, photographie prise en 1898 à \ldots}
\end{figure}
```

- Genuine captions for “non-floating” illustration (with numbering etc.):

```
\usepackage{caption} % option [hypcap=true] will be recquired  latter
[...]
\begin{minipage}{14cm}
\includegraphics[width=35mm]{hadamard-pic}
\captionof{figure}{Portrait de Jacques Hadamard}
\end{minipage}
```

- Package `caption` enables customization with, for example:

```
\captionsetup[figure]{labelsep=endash,labelfont={rm,bf},%
   textfont=sl,font=small}
```

which can be set globally or inside a given `figure`.

Subcaptions

- Package subcaption allows captioning if composite figures :

```
\usepackage{caption}
\usepackage{subcaption}
[...]
\begin{figure}
  \begin{subfigure}[t]{0.7\textwidth}
    \includegraphics[width=0.7\textwidth]{example-image-a}
    \caption{of subfig a}
  \end{subfigure}
  \begin{subfigure}[t]{0.49\textwidth}
    \includegraphics[width=0.7\textwidth]{example-image-b}
        \caption{of subfig b}
  \end{subfigure}
  \caption{of whole figure}
\end{figure}
```

- The subfigure environment is defined in subcaption, but is not mandatory: any grouping is sufficient.
- subfigure & subfig : obsolete and incompatible with hyperref.