

# Intermediate Latex

## Using Graphics in LaTeX

Jean Hare

Sorbonne Université  
Laboratoire Kastler Brossel  
ED Physique en Île-de-France  
[jean.hare@lkb.ens.fr](mailto:jean.hare@lkb.ens.fr)

Support documents at  
<https://www.edpif.org/documents/latex/intermediate>

April 2024

# Outline

- 1 Introduction
- 2 Essential Tools
- 3 Producing pictures
- 4 A little tour on the Inkscape side
- 5 An introduction to PGF and TikZ
- 6 PGFPlots
- 7 Floats and captions

# Sommaire

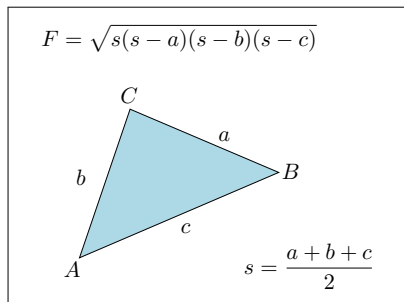
- 1 Introduction
- 2 Essential Tools
- 3 Producing pictures
- 4 A little tour on the Inkscape side
- 5 An introduction to PGF and TikZ
- 6 PGFPlots
- 7 Floats and captions

# Initial L<sup>A</sup>T<sub>E</sub>X

- The problem of graphics in (La)TeX is a very long lasting one.
- Some primitive attempts: the L<sup>A</sup>T<sub>E</sub>X `picture` environment (with extensions `epic`, `epic`, `bezier`, `overpic`, `pict2e` ...)

A `pict2e` sample:

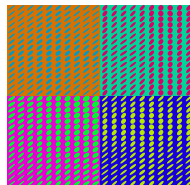
```
\usepackage{pict2e}
\begin{document}
\setlength{\unitlength}{1cm}
\fbbox{
\begin{picture}(6,4.5)
  \put(0,0){\color{LightBlue}
    \polygon*(0.9,0.65)(4.05,2.)(1.7,3)}}
  \put(0,0){
    \polygon(0.9,0.65)(4.05,2.)(1.7,3)}
  \put(0.65,0.35){$A$} \put(4.1,1.9){$B$}
  \put(1.55,3.1){$C$}
  \put(3.1,2.5){$a$} \put(0.85,1.8){$b$}
  \put(2.5,1.05){$c$}
  \put(0.3,4){$\displaystyle
    F=\sqrt{s(s-a)(s-b)(s-c)}$}
  \put(3.5,0.4){$\displaystyle
    s=\frac{a+b+c}{2}$}\end{picture}}
\end{document}
```



# PostScript & new libraries

- For many years, the best solution was to create graphics with external software and include **eps** (or since 2017 **pdf**) with `\includegraphics`.

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 552 296 840
%%Title: C:\vasarely.eps
%%Creator: GSview
%%CreationDate: Thu Apr 06 09:53:24 2023
%%Pages: 1
%%EndComments
%%Page: 1 1
```



- Various TEX-based solutions were developed later. Most important are:
  - PSTricks**, macros for inclusion of PostScript drawings inside TeX or LaTeX code, in a `pspicture` environment. Postscript programming. For pdfTeX, requires package `auto-pst-pdf`). Very popular, tens of extensions. Website: <http://tug.org/PSTricks>
  - PGF/TikZ** compatible with both traditional TeX/LaTeX and with pdf(La)TeX. Growing popularity, tens of extensions. Website: <https://github.com/pgf-tikz/pgf>

# Sommaire

- 1 Introduction
- 2 Essential Tools**
- 3 Producing pictures
- 4 A little tour on the Inkscape side
- 5 An introduction to PGF and TikZ
- 6 PGFPlots
- 7 Floats and captions

# Basic Tools

- Package **graphicx** and its `\includegraphics[options]{filename}` is the basic tool to include pictures produced by an external software:
    - For pdf $\LaTeX$ , with PDF output, allowed formats are bitmaps JPG, PNG and vectorial/bitmap PDF. On loading **graphicx** don't set **driver** option, in `\includegraphics` don't write extension.
    - For  $\LaTeX$  with DVI output, allowed formats are EPS, and JPG.
  - For pdf $\LaTeX$ , EPS is converted to PDF by **GhostScript**-based tools\*:
    - by CLI programs, like **epstopdf** (.exe or .pl)
    - by GUI programs, like **GSView**
    - in pdf $\LaTeX$  run, package **epstopdf** converts them **on the fly** to PDF.
  - Most used options of `\includegraphics` provide
    - size with **width=** or **height=** (or both) or **scale=**,
    - trimming with **trim=left bottom right top,clip**,
    - rotation with **angle=** and **origin=**, select page in pdf with **page=**.
  - Global path to files set by e.g. `\graphicspath{{../img/}}{fig2/}}`
- \*ghostscript (**gs** or **mgs.exe**) is already present in your OS/distribution.

# Other options

- The size of the graphics can be automatically adjusted by using `adjustbox` loaded with option `export` (or `Export`)
  - `export` redefines `\includegraphics` as `\adjincludegraphics`
  - `\adjincludegraphics` introduces many additional keys; the most useful for size control are `min width`, `max width`, `min height`, `max height`, `min totalheight`, `max totalheight`.
- For both, the dimensions provided to the `width` or other size options can be defined by using  $\epsilon$ -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`,...



# Sommaire

- 1 Introduction
- 2 Essential Tools
- 3 Producing pictures**
- 4 A little tour on the Inkscape side
- 5 An introduction to PGF and TikZ
- 6 PGFPlots
- 7 Floats and captions

# Producing pictures

- For diagrams and curves, use a **vector format** and for images prefer **pdf**, or **png** by default.
- Use neither **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 the old **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 instead the script **jpeg2ps**, or the more powerful **sam2p**
  - If you need to annotate it, prefer **Inkscape** or TikZ. Don't export to **jpeg**.
  - We often turn to **Photoshop**, **GIMP** and **ImageMagick**, but don't neglect that you can do much more by using **ImageJ** or **IGOR Pro** which are *scientific* softwares.
- Tools for TikZ will be presented below.

# Fonts problems in figures

The **main problems** with figures come from the fonts.

- 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 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” or in French «jeu partiel incorporé»).

# Fonts problems in figures : solutions

- You could 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` uses a personalized preamble and memorizes the  $\text{\LaTeX}$  formulas used for easier re-editing.
- To embed fonts *a posteriori* you could 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"
```

Adapt: On line 1, the path to `Ghostscript` and to the `TeX` distribution `type1` fonts, and on lines 5 the target filenames or paths.

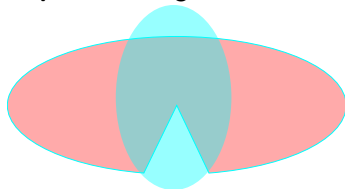
- In desperate cases: [this post](#) or Acrobat Pro ...

# Sommaire

- 1 Introduction
- 2 Essential Tools
- 3 Producing pictures
- 4 A little tour on the Inkscape side
- 5 An introduction to PGF and TikZ
- 6 PGFPlots
- 7 Floats and captions

# Importing SVG

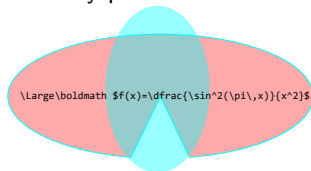
- pdf $\text{\LaTeX}$  does not natively support SVG, (internal format of **Inkscape**)
- **Inkscape** can **export** to useful formats: PDF, PDF+latex, PGF/TikZ.
- In the (pure) PDF route, one gets a PDF file ready to include, but fonts...
- The PDF+latex, provides two files : **mypic.pdf** file and **mypic.pdf\_tex**
  - The PDF **mypic.pdf** file contains only the drawings.
  - The **mypic.pdf\_tex** is a  $\text{\LaTeX}$  file with a **picture** environment, which contains the text, to be typeset by  $\text{\LaTeX}$ , and an `\includegraphics` of **mypic.pdf**
  - The main documents simply uses `\input{mypic.pdf_tex}`
- **Example:** Wanting to annotate a graphic (a dummy colored shape)



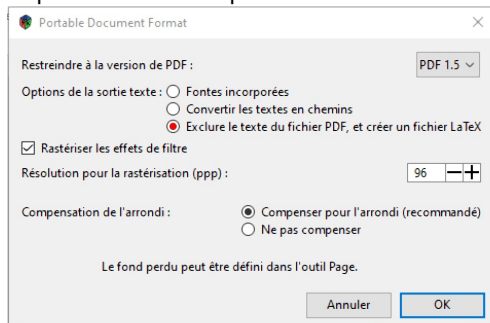
with 
$$f(x) = \frac{\sin^2(\pi x)}{x^2}$$

# PDF-Latex route, continued

- 1 Insert the  $\text{\LaTeX}$  code in a textbox of Inkscape, carefully positioned :



- 2 Export it with the options :



- 3 The file `.pdf_tex` ends with:

```

53 \begin{picture}(1,0.54576462)%
54 \lineheight{1}%
55 \setlength\tabcolsep{0pt}%
56 \put(0,0){\includegraphics[width=\unitlength,page=1]{inkscape-export-pdf-LaTeX.pdf}}
57 \put(0.51560458,0.24196573){\makebox(0,0)[t]{\lineheight{1.25}\smash{%
58 \begin{tabular}[t]{c}%
59 \Large\boldmath $\displaystyle f(x)=\frac{\sin^2(\pi x)}{x^2}$%
60 \end{tabular}}}%
61 \end{picture}%

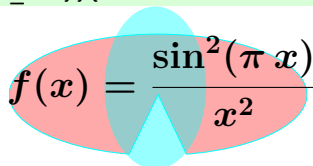
```

# PDF Latex route, again

- 4 Use, possibly in a `figure` environment, a code like:

```
1 \def\filename{mypic}
2 \def\svgwidth{0.5\linewidth}
3 \IfFileExists{\filename.pdf_tex}%
4 {\input{\filename.pdf_tex}}{Error on \filename}
```

- 5 With the result :



$$f(x) = \frac{\sin^2(\pi x)}{x^2}$$

- 6 To automate this process, after line 1 (with `-shell-escape`):

```
\immediate\write18{%
  inkscape -D \filename.svg -o \filename.pdf --export-latex}
```

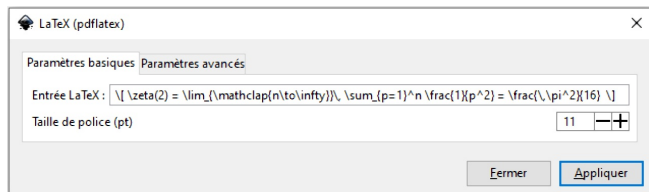
- 7 If often used, a better solution would be to define for this purpose a macro that also performs a check for modifications date. This mostly what the package `svg` does. It automates a lot of things, at the price of some additional complexity...

For further examples and explanations have a look at `svg-inkscape` package.



# The PDF-only route

- In this route, one can embed **outlines** rendered by  $\text{\LaTeX}$ , as described here.
- Inkscape** includes an extension to typeset  $\text{\LaTeX}$  texts and mathematics.  
Access by: **Extensions»Render»Mathematics»LaTeX**  
Or **Extensions»Text»LaTeX (pdflatex)**. This opens a popup where  $\text{\LaTeX}$  code can be entered.

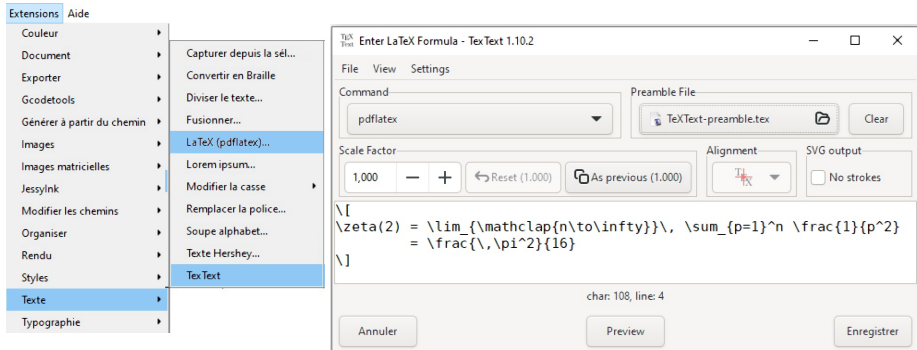


- It calls **pdflatex** to create a vectorial and editable version of the result.

$$\zeta(2) = \lim_{n \rightarrow \infty} \sum_{p=1}^n \frac{1}{p^2} = \frac{\pi^2}{16} \quad \Rightarrow \quad \zeta(\textcolor{red}{2}) = \lim_{n \rightarrow \infty} \sum_{p=1}^n \frac{1}{\textcolor{green}{p}^2} = \frac{\pi^2}{\textcolor{violet}{16}}$$

# Annotate with $\text{\LaTeX}$ in Inkscape using TeXText

- 1 With **Inkscape** 1.1 and 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
  - Remembers each of the previous formulas, for easier reediting
  - In case of errors, displays the pdf<sub>l</sub>atex terminal output.
  - Once installed, access by: **Extensions** » **Text** » **TeX Text**



# Sommaire

- 1 Introduction
- 2 Essential Tools
- 3 Producing pictures
- 4 A little tour on the Inkscape side
- 5 An introduction to PGF and TikZ**
- 6 PGFPlots
- 7 Floats and captions

# Installation and documentation

- PGF means “Portable Graphics Format” and is a low level language.
- TikZ 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 3.1.10) is 1320 pages long !!
- The release contains also an 24 pages PDF entitled “Minimal Introduction to Tikz”
- A good introduction “en français” entitled “TikZ 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/TikZ.

# First TikZ 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}
```



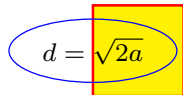
- Close polygon with `--cycle` and add some colors:

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



- A standalone one produced by `\node`, centered on coordinates (0,0.5) and exhibiting a LaTeX formula. Another, inside `\draw`, with `node` (no `\`), attached to point (0,1) and offset by `below right`.

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



**Note:** To be able to draw the ellipse, one should have in the preamble:

```
\usetikzlibrary{shapes}
```

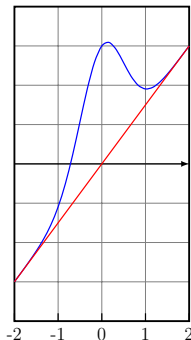
# First TikZ plot

- If `babel` was loaded with `french` option, disable active characters with:  
`\shorthandoff{?!:;}` or better `\usetikzlibrary{babel}`
- Define a function in `tikzpicture` option and plot it:

```

1 \begin{tikzpicture}[thick,yscale=0.9,domain=-2:2,
2   declare function={
3     a=1.5; w=2; % constants=functions w/o argument,
4     f(\x)=a*(\x+2*exp(-w*\x*\x));
5   }]
6 \draw[help lines,xstep=1](-2,-4) grid (2,4); % grid
7 \draw[very thick] (-2,-4) rectangle (2,4); % frame
8 \draw[->,>=latex] (-2,0)--(2,0);
9 \foreach \x in {-2,...,2} {% tick marks
10   \draw (\x,-3.8)--(\x,-4) node[below] {\x};
11 }
12 \draw[blue] plot [smooth] (\x,{f(\x)});
13 \draw[red] plot [samples=2] (\x,a*\x);
14 \end{tikzpicture}

```



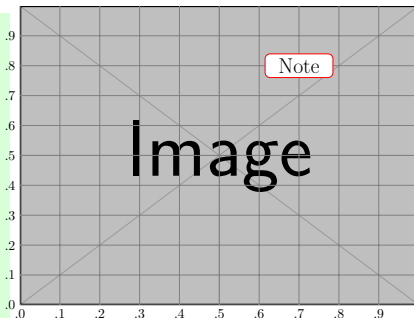
# Using TikZ for annotations (see also [this website](#))

How to place an annotation at a defined position of an image ?

```

1 \tikzset{annot/.style={draw=red,fill=white,
2   thick,rounded corners, minimum width=5em}}
3 \begin{tikzpicture}
4 \node[inner sep=0pt] (img) at (0,0) {
5 \includegraphics{example-image}};
6 \begin{scope}[shift=(img.south west),
7   x={(img.south east)},y={(img.north west)}]
8 \draw[help lines,step=.1] (0,0) grid (1,1);
9 \foreach \x in {0,...,9} { % ticks
10   \node [below] at (\x/10,0) {\x};
11   \node [left] at (0,\x/10) {\x}; }
12 \node[annot] at (0.7,0.8) {\LARGE Note};
13 \end{scope}
14 \end{tikzpicture}

```



- Lines 4 & 5: place image in a node names (**img**) with LL corner at origin
- Line 6 to 14: using a scope to define and use scaled coordinates
- Line 7: set origin of new coordinates to LL corner of image (**img.south west**)
- Line 9-12: draw a (temporary) grid on image, with ticks marked by lines 10-12
- Line 13: place annotation at (0.7,0.8) scaled coordinates, styled by line 2-3.

# Tools making TikZ more efficient

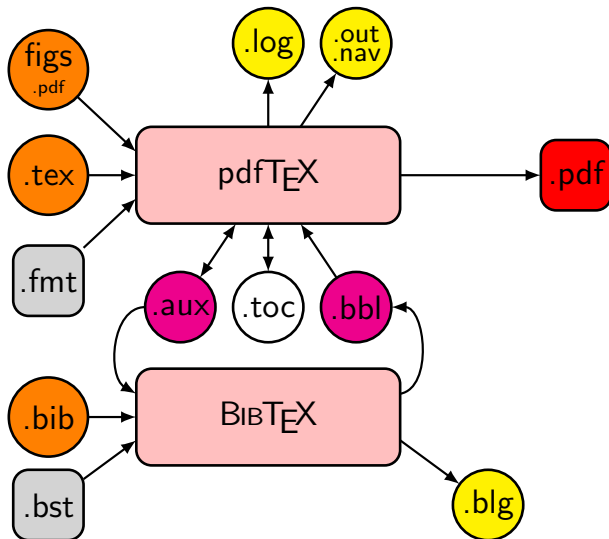
Here is a small list of open source softwares more or less TikZ oriented.

- **TikZiT** (multiplatform) 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. The most efficient to learn TikZ without too much effort.
- Other softwares can export PGF/TikZ code 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


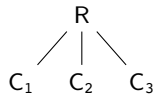


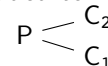
## Compilation chain: the code

```
1 % styles
2 \tikzset{mythick/.style={draw=black,line width=0.3mm,font=\sffamily,align=center},
3 motor/.style={minimum width=3cm,minimum height=1.1cm,rectangle,rounded corners=5pt,
4 fill=pink,font=\bfseries\sffamily,mythick},
5 circbx/.style={mythick,circle,minimum width=#1,minimum height=#1},
6 rectbx/.style={mythick,rectangle,minimum width=#1,minimum height=#1, rounded corners=5pt},
7 arr/.style={line width=0.25mm,>=latex}}
8 \begin{tikzpicture}[x=0.50mm, y=0.50mm, inner sep=0pt, outer sep=0pt,]
9 % pdflatex and files
10 \node[motor] (pdfTeX) at (60,0) {\hologo{pdfTeX}};
11 \node[circbx=9mm,fill=orange] (fTeX) at (10,0) {.tex};
12 \node[circbx=9mm,fill=orange] (ffigpdf) at (10,24) {figs\[-1ex\]{\tiny.pdf}};
13 \node[rectbx=8mm,fill=LightGray] (ffmt) at (10,-24) {.fmt};
14 \node[rectbx=8mm,fill=red] (fpdf) at (130,0) {.pdf};
15 \node[circbx=8mm,fill=magenta] (faux) at (40,-30) {.aux};
16 \node[circbx=8mm,fill=white] (ftoc) at (60,-30) {.toc};
17 \node[circbx=8mm,fill=magenta] (fbbl) at (80,-30) {.bbl};
18 \node[circbx=8mm,fill=yellow] (flog) at (60,30) {.log};
19 \node[circbx=8mm,fill=yellow,font=\footnotesize\sffamily] (fout)
20 at (80,30) {.out\[-5pt\].nav};
21 % pdflatex arrowsG
22 \draw[arr,->] (ffmt) -- (pdfTeX.190); \draw[arr,->] (fTeX) -- (pdfTeX);
23 \draw[arr,->] (ffigpdf) -- (pdfTeX.170);
24 \draw[arr,->] (pdfTeX) -- (flog); \draw[arr,->] (fbbl) -- (pdfTeX);
25 \draw[arr,->] (pdfTeX) -- (fpdf); \draw[arr,->] (pdfTeX) -- (fout);
26 \draw[arr,<->] (faux) -- (pdfTeX); \draw[arr,<->] (ftoc) -- (pdfTeX);
27 % bibtex and files
28 \node[motor] (bibtex) at (60,-55) {\hologo{BibTeX}};
29 \node[circbx=9mm,fill=orange] (fbib) at (10,-55) {.bib};
30 \node[rectbx=8mm,fill=LightGray] (fbst) at (10,-75) {.bst};
31 \node[circbx=8mm,fill=yellow] (fblg) at (110,-75) {.blg};
32 % bibtex arrows
33 \draw[arr,->] (faux.west) to [out=180, in=150] (bibtex.170);
34 \draw[arr,->] (fbib) -- (bibtex); \draw[arr,->] (fbst) -- (bibtex.190);
35 \draw[arr,->] (bibtex.10) to [out=10, in=0] (fbbl.east);
36 \draw[arr,->] (bibtex.350) -- (fblg);
37 \end{tikzpicture}
```

# Drawing trees with the `forest` package

- Basic syntax uses (square) brackets `[ ]` to define nodes
  - The `[ ]` starts with the node name, that will be printed,
  - Then, insert in it the child nodes, which can be nested at will.

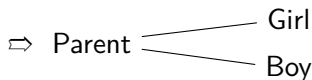
code	<code>[R]</code>	<code>[R[C_1][C_2]]</code>	<code>[R[C_1][C_2][C_3]]</code>
result	R	 <pre>       R      / \     C1  C2           </pre>	 <pre>       R      /   \     C1 C2 C3           </pre>

- The default setting can be adjusted by options added to `forest` :
  - `grow=` to change the direction; e.g. `grow=east`  $\Rightarrow$ 

  - `name=` to change the text, e.g. `[Parent, name=P]` `[G][B]`
  - `l sep=`/`s sep=` to set the distance in the average/perpendicular direction.
  - Finally

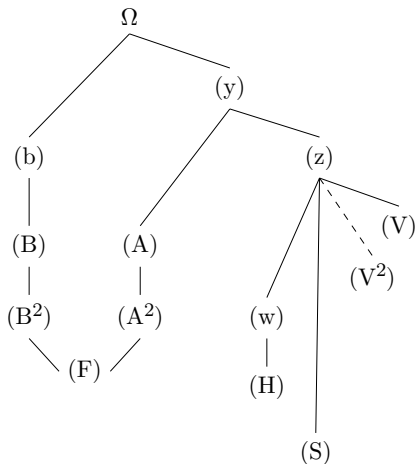
```

\begin{forest}
  grow=east, l sep=1.5cm, s sep=0.5ex
  [Parent, name=P
    [Boy, name=B]
    [Girl, name=G]
  ]
\end{forest}

```



# A *stemma codicum* with **forest** package



```

\begin{document}
\forestset{sn edges/.style={for tree=%
  {parent anchor=south,child anchor=north}}}
\begin{forest}
[Ω, sn edges, grow=south,
  l sep=0.5cm, s sep=0.7cm
  [(b),tier=second, l sep=0.7cm, name=B1
    [(B), tier=AB, name=B
      [(B²), tier=AB2,name=B2]
    ]
  ]
  [(y), s sep=1cm,
    [(A), tier=AB,name=A
      [(A²), tier=AB2, name=A2]
    ]
    [(z),tier=second, name=Z
      [(w),tier=AB2,name=W [(H), name=H
        [ ,phantom, tier=last]]
      ]
      [(S), name=S,tier=last]
      [ , phantom]
      [(V²), edge= dashed, name=V2, below]
      [(V), above=12pt, right,name=V]
    ]
  ]
]
\end{forest}
\node[yshift=-8mm] (F) at ($(A2)!0.5!(B2)$) {(F)};
\draw (A2.south)--(F.east) (F.west)--(B2.south) ;
\end{document}

```

# Sommaire

- 1 Introduction
- 2 Essential Tools
- 3 Producing pictures
- 4 A little tour on the Inkscape side
- 5 An introduction to PGF and TikZ
- 6 PGFPlots**
- 7 Floats and captions

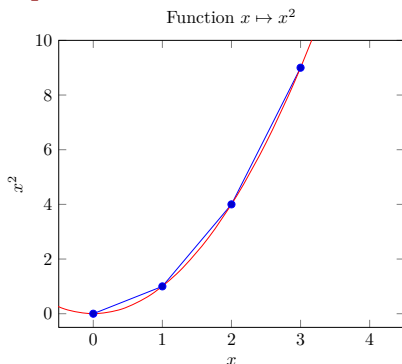
# PGFPlots

- PGFPlots is a powerful PGF-based package, for data representation (2D or 3D).
- The manual is large (573 pages), but contains very good tutorials.
- Load is with `\usepackage{pgfplots}` and `\pgfplotsset{compat=1.18}`
- It uses an `axis` environment inside a `tikzpicture`, as follows:

```

1 \begin{tikzpicture}
2 \begin{axis}[
3   title={Function  $x \mapsto x^2$ },
4   xlabel={ $x$ }, ylabel={ $x^2$ },
5   xmin=-0.5, xmax=4.5,
6   ymin=-0.5, ymax=10, yscale=1
7 ]
8 \addplot coordinates
9   {(0,0) (1,1) (2,4) (3,9)};
10 \addplot [red,smooth] {x^2};
11 \end{axis}
12 \end{tikzpicture}

```



This example shows how to plot data points or function, and how to set labels.

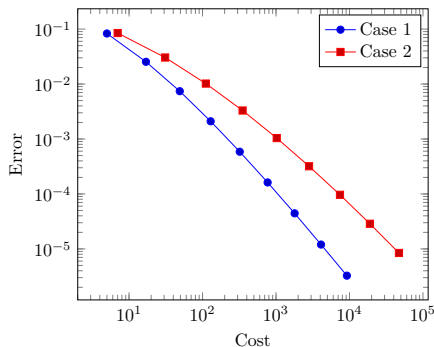
# Log scale, legend

One can also use log/log or semi-log axes, add automatic legend, etc

```

1 \begin{tikzpicture}
2 \begin{loglogaxis}[
3   xlabel=Cost, ylabel=Error
4 ]
5 \addplot coordinates {
6   (5, 8.31e-02) (17, 2.54e-02)
7   (49, 7.41e-03) (129, 2.10e-03)
8   (321, 5.87e-04) (769, 1.62e-04)
9   (1793, 4.44e-05) (4097, 1.20e-05)
10  (9217, 3.26e-06) };
11 \addplot coordinates {
12  (7, 8.47e-02) (31, 3.04e-02)
13  (111, 1.02e-02) (351, 3.30e-03)
14  (1023, 1.04e-03) (2815, 3.19e-04)
15  (7423, 9.66e-05) (18943, 2.87e-05)
16  (47103, 8.44e-06) };
17 \legend{Case 1, Case 2}
18 \end{loglogaxis}
19 \end{tikzpicture}

```



example from the manual

# Feeding data

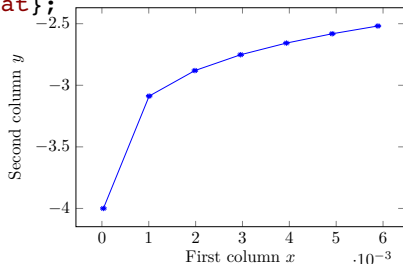
- We have seen two methods for feeding PGFPlots with data:
  - Provide points one by one as a list of `coordinates` (separator = space)
  - Defining a function in TikZ syntax ( basically only  $y$  is provided)

There are several other methods :

- Use a table stored in a text file `mydata.dat` like below :

```
\addplot[options] table {mydata.dat};
```

```
# mydata.dat
x_0 f(x)
# some comment line
3.16693e-05 -4.00001e+00
1.00816e-03 -3.08781e+00
1.98466e-03 -2.88058e+00
2.96117e-03 -2.75205e+00
3.93767e-03 -2.65736e+00
4.91417e-03 -2.58181e+00
5.89067e-03 -2.51862e+00
```



- The `table` content can also be specified explicitly by
 

```
table [columns-identif] {data-in-table-form};
```

 (see manual).



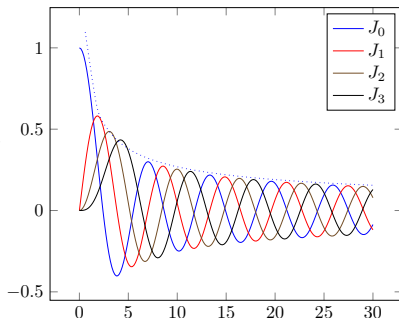
# Feeding data, continued

- 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), there is a predefined function:

```
\begin{tikzpicture}
\begin{axis}[no markers]
\pgfplotsinvokeforeach{0,1,2,3}{%
  \addplot+ gnuplot[id=besselj#1,
    domain=0:30,samples=200] {besjn(#1,x)};
  \addlegendentry{$J_{\#1}$}%
}
\addplot gnuplot [blue,dotted,smooth,
  id=sqrt, domain=0.6:30] {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.

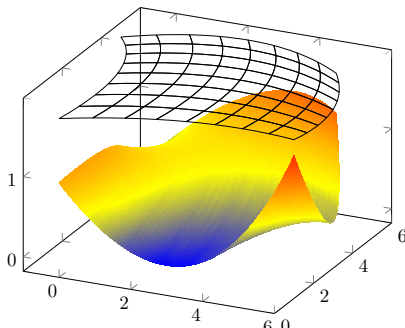
# Some cool examples I/II

```

\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}

```

Separate Grids



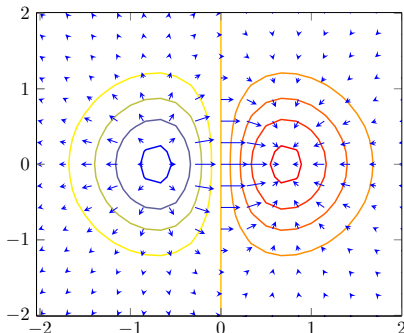
# Some cool examples II/II

```

\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}

```

$x \exp(-x^2 - y^2)$  and its gradient



# Sommaire

- 1 Introduction
- 2 Essential Tools
- 3 Producing pictures
- 4 A little tour on the Inkscape side
- 5 An introduction to PGF and TikZ
- 6 PGFPlots
- 7 Floats and captions**

# Regulars floats : `figure` and `table`

A float is an environment placed by  $\text{\TeX}$  at an optimized position (after its definition), trying to keep the page organization as clever as possible.

- $\text{\LaTeX}$  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 `[thbp]` 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  $\text{\LaTeX}$  users can experience.

- The best position is generally `tbp`, but what ever you choose, it will interfere with page breaking control, and floats the floats to the end.
- `[h]` has *badness* maximal and must be avoided, but some times `!tbph` could be used to relax all subtle constraints.
- To ensure that floats are not floated to the end one can:
  - Allows  $\text{\LaTeX}$  to be more tolerant about floats placement with:

```
\renewcommand{\topfraction}{0.80} % max fraction at top
\renewcommand{\bottomfraction}{0.75} % max fraction at bottom
\renewcommand{\textfraction}{0.15} % minimal text w. figs
\renewcommand{\floatpagefraction}{0.4} % minimum fraction of floatpage
\setcounter{topnumber}{2}\setcounter{bottomnumber}{2} % floats/page
\setcounter{totalnumber}{4}
```

- Force a float page (`p`) with command `\afterpage{\clearpage}`.
- Limit floating with command `\FloatBarrier` of package `placeins` :  
Option `section` automatically adds `\FloatBarrier` to `\section`.

# 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 :

```

1      ' Lanne suivante, il entra premier lcole' [...].
2      \subsection{'Carriere d'universitaire'}
3      \begin{floatingfigure}[r]{40mm}
4      \flushright
5      \includegraphics[width=35mm]{hadamard-pic}
6      {\centering Jacques \textsc{Hadamard}\par}
7      \end{floatingfigure}
8      'En 1889, il enseigna au lyce 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

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).



*Jacques* HADAMARD

# Captions

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

```

1 \begin{figure}[htbp]
2 \includegraphics[width=35mm]{hadamard-pic}
3 \caption[Portrait de Jacques Hadamard] % Short title for \lof
4 {Jacques \textsc{Hadamard}, 'photographie prise en 1898}
5 \end{figure}

```

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

```

1 \usepackage{caption} % option [hycap=true] will be required latter
2 [...]
3 \begin{minipage}{14cm}
4 \includegraphics[width=35mm]{hadamard-pic}
5 \captionof{figure}{Portrait de Jacques Hadamard}
6 \end{minipage}

```

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

```

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

```

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

# Subcaptions

- Package `subcaption` allows captioning if composite figures :

```

1 \usepackage{caption}
2 \usepackage{subcaption}
3 [...]
4 \begin{figure}
5   \begin{subfigure}[t]{0.7\textwidth}
6     \includegraphics[width=0.7\textwidth]{example-image-a}
7     \caption{'subfig a'}
8   \end{subfigure}
9   \begin{subfigure}[t]{0.49\textwidth}
10    \includegraphics[width=0.7\textwidth]{example-image-b}
11    \caption{'subfig b'}
12  \end{subfigure}
13  \caption{'for whole figure'}
14 \end{figure}

```

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